EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

Access Point

Administration Guide

Domibus 5.1.3

Version [20.9]

Status [Final]

Date: 05/04/2024

Table of Contents

1. INTRODUCTION 9
1.1. Purpose 9
1.2. References 9

2. CONVENTIONS 11
2.1. Example 1: Sample Oracle Statement 11
2.2. Example 2: Sample Configuration file 11

3. PREREQUISITES 12
3.1. Binaries repository 12

4. DOMIBUS DEPLOYMENT 13
4.1. Database Configuration 13
4.1.1. MySQL configuration 13
4.1.2. Oracle configuration 15
4.1.3. MySQL and Oracle Deletion scripts 16
4.2. Domibus on WebLogic 17
4.2.1. Single Server Deployment 18
4.2.2. Clustered Deployment 28
4.3. Domibus on Tomcat 40
4.3.1. Pre-Configured Single Server Deployment 40
4.3.2. Single Server Deployment 43
4.3.3. Clustered Deployment 45
4.4. Domibus on WildFly 47
4.4.1. Pre-Configured Single Server Deployment 47
4.4.2. Single Server Deployment 54
4.4.3. Clustered Deployment 61
4.5. Domibus Secure Deployment Recommendations 66
4.5.1. Introduction 66
4.5.2. Recommendations 66

5. DOMIBUS CONFIGURATION 68
5.1. Security Configuration 68
5.1.1. Security Policies 68
5.1.2. Default authorization 69
5.1.3. Certificates 70
5.1.4. Security Profiles 71
5.2. Domibus Properties 74
5.2.1. Password encryption 112

6. PLUGIN MANAGEMENT 113
6.1. Default Plugins 113
6.1.1. JMS Plugin 113
6.1.2. WS Plugin 113

6.1.3. File System Plugin 113
6.2. Custom Plugin 113
6.2.1. Plugin registration 113
6.3. Plugin authentication 114
6.4. Plugin notifications 115
6.5. Plugin Ehcache files 115

7. PMODE CONFIGURATION 116
7.1. Configuration 116
7.1.1. Adding a new participant 116
7.1.2. Sample PMode file 117
7.1.3. Domibus PMode configuration to ebMS3 PMode Mapping 121
7.1.4. Upload new Configuration 126
7.1.5. Message properties validation 131

8. TWO-WAY MEP SCENARIO 133
8.1. PushAndPush binding 134
8.2. PushAndPull binding 135
8.3. PullAndPush binding 136

9. SPECIAL SCENARIO: SENDER AND RECEIVER ARE THE SAME 138
9.1. PMode Configuration 138
9.2. Message structure 138
9.3. Message ID convention 139

10. ADMINISTRATION TOOLS 140
10.1. Administration Console 140
10.2. Multitenancy 141
10.3. Message Log 142
10.4. Message Filtering 143
10.5. Application Logging 145
10.5.1. Domibus log files 145
10.5.2. Logging properties 146
10.5.3. Error Log page 147
10.6. PMode 148
10.7. Queue Monitoring 149
10.8. Configuration of the queues 157
10.8.1. Tomcat 157
10.8.2. WebLogic 158
10.8.3. WildFly 158
10.9. Certificates 158
10.9.1. saveCertificateAndLogRevocationJob 160
10.10. Users 160
10.10.1. Adding new users 161
10.10.2. Changing passwords 162

10.10.3. User Account Lockout Policy 165
10.11. Plugin Users 166
10.12. Audit 167
10.13. Alerts 168
10.13.1. Example: Alerts on SEND_FAILURE 169
10.14. Connection Monitoring 169
10.15. Logging 171
10.16. Domains 171
10.17. Properties 171

11. LARGE FILES SUPPORT 173
11.1. Split and Join 173

12. DATA ARCHIVING 175
12.1. What's archiving? 175
12.2. Data Retention Policy 175
12.3. Data Extraction 176

13. DATABASE PARTITIONING 177
13.1. Configure partitions – Oracle 177
13.2. Partitioning anexisting 5.0 database that is not partitioned 178
13.3. Data retention with partitions 178
13.4. Partitions alerts 179

14. EARCHIVING 180
14.1. Continuous export 180
14.2. Sanitizer export 181
14.3. Audit 187
14.4. Retention policy 187
14.5. eArchiving interface 187
14.5.1. Security 188
14.5.2. Get batch by batch ID 188
14.5.3. Get the messageId exported in a batch 189
14.5.4. History of exported batches 190
14.5.5. Request to export a batch based on batch id 191
14.5.6. Notification from the archiving client that it has successfully archived or failed to archive a specific batch 192
14.5.7. Get messages which were not archived within a specific period 192
14.5.8. Get the current start date of the continuous export 193
14.5.9. Set the current start date of the continuous export 194
14.5.10. Get the current start date of the sanity export 194
14.5.11. Set the current start date of the sanity export 194
14.5.12. Receive notification when a batch has been exported in the shared folder 195
14.5.13. Receive notification when an expired batch has been deleted 196

15. NON REPUDIATION 197

16. TLS CONFIGURATION 198
16.1. TLS Configuration 198
16.1.1. Transport Layer Security in Domibus 198
16.1.2. Client Side Configuration 198
16.1.3. Server side configuration 199

17. DYNAMIC DISCOVERY OF UNKNOWN PARTICIPANTS 204
17.1. Overview 204
17.2. Domibus configuration for PEPPOL 204
17.3. PMode configuration for PEPPOL 205
17.3.1. Sender PMode 205
17.3.2. Receiver PMode 206
17.3.3. Sender and Receiver PMode 206
17.4. Policy and certificates for PEPPOL 207
17.5. Message format for PEPPOL 208
17.6. SMP entry 209
17.7. Domibus configuration for OASIS 209
17.8. PMode configuration for OASIS 209
17.8.1. Sender PMode 209
17.8.2. Receiver PMode 211
17.9. Policy and certificates for OASIS 211
17.10. Message format for OASIS 211

18. MESSAGE PULLING 213
18.1. Setup 213
18.2. Configuration restriction 215

19. MULTITENANCY 216
19.1. Configuration 217
19.1.1. Database general schema 218
19.1.2. Creating new tenant domains 219
19.1.3. Tomcat 221
19.1.4. WebLogic and WildFly 222
19.1.5. WebLogic specific configuration 222
19.2. PMode 223
19.3. Tenant domain Properties 223
19.4. Super Properties 229
19.5. Logging 229
19.6. Users 231
19.7. Plugins 231
19.7.1. Plugin Users 231
19.8. Switching from non Multitenancy to Multitenancy mode 232

20. ALERTS 233
20.1. Description 233

20.2. Main configuration 233
20.3. Message status change alerts 236
20.4. Authentication Alerts 236
20.5. User Password alerts 239
20.6. Plugin User Password alerts 240
20.7. Certificate scanner alerts 241
20.8. Configuration example 242
20.8.1. Example: domibus.properties 242
20.8.2. Example: domain_name-domibus.properties 244

21. DSS EXTENSION CONFIGURATION 245
21.1. Overview 245
21.2. Installation 245
21.2.1. Enable Unlimited Strength Jurisdiction Policy 245
21.2.2. Download and install DSS extension 245
21.2.3. Configure proxy 246
21.2.4. DSS extension truststores 246
21.2.5. Configure LOTL truststore 247
21.2.6. Configure custom trusted list 247
21.2.7. Configure PMode policy 248
21.2.8. Dss extension activation 248
21.3. DSS extension properties 248

22. SETTING LOGGING LEVELS AT RUNTIME 252
22.1. Description 252

23. EU LOGIN INTEGRATION 254
23.1. Description 254
23.2. Installation and Configuration 254
23.2.1. Installation 254
23.2.2. Configuration 255
23.3. Domibus UI changes 256

24. DOMIBUS STATISTICS 258
24.1. Metrics type 258
24.1.1. JVM metrics 258
24.1.2. Custom metrics 258
24.1.3. JMS Queues count metrics 259
24.2. Metrics access 259
24.2.1. Log file 259
24.2.2. Servlet 259
24.2.3. Jmx 269

25. PAYLOAD ENCRYPTION 271

26. MESSAGE PRIORITIZATION 272

26.1. Introduction....272
26.2. Solution overview....272
26.3. Solution detail....272
26.3.1. Using the underlying JMS infrastructure....273
26.3.2. Using dedicated JMS listeners (with a specific concurrency) for each configured message priority....274

27. SSL OFFLOADING....276
27.1. Configuration....276

28. OPERATIONAL GUIDELINES....279
28.1. JMS Queue Management....279
28.2. Log Management....279
28.2.1. Log Level....279
28.2.2. Log Rotation and Archiving....280
28.2.3. Log Monitoring....280
28.3. Capacity Planning....280
28.3.1. JVM Memory Management....280
28.3.2. CPU, IO operations and Disk Space Monitoring....280
28.4. Database Management....280
28.4.1. Database Monitoring....280
28.4.2. Database Archiving....280
28.4.3. Monitor Message Life Cycle....280
28.5. Domibus Monitoring/Domibus IsAlive AP....281
28.5.1. Database Monitoring....281
28.5.2. Check Domibus DB, JMS Broker and Quartz Trigger isAlive....281
28.5.3. DataBase Monitoring:....281
28.5.4. JMS Monitoring:....282
28.5.5. Quartz Trigger Monitoring:....282

29. ANNEX 1 - USAGE OF CERTIFICATES IN PEPPOL AND OASIS....285

30. LIST OF FIGURES....286

31. CONTACT INFORMATION....287

1. INTRODUCTION

This Administration Guide is intended for Server Administrators in charge of installing, managing and troubleshooting an eDelivery Access Point.

1.1. Purpose

The purpose of this guide is to provide detailed information on how to deploy and configure Domibus on WebLogic, Tomcat and WildFly with MySQL or Oracle. It also provides detailed descriptions of related Security Configurations (Policies, Certificates), Message Filtering, PMode Configuration, Application Monitoring, Custom Plugins Registration, JMS Monitoring, Data Archiving, Troubleshooting and TLS Configuration.

1.2. References

Ref.	Document	Content outline
[REF1]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus	Location of the release artefacts on the Digital site
[REF2]	https://dev.mysql.com/downloads/connector/j/	Location to download the MySQL JDBC driver from the Official website
[REF3]	https://download.oracle.com/otn-pub/otn_software/jdbc/211/ojdbc8.jar	Location of the Oracle JDBC driver from the Official website
[REF4]	https://docs.wildfly.org/26.1/High_Availability_Guide.html#Clustering_and_Domain_Setup_Walkthrough	Location of the Official documentation about how to setup a cluster on WildFly 26.1
[REF5]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/PKI+Service	eDelivery Public Key Infrastructure (PKI) Service Offering Document

Ref.	Document	Content outline
[REF6]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus	Location of the latest Domibus release about the Single Web Portal
[REF7]	https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/6.0/html/XML_Configuration_Reference/files/cxf-http-conf-2_7_0_xsd_Element_http-conf_tlsClientParameters.html	RedHat page for the XML Configuration Reference of the *http-conf:tlsClientParameters* element
[REF8]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/SMP	SMP (Service Metadata Publisher) and Dynamic Discovery in AS4 Gateways
[REF9]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/SMP	Space describing the SMP (Service Metadata Publisher)
[REF10]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eDelivery+AS4	eDelivery AS4 Profile
[REF11]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus	Software Architecture Document (SAD)
[REF12]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus	JMS Plugin Interface Control Document (ICD)
[REF13]	https://ec.europa.eu/digital-building-blocks/wikis/display/CEKB/External+Articles	eDelivery Knowledge Base containing troubleshooting and How To articles.
[REF14]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Guidance+on+Digital+Certificates	Guidance on digital certificates used in eDelivery.

2. CONVENTIONS

The commands and configuration files listed in this document usually contain a mix of reserved words (commands, instructions and system-related special words) and user-defined words (chosen by the user) as well as comments and preferred values for specific variables. The conventions used in this document, to distinguish between them, are the following:

- To keep this document release agnostic as much as possible, the strings "x-y-z" or "x.y.z" are intended to refer to the version of Domibus discussed in this version of the document, in the present case "Domibus 5.1.3".
- **Bold** is used for "reserved" words and commands.
- *Normal italic* together with a short description of the argument is used for user-defined names (chosen by you to designate items like users, passwords, database etc.). Normally contains at least 2 words separated by "_". Could be highlighted at times.
- ***Bold and Italic*** is used for advisable values which can be changed by the user depending on their infrastructure. Could be highlighted at times.
- Comments are sometimes added to describe the purpose of the commands, usually enclosed in brackets ().

By default, non-OS specific paths will be described using Linux patterns.

2.1. Example 1: Sample Oracle Statement

create user *edelivery_user* **identified by** *edelivery_password*;

grant all privileges to *edelivery_user*;

(Where *edelivery_user* and *edelivery_password* are names chosen by the user)

2.2. Example 2: Sample Configuration file

jdbc.datasource.0.driver.name=com.mysql.cj.jdbc.Driver

jdbc.datasource.0.driver.url=jdbc:mysql://*localhost:3306/* *domibus_schema*

jdbc.datasource.0.driver.password=*edelivery_password*

jdbc.datasource.0.driver.username=*edelivery_user*

(Where:

- *edelivery_user, domibus_schema* and *edelivery_password* are names chosen by the user.

- ***localhost:3306*** represents hostname:port parameters of the MySQL database.)

3. PREREQUISITES

Please install the following software on the target system. For further information and installation details, we kindly advise you to refer to the software owner's documentation.

- Java 8 features / compile with Oracle JDK 8. Tested to run correctly with::

  - Oracle JRE version 8u291+ for Tomcat, WildFly and WebLogic:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html:

  - OpenJDK 11.0.11 for WildFly and Tomcat (tested with AdoptOpenJDK version 11.0.9.1+1):
  https://openjdk.java.net/projects/jdk/11/

- One of the supported Database Management Systems:
  - MySQL 8.0.13 or above
  - Oracle 12c R2 and Oracle 19c (tested version, future versions might work)
- If you do not plan to deploy Domibus according to the Pre-Configured Single Server Deployment method, you must also install one of the supported application/web servers:
  - WebLogic Version 12.2.1.4 (tested version, future versions might also work)
  - WildFly 26.1.x Final (tested version, future versions might also work)
  - Apache Tomcat 9.x (tested version, future versions might also work)
- All Domibus installation resources, including full distribution and documentation can be found on the Single Web Portal:

  https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus

- Domibus application is supported in all modern web browsers, but it will not work with Internet Explorer.

SECURITY NOTE: To ensure their system's security, users installing any of the Domibus packages labelled as "Full Distribution" have the responsibility to update the application servers to the latest version after the installation.

3.1. Binaries repository

All the artefacts can be directly downloaded from the Digital site (cf.[REF1]).

4. DOMIBUS DEPLOYMENT

Remark:

*The variable **edelivery_path** referring to the folder where the package is installed will be used later in this document.*

4.1. Database Configuration

For this step, you will have to use the following resources (see section §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-sql-scripts.zip**

4.1.1. MySQL configuration

1. Unzip **domibus-msh-distribution-X.Y.Z-sql-scripts.zip** in *edelivery_path***/sql-scripts.**
2. Open a command prompt and navigate to this directory: *edelivery_path***/sql-scripts**.
3. Execute the following MySQL commands at the command prompt:

mysql -h *localhost* -u *root_user* **--password=*root_password* -e "drop schema if exists *domibus_schema*; create schema *domibus_schema*; alter database** *domibus_schema* **charset= utf8mb4 collate= utf8mb4_bin; create user *edelivery_user*@localhost identified by '*edelivery_password*'; grant all on** *domibus_schema.** **to** *edelivery_user@localhost*;"

The above script creates a schema (*domibus_schema*) and a user (*edelivery_user*) that have all the privileges on the schema**.**

mysql -h *localhost* -u *root_user* **--password=*root_password* -e "*grant xa_recover_admin on *.* to edelivery_user@localhost;*"**

mysql -h *localhost* -u *edelivery_user* **--password=*edelivery_password domibus_schema* < mysql-x.y.z.ddl**

The above script creates the required objects in *domibus_schema.*

mysql -h *localhost* -u *edelivery_user* **--password=*edelivery_password domibus_schema* < mysql-x.y.z-data.ddl**

The above script populates the tables with some predefined data in *domibus_schema*.

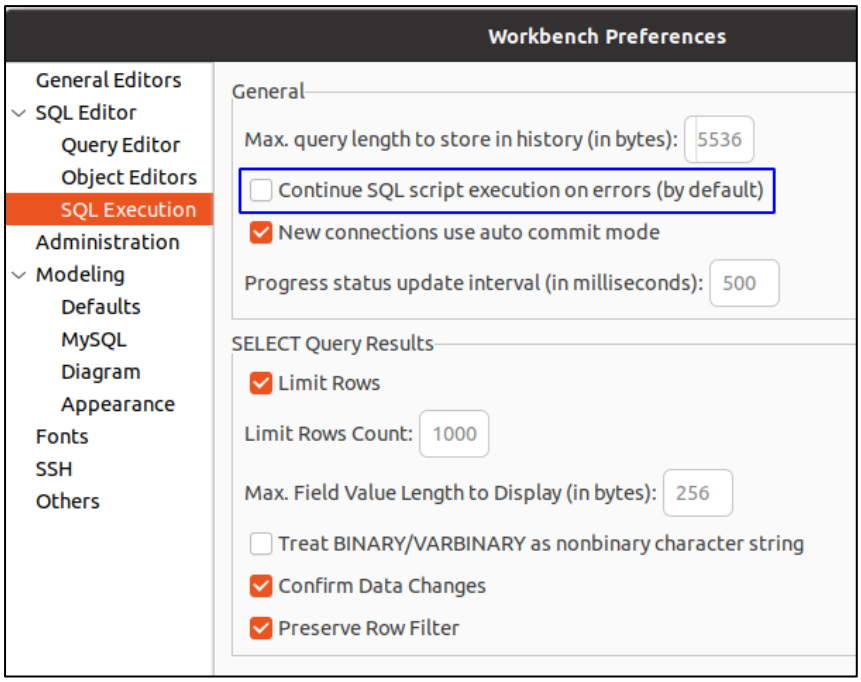
Alternative ways to run the update scripts:

The recommended way of running the update scripts is with the MySql command line:

mysql -h localhost -u root_user --password=root_password domibus_schema < script_file.ddl

If you choose to execute the update scripts in other ways, please make sure that the execution of the script stops in case of errors. Here are some known issues about running the scripts with other MySql clients:

- Connecting with the mysql command-line client (or MySql Workbench) to the database then running the command source script_file.ddl : with this method, the execution oof the script continues even in case of error;
- MySql Workbench does not always stop on errors when running script files. Make sure the following setting is disabled:



- DataGrip will stop on errors only if you open the script file, select all the lines from the file and execute them. Running the update with “Run SQL script...” from the context menu or with the “Current file run/debug configuration” will not stop on errors;
- Oracle SQL Developer will not stop on errors in the update files for MySql databases.

MySQL server configuration

Please find below some information regarding the MySQL server configuration:

1. (Optional) Storing payload messages in a database with size over 30 MB.

   Domibus can temporarily store the messages in the database.

   Therefore, it is required to increase the maximum allowed size of packets. Update the default properties of **my.ini** (Windows) or **my.cnf** (Linux).

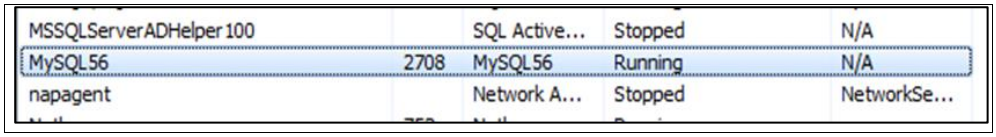
   - **max_allowed_packet** property

```
# The maximum size of one packet or any generated or intermediate string, or any
# parameter sent by the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet=512M
```

- **innodb_log_file_size** property

```
# Size of each log file in a log group. You should set the combined size
# of log files to about 25%-100% of your buffer pool size to avoid
# unneeded buffer pool flush activity on log file overwrite. However,# note that larger logfile size will increase the time needed for the recovery process
innodb_log_file_size=5120M
```

- Restart MySQL service (Windows):



MySQL service

2. (Optional) For storing payload messages in a file system instead of a database, see §5.2-"Domibus Properties".

3. For MySQL 8 and Connector/J 8.0.x please set the database timezone to UTC. One way of setting the timezone is to modify the MySQL **my.ini** configuration file by adding the following property with the adjusted timezone to UTC. The connector is configured now to use the UTC server timezone by default. For future date time values – e.g. next attempts for the retry mechanisms) – we also save the timezone offset when persisting to be able to recreate the correct instant when reading back later on, in the event the timezone offset will have changed while waiting for the future event to occur.

```
default-time-zone='+00:00'
```

Remark:

If you are using Windows, make sure to have the parent directory of mysql.exe added to your PATH variable.

4.1.2. Oracle configuration

1. Unzip **domibus-msh-distribution-X.Y.Z-sql-scripts.zip** in *edelivery_path***/sql-scripts**

2. Open a command prompt and navigate to this directory: *edelivery_path***/sql-scripts**.

3. Open a command line session, log in and execute the following commands:

sqlplus sys as sysdba (password should be the one assigned during the Oracle installation)
==========
Once logged in Oracle:
CREATE USER <***edelivery_user***> IDENTIFIED BY <***edelivery_password***>
DEFAULT TABLESPACE <tablespace>
QUOTA UNLIMITED ON <tablespace>;
GRANT CREATE SESSION TO <edelivery_user>;
GRANT CREATE TABLE TO <edelivery_user>;
GRANT CREATE VIEW TO <edelivery_user>;
GRANT CREATE SEQUENCE TO <edelivery_user>;
GRANT CREATE PROCEDURE TO <edelivery_user>;
GRANT CREATE JOB TO <edelivery_user>;
GRANT EXECUTE ON DBMS_XA TO <edelivery_user>;
GRANT EXECUTE ON DBMS_LOCK TO <edelivery_user>;
GRANT SELECT ON PENDING_TRANS$ TO <edelivery_user>;
GRANT SELECT ON DBA_2PC_PENDING TO <edelivery_user>;
GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO <edelivery_user>;

CONNECT <***edelivery_user***>
SHOW USER; (should return: edelivery_user)
@oracle-x.y.z.ddl
@oracle-x.y.z-data.ddl

EXIT
==========

Remarks:

1. Replace <edelivery_user> and <edelivery_password> with corresponding values.

2. <tablespace> is created and assigned by your DBA; for local/test installations just replace it with user's tablespace. The quota could be limited to a specific size.

3. DDL/SQL scripts must be run with the @ sign from the location of the scripts

Please find below some information regarding the Oracle configuration:

The Oracle JDBC driver is configured now to use the UTC server timezone by default when persisting and reading date time values. For future date time values – e.g. next attempts for the retry mechanisms – we also save the timezone offset when persisting to be able to recreate the correct instant when reading back later on, in the event the timezone offset will have changed while waiting for the future event to occur.

4.1.3. MySQL and Oracle Deletion scripts

A deletion script for MySQL and Oracle Domibus DB is available in the **domibus-msh-distribution-X.Y.Z-sql-scripts.zip**.

The purpose of the script is to delete all messages within a user-defined period to recover disk space. The script requires a START_DATE parameter and an END_DATE parameter to be set.

The tables affected by the execution of this script are:

- TB_MESSAGING
- TB_ERROR_LOG
- TB_PARTY_ID
- TB_RECEIPT_DATA
- TB_PROPERTY
- TB_PART_INFO
- TB_RAWENVELOPE_LOG
- TB_ERROR
- TB_USER_MESSAGE
- TB_SIGNAL_MESSAGE
- TB_RECEIPT
- TB_MESSAGE_INFO
- TB_MESSAGE_LOG
- TB_MESSAGE_UI

Any information relevant to a message received or sent during the predefined period, will be removed from these tables.

In order to execute this script, it is advised to use a UI tool such as SQL developer or MySQL workbench.

Important: to keep the JMS queues synchronized with the DB data that will be deleted by this script, the Domibus Administrator should remove manually the associated JMS messages from the plugin notifications queues

4.2. Domibus on WebLogic

This section does not include the installation of WebLogic server. It is assumed that the WebLogic Server is installed and a Domain is created.
Hereafter the domain location will be referred as *DOMAIN_HOME (user-defined name)*.

Remarks:

- *The Apache CXF library referred by Domibus, internally uses the environment variable java.io.tmpdir to buffer large attachments received. If the property java.io.tmpdir is not specified, then by default this points to the <Weblogic_domain_directory>. It is recommended to point this to a local directory '_tmp' on each managed server and accessible by the Weblogic application server. The disk space allocated for '_tmp' directory would depend on the size of attachments received. On production environment it is recommended to provide 100GB for '_tmp'.*

- *CXF has a limitation of being able to validate signatures of only 28 payload attachments at a time. As a result, Domibus cannot send/receive more than 28 attachments in a single AS4 message.*

- *User can modify default JNDI name property in the **domibus.properties** by setting:*
  - *#Weblogic JDBC-DataSource JNDI Name*
  - *domibus.jdbc.datasource.jndi.name=jdbc/cipaeDeliveryDs*
  - *#Weblogic JDBC-DataSource Quartz JNDI Name*

4.2.1. Single Server Deployment

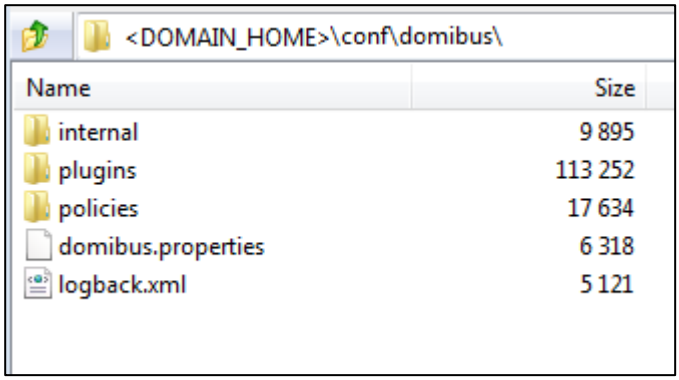
For this step, you will have to use the following resources (see section §3.1–"*Binaries repository* " for the download location):

- **domibus-msh-distribution-X.Y.Z-weblogic-war.zip**
- **domibus-msh-distribution-X.Y.Z-weblogic-configuration.zip**

At least one of the following plugins should be downloaded and installed:

- **domibus-msh-distribution-X.Y.Z-default-ws-plugin.zip**
- **domibus-msh-distribution-X.Y.Z-default-jms-plugin.zip**
- **domibus-msh-distribution-X.Y.Z-default-fs-plugin.zip**

1. Download and unzip **domibus-msh-distribution- X.Y.Z-weblogic-configuration.zip** in the directory *DOMAIN_HOME***/conf/domibus**



2. Download and unzip **domibus-msh-distribution- X.Y.Z-weblogic-war.zip** in a temporary folder to prepare it for deployment.

3. Configure your Keystore based on section §5.1.3 – "Certificates"

4. Add the following lines in:

   - For Windows: *DOMAIN_HOME***\bin\setDomainEnv.cmd**

     - Locate the **set DOMAIN_HOME** statement and add the following lines after:

```
…
set DOMAIN_HOME
# Added for Domibus
**********************************************************************
set EXTRA_JAVA_PROPERTIES=%EXTRA_JAVA_PROPERTIES% -
Ddomibus.config.location=%DOMAIN_HOME%/conf/domibus -Djava.io.tmpdir=<path to _tmp
directory>
#
set JAVA_OPTIONS=%JAVA_OPTIONS%
-Dweblogic.transaction.allowOverrideSetRollbackReason=true
#
**********************************************************************
*******
…
```

- For Linux: *DOMAIN_HOME***/bin/setDomainEnv.sh**

  - Locate the **export DOMAIN_HOME** statement and add the following lines after:

```
…
export DOMAIN_HOME
# Added for Domibus
**********************************************************************
EXTRA_JAVA_PROPERTIES="$EXTRA_JAVA_PROPERTIES -
Ddomibus.config.location=$DOMAIN_HOME/conf/domibus -Djava.io.tmpdir=<path to _tmp
directory>"
export EXTRA_JAVA_PROPERTIES
#
**********************************************************************************
*******
JAVA_OPTIONS="${JAVA_OPTIONS}
-Dweblogic.transaction.allowOverrideSetRollbackReason=true"
export JAVA_OPTIONS
…
```

5. Run the WebLogic Scripting Tool (WLST) to create the JMS resources and the Database datasources from the command line:

   - Download the WSLT Package from following location: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/public/eu/europa/ec/digit/ipcis/wslt-api/1.9.1/wslt-api-1.9.1.zip

   - Configure the WSLT API tool

     - Unzip the **wslt-api-1.9.1.zip**

     - Define the **WL_HOME** as a system environment variable to point to the WebLogic 'wlserver' directory as defined in the **DOMAIN_HOME/bin/setDomainEnv.[cmd|sh]**

       e.g. WL_HOME=/wls12130/wlserver

   - Take the script **WeblogicSingleServer.properties** from **domibus-msh-distribution-X.Y.Z-weblogic-configuration.zip** under the scripts directory and copy the **WeblogicSingleServer.properties** file into the **wslt-api-1.9.1** directory and adapt the following properties:

     - Adapt the properties for connecting to the WebLogic domain:

```
domain.loading.type=connect
domain.connect.url=t3://localhost:7001
domain.connect.username=weblogic_name
domain.connect.password=weblogic_password
domain.name=my_domain1
```

     - Adapt the jdbc.datasource properties for the datasources:

- Common to Oracle and MySQL:

```
#################################################################
## Domain configuration
#################################################################
## Variables
##---------Cross module--------
#Domibus application module target
application.module.target = AdminServer

##---------JMS configuration--------
#Domibus JMS application server name
jms.server.name = eDeliveryJMS
#Domibus JMS application module name
jms.module.name = eDeliveryModule
#Domibus JMS file store name
jms.server.store = eDeliveryFileStore
#Domibus JMS application module group
jms.queue.subdeployment.name = eDeliverySubD

##---------Database configuration--------
#Domibus database url
jdbc.datasource.driver.url=jdbc:oracle:thin:@127.0.0.1:1521:<SID/Service>
#Domibus database user name
jdbc.datasource.driver.username=your_username
#Domibus database user password
jdbc.datasource.driver.password=your_password
```

- For Oracle database:

```
jdbc.datasource.0.name=eDeliveryDs
jdbc.datasource.0.targets=${application.module.target}
jdbc.datasource.0.jndi.name=jdbc/cipaeDeliveryDs
jdbc.datasource.0.pool.capacity.max=50
jdbc.datasource.0.pool.connection.test.onreserv.enable=true
jdbc.datasource.0.pool.connection.test.onreserv.sql=SQL SELECT 1 FROM DUAL
jdbc.datasource.0.driver.name=oracle.jdbc.driver.OracleDriver
jdbc.datasource.0.driver.url=${jdbc.datasource.driver.url}
jdbc.datasource.0.driver.password=${jdbc.datasource.driver.password}
jdbc.datasource.0.driver.username=${jdbc.datasource.driver.username}
jdbc.datasource.0.driver.properties.items=0

jdbc.datasource.1.name=edeliveryNonXA
jdbc.datasource.1.targets=${application.module.target}
jdbc.datasource.1.jndi.name=jdbc/cipaeDeliveryNonXADs
jdbc.datasource.1.transaction.protocol=None
jdbc.datasource.1.pool.capacity.max=50
jdbc.datasource.1.pool.connection.test.onreserv.enable=true
jdbc.datasource.1.pool.connection.test.onreserv.sql=SQL SELECT 1 FROM DUAL
jdbc.datasource.1.driver.name=oracle.jdbc.OracleDriver
jdbc.datasource.1.driver.url=${jdbc.datasource.driver.url}
jdbc.datasource.1.driver.password=${jdbc.datasource.driver.password}
jdbc.datasource.1.driver.username=${jdbc.datasource.driver.username}
jdbc.datasource.1.driver.properties.items=0
```

Remark:

MySQL configuration is commented by default. To enable MySQL, remove the comment (#) from the lines below. Do not forget to add the comment (#) for Oracle to disable it.

- For MySQL:

```
#jdbc.datasource.0.name=eDeliveryDs
#jdbc.datasource.0.targets=${application.module.target}
#jdbc.datasource.0.jndi.name=jdbc/cipaeDeliveryDs
#jdbc.datasource.0.transaction.protocol=LoggingLastResource
#jdbc.datasource.0.pool.connection.test.onreserv.enable=true
#jdbc.datasource.0.pool.connection.test.onreserv.sql=SQL SELECT 1
#jdbc.datasource.0.driver.name=com.mysql.cj.jdbc.Driver
#jdbc.datasource.0.driver.url=${jdbc.datasource.driver.url}
#jdbc.datasource.0.driver.password=${jdbc.datasource.driver.password}
#jdbc.datasource.0.driver.username=${jdbc.datasource.driver.username}
#jdbc.datasource.0.driver.properties.items=0

#jdbc.datasource.1.name=edeliveryNonXA
#jdbc.datasource.1.targets=${application.module.target}
#jdbc.datasource.1.jndi.name=jdbc/cipaeDeliveryNonXADs
#jdbc.datasource.1.transaction.protocol=None
#jdbc.datasource.1.pool.capacity.max=50
#jdbc.datasource.1.pool.connection.test.onreserv.enable=true
#jdbc.datasource.1.pool.connection.test.onreserv.sql=SQL SELECT 1
#jdbc.datasource.1.driver.name=com.mysql.cj.jdbc.Driver
#jdbc.datasource.1.driver.url=${jdbc.datasource.driver.url}
#jdbc.datasource.1.driver.password=${jdbc.datasource.driver.password}
#jdbc.datasource.1.driver.username=${jdbc.datasource.driver.username}
#jdbc.datasource.1.driver.properties.items=0
```

- Adapt the property for location of the filestore **persistent.filestore.0.location**.

  Example:

```
persistent.filestore.0.location=DOMAIN_HOME/filestore
```

Remark:

Make sure that the path for the filestore contains forward slashes (/).

- Adapt if necessary the JMX security configuration:

  Example:

```
################################################################
## Policy configuration
################################################################
security.policies.0.mode = CREATE
security.policies.0.resource = type=<jmx>, operation=invoke, application=,
mbeanType=weblogic.management.runtime.JMSDestinationRuntimeMBean
security.policies.0.realm = myrealm
security.policies.0.authorizer = XACMLAuthorizer
security.policies.0.expression= Rol(Admin)|Grp(Administrators)|Grp(JMSManagers)
security.policies.items = 1
################################################################
## Users configuration
################################################################
security.users.0.realm=myrealm
security.users.0.name=jmsManager
security.users.0.password=jms_Manager1
security.users.0.comment=
security.users.0.authenticator=DefaultAuthenticator
security.users.items=1
################################################################
## Groups configuration
################################################################
security.groups.0.realm=myrealm
security.groups.0.name=JMSManagers
security.groups.0.description=
security.groups.0.authenticator=DefaultAuthenticator
security.groups.items=1
################################################################
## Groups Membership configuration
################################################################
security.group.member.0.user=jmsManager
security.group.member.0.groups=JMSManagers
security.group.member.0.realm=myrealm
security.group.member.0.authenticator=DefaultAuthenticator
security.group.member.items=1
```

- Start the WebLogic domain from within *DOMAIN_HOME:*
  - For Windows:

    startWebLogic.cmd
  - For Linux:

    startWebLogic.sh
- Execute the following command from within the **wlstapi-1.9.1/bin** directory:
  - For Windows:

    wlstapi.cmd ../scripts/import.py --property ../WeblogicSingleServer.properties
  - For Linux:

    wlstapi.sh ../scripts/import.py --property ../WeblogicSingleServer.properties

REMARK:
In order to send messages containing bodyload payloads, you must ensure the Weblogic server is started with the following extra parameter:

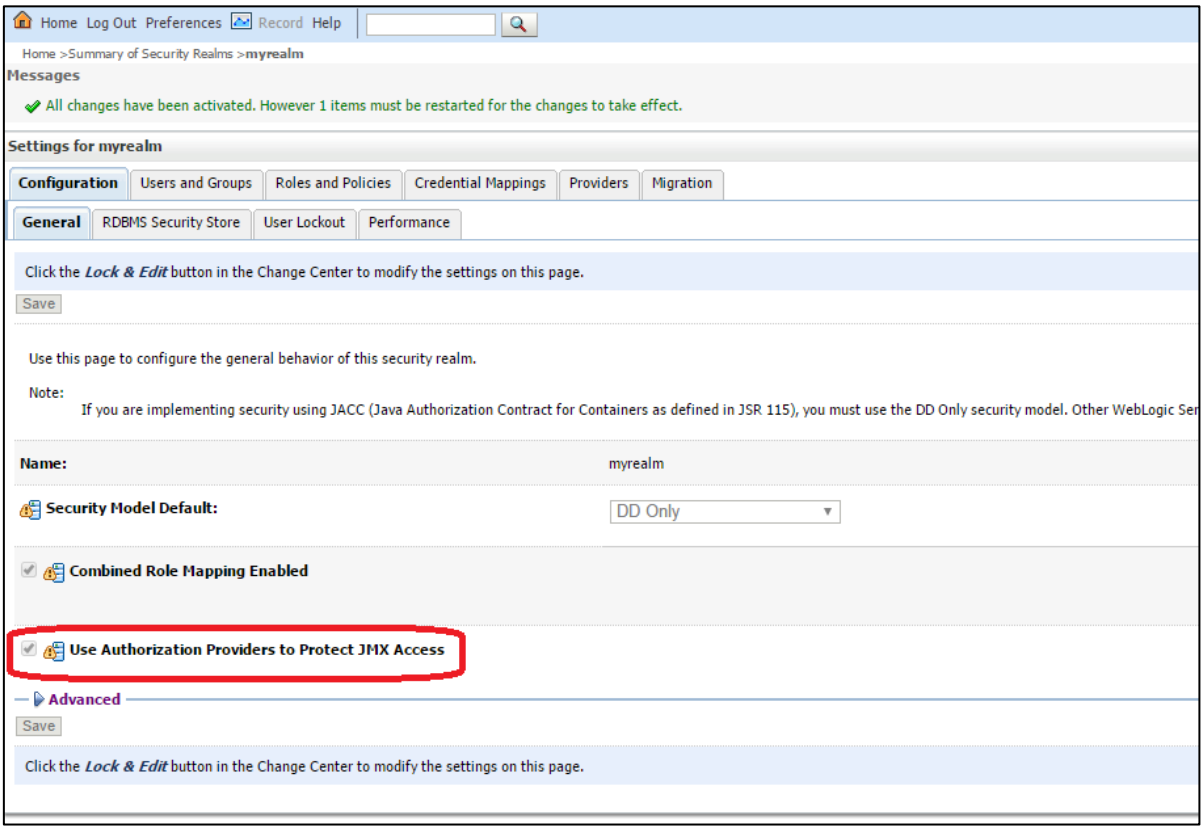
-
Dorg.apache.cxf.binding.soap.messageFactoryClassName=com.sun.xml.internal.messaging.saaj.soap.ver1_2.SOAPMessageFactory1_2Impl

Expected result:

```
Saving all your changes ...
Saved all your changes successfully.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Location changed to serverRuntime tree. This is a read-only tree with DomainMBean as the root.
For more help, use help('domainConfig')

Disconnected from weblogic server: AdminServer
```

6. Activate the use of the authorization providers to protect the JMX access:



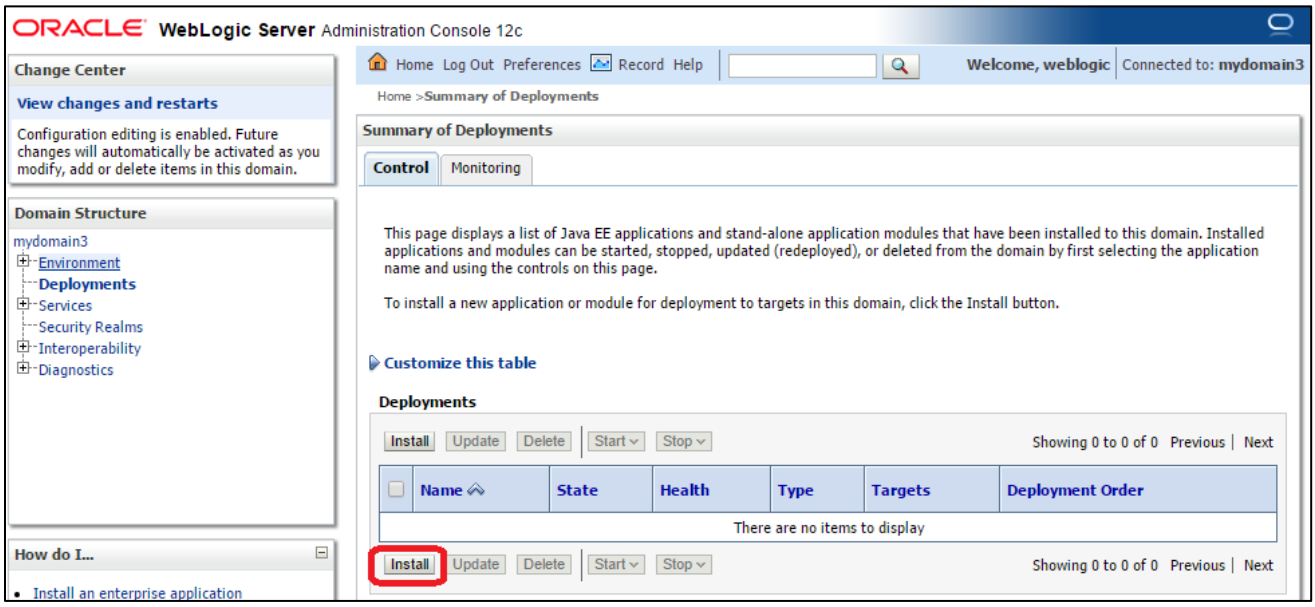
7. The database dialect is pre-configured to use the Oracle database. If you are using a MySQL database, you should adapt the following properties in <*DOMAIN_HOME*>**/conf/domibus/domibus.properties** as highlighted in the example below:

```
# ------------------------------ EntityManagerFactory ----------------------
domibus.entityManagerFactory.jpaProperty.hibernate.connection.driver_class=
com.mysql.cj.jdbc.Driver
domibus.entityManagerFactory.jpaProperty.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

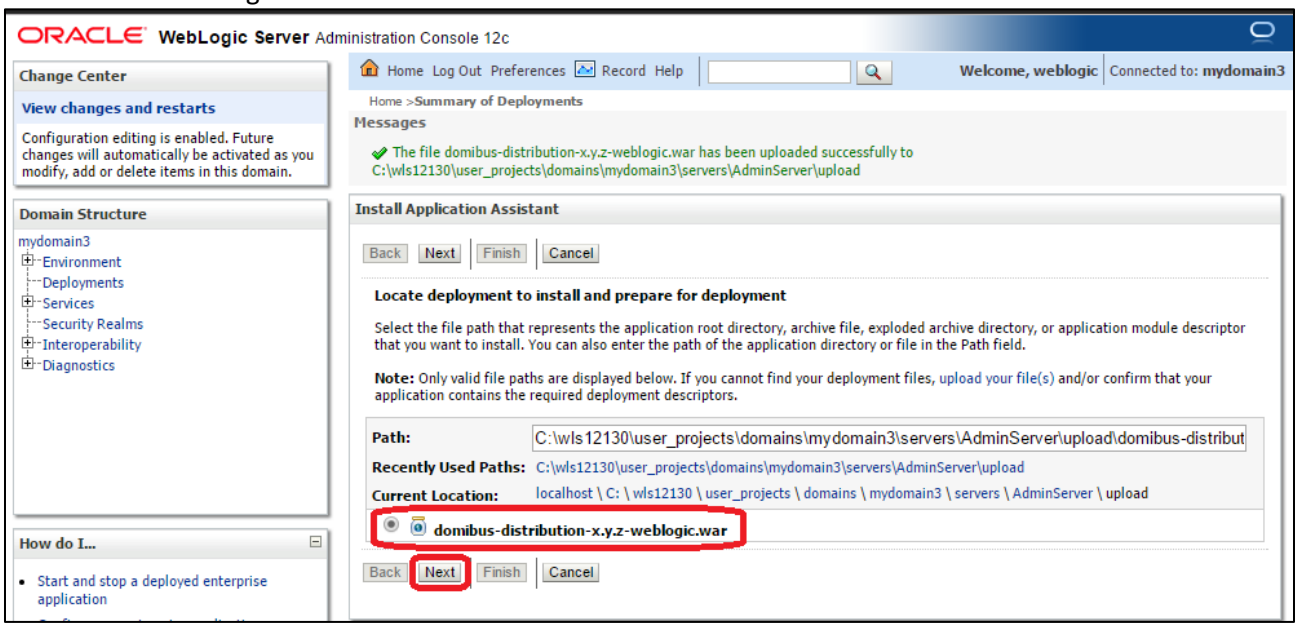
8. Install the WS Plugin. For more details, see section §6.2.1.2 – *"WebLogic"*.

9. Deploy **domibus-msh-distribution-X.Y.Z-weblogic.war**

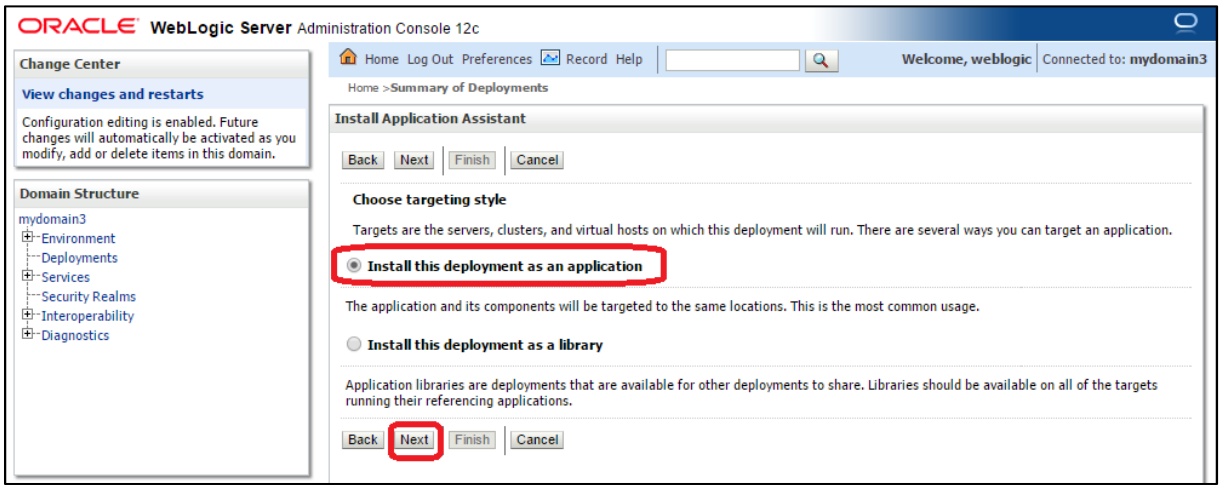
- Click on **Install**:



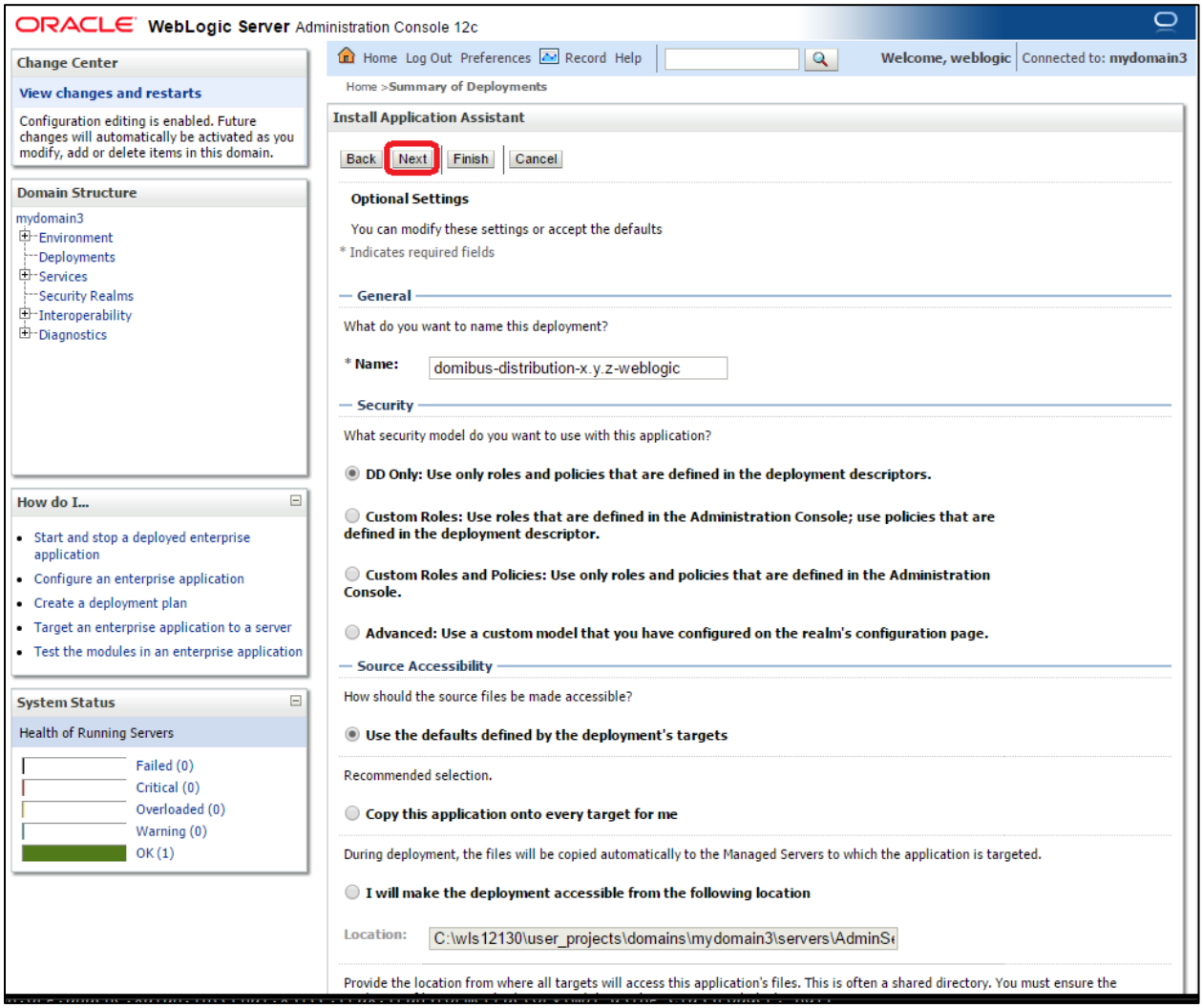
- Navigate to the location of the **.war** file and click **Next**:



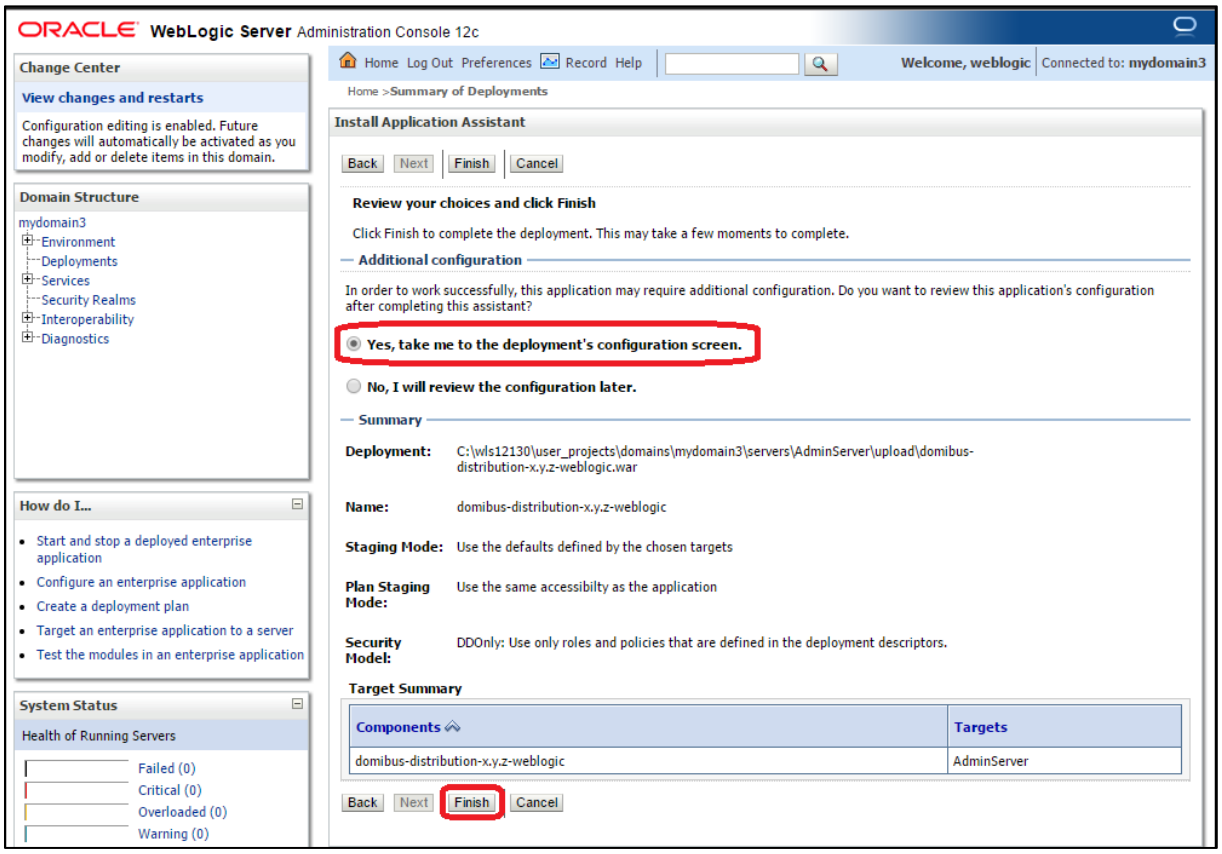
- Choose **Install this deployment as an application** and click **Next**:



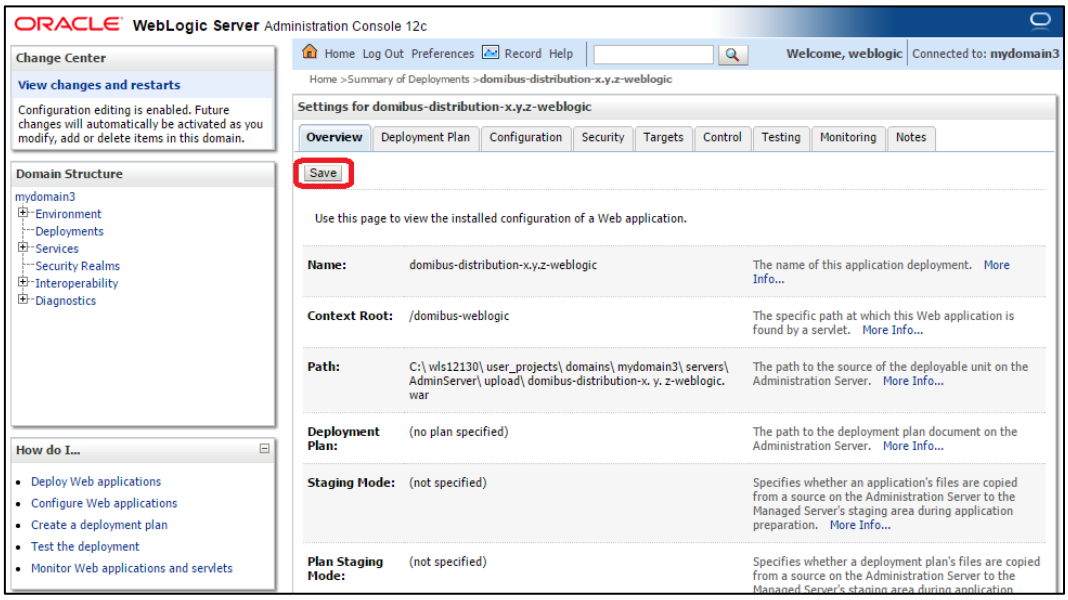
- Accept the default options and click **Next**:



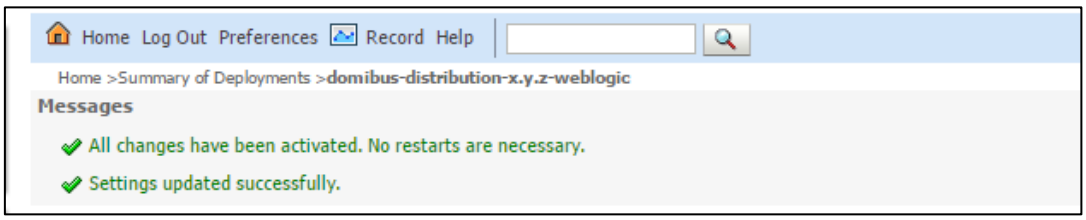
- Select the following option and click **Finish**:



- Here is an overview of the resulting settings, you can now click on the **Save** button:



The expected positive response to the deployment request should be the following:



10. Verify the installation by navigating with your browser to http://*localhost*:*7001*/domibus: if you can access the page it means the deployment was successful.

    (By default: User = ***admin***; for the password, look in the logs for the phrase: "Default password for user admin is"*).*

    ***Remark**:*

    It is recommended to change the passwords for the default users (See §10.1 – Administration for further information).

Expected result:

4.2.2. Clustered Deployment

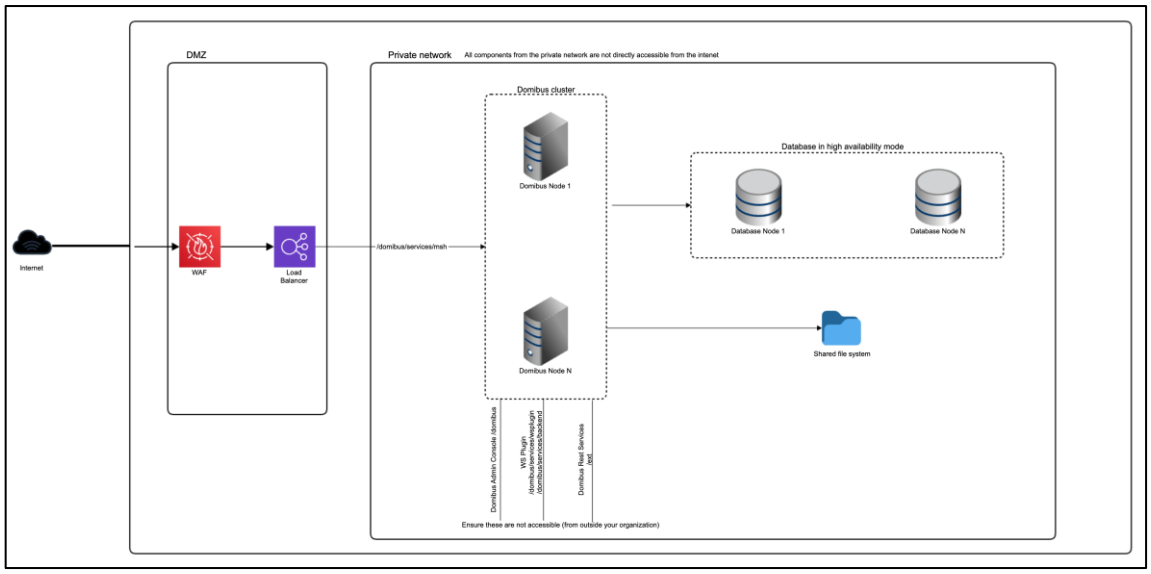


Figure 1 - Diagram representing the Deployment of Domibus in a Cluster on WebLogic

Remark:

In this section, we assume that a Domain and a WebLogic Cluster are already setup.

For this step, you will have to use the following resources (see section §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-weblogic-war.zip**
- **domibus-msh-distribution-X.Y.Z-weblogic-configuration.zip**

At least one of the following plugins should be downloaded and installed:

- **domibus-msh-distribution-X.Y.Z-default-ws-plugin.zip**
- **domibus-msh-distribution-X.Y.Z-default-jms-plugin.zip**

- **domibus-msh-distribution-X.Y.Z-default-fs-plugin.zip**

1. Download and unzip **domibus-msh-distribution- X.Y.Z-weblogic-configuration.zip** in a shared location that is accessible by all the nodes from the cluster. We will refer to this directory as *shared_edelivery_path*.

2. Download and unzip **domibus-msh-distribution- X.Y.Z-weblogic-war.zip** in a temporary folder to prepare it for deployment.

3. Configure your Keystore based on section §5.1.3 – *"Certificates"*.

4. Add the following lines in:

    - For Windows: *DOMAIN_HOME***\bin\setDomainEnv.cmd**

        - Locate the **set DOMAIN_HOME** statement and add the following lines after:

```
…
set DOMAIN_HOME
# Added for Domibus
*****************************************************************
set EXTRA_JAVA_PROPERTIES=%EXTRA_JAVA_PROPERTIES% -
Ddomibus.config.location=%SHARED_LOCATION%/conf/Domibus -Djava.io.tmpdir=<path to
_tmp directory>
#
*****************************************************************************
*******
set JAVA_OPTIONS=%JAVA_OPTIONS%
-Dweblogic.transaction.allowOverrideSetRollbackReason=true
…
```

Remark: SHARED_LOCATION is the shared location where Domibus configuration is found for a clustered installation.

    - For Linux: *DOMAIN_HOME***/bin/setDomainEnv.sh**

        - Locate the **export DOMAIN_HOME** statement and add the following lines after:

```
…
export DOMAIN_HOME
# Added for Domibus
*****************************************************************
EXTRA_JAVA_PROPERTIES="$EXTRA_JAVA_PROPERTIES -
Ddomibus.config.location=$SHARED_LOCATION/conf/domibus -Djava.io.tmpdir=<path to _tmp
directory>"
export EXTRA_JAVA_PROPERTIES
#
*****************************************************************************
*******
JAVA_OPTIONS="${JAVA_OPTIONS}
-Dweblogic.transaction.allowOverrideSetRollbackReason=true”
export JAVA_OPTIONS
…
```

Remark: SHARED_LOCATION is the shared location where Domibus configuration is found for a clustered installation.

5. Run the WebLogic Scripting Tool (WLST) to create the necessary JMS resources and Database datasources from the command line:

   - Download the WLST Package from the following location: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/eDelivery/eu/europa/ec/digit/ipcis/wslt-api/1.9.1/wslt-api-1.9.1.zip

   - Configure the WSLT API tool:

     - Unzip the **wslt-api-1.9.1.zip**

     - Define the WL_HOME (SET or export command depending on your operating system) environment variable to point to the WebLogic **wlserver** directory

       e.g. WL_HOME=/wls12130/wlserver

   - Take the script **WeblogicCluster.properties** from **domibus-msh-distribution-X.Y.Z-weblogic-configuration.zip** under the scripts directory and copy the **WeblogicCluster.properties** file into the **wslt-api-1.9.1** directory and apply the following changes:

     - Adapt the properties for connecting to the WebLogic domain:

       Common to Oracle and MySQL:

```
#################################################################
## Domain configuration
#################################################################
## Variables
##---------Cross module--------
#Domibus application module target
application.module.target= cluster_name


##---------JMS configuration--------
#Domibus JMS application server name
jms.server.name = eDeliveryJMS
#Domibus JMS application module name
jms.module.name=eDeliveryModule
#Domibus JMS file store name
jms.server.store=eDeliveryFileStore
#Domibus JMS application module group
jms.queue.subdeployment.name=eDeliverySubD
```

```
##---------Database configuration--------
#Domibus database url

jdbc.datasource.driver.url= jdbc:oracle:thin:@127.0.0.1:1521:<SID/Service>
#Domibus database user name
jdbc.datasource.driver.username= your_username
#Domibus database user password
jdbc.datasource.driver.password= your_password
```

For Oracle database:

```
###################################################################
## JDBC datasource Server [eDeliveryDs]
###################################################################
# Oracle configuration
jdbc.datasource.0.name=eDeliveryDs
jdbc.datasource.0.targets=${application.module.target}
jdbc.datasource.0.jndi.name=jdbc/cipaeDeliveryDs
jdbc.datasource.0.pool.capacity.max=50
jdbc.datasource.0.pool.connection.test.onreserv.enable=true
jdbc.datasource.0.pool.connection.test.onreserv.sql=SQL SELECT 1 FROM DUAL
jdbc.datasource.0.driver.name= oracle.jdbc.driver.OracleDriver
jdbc.datasource.0.driver.url=${jdbc.datasource.driver.url}
jdbc.datasource.0.driver.password=${jdbc.datasource.driver.password}
jdbc.datasource.0.driver.username=${jdbc.datasource.driver.username}
jdbc.datasource.0.driver.properties.items=0

###################################################################
## JDBC datasource Server [edeliveryNonXA]
###################################################################
# Oracle configuration
jdbc.datasource.1.name=edeliveryNonXA
jdbc.datasource.1.targets=${application.module.target}
jdbc.datasource.1.jndi.name=jdbc/cipaeDeliveryNonXADs
jdbc.datasource.1.transaction.protocol=None
jdbc.datasource.1.pool.capacity.max=50
```

```
jdbc.datasource.1.pool.connection.test.onreserv.enable=true
jdbc.datasource.1.pool.connection.test.onreserv.sql=SQL SELECT 1 FROM DUAL
jdbc.datasource.1.driver.name=oracle.jdbc.OracleDriver
jdbc.datasource.1.driver.url=${jdbc.datasource.driver.url}
jdbc.datasource.1.driver.password=${jdbc.datasource.driver.password}
jdbc.datasource.1.driver.username=${jdbc.datasource.driver.username}
jdbc.datasource.1.driver.properties.items=0
```

Remark:

MySQL configuration is commented out by default. To enable MySQL, remove the comment (#) from the lines below. Do not forget to add the comment (#) for Oracle to disable it.

For MySQL:

```
################################################################
## JDBC datasource Server [eDeliveryDs]
################################################################



# MySQL configuration
jdbc.datasource.0.name=eDeliveryDs
jdbc.datasource.0.targets=${application.module.target}
jdbc.datasource.0.jndi.name=jdbc/cipaeDeliveryDs
jdbc.datasource.0.pool.capacity.max=50
jdbc.datasource.0.transaction.protocol=LoggingLastResource
jdbc.datasource.0.pool.connection.test.onreserv.enable=true
jdbc.datasource.0.pool.connection.test.onreserv.sql=SQL SELECT 1
jdbc.datasource.0.driver.name=com.mysql.cj.jdbc.Driver
jdbc.datasource.0.driver.url=${jdbc.datasource.driver.url}
jdbc.datasource.0.driver.password=${jdbc.datasource.driver.password}
jdbc.datasource.0.driver.username=${jdbc.datasource.driver.username}
jdbc.datasource.0.driver.properties.items=0
# MySQL configuration
jdbc.datasource.1.name=edeliveryNonXA
jdbc.datasource.1.targets=${application.module.target}
jdbc.datasource.1.jndi.name=jdbc/cipaeDeliveryNonXADs
jdbc.datasource.1.transaction.protocol=None
jdbc.datasource.1.pool.capacity.max=50
jdbc.datasource.1.pool.connection.test.onreserv.enable=true
```

jdbc.datasource.1.pool.connection.test.onreserv.sql=SQL SELECT 1

jdbc.datasource.1.driver.name=com.mysql.cj.jdbc.Driver

jdbc.datasource.1.driver.url=${jdbc.datasource.driver.url}

jdbc.datasource.1.driver.password=${jdbc.datasource.driver.password}

jdbc.datasource.1.driver.username=${jdbc.datasource.driver.username}

jdbc.datasource.1.driver.properties.items=0

Adapt the property for location of the filestore

persistent.filestore.0.location.

Example:

persistent.filestore.0.location=DOMAIN_HOME/filestore

Remark:

Make sure that the path for the filestore contains forward slashes (/).

Adapt if necessary the JMX security configuration.

jms.module.0.targets=*cluster_name*

Set the domibus.deployment.clustered option to true:

domibus.deployment.clustered=*true*

- Start the WebLogic domain from within *DOMAIN_HOME:*
  - For Windows:

startWebLogic.cmd

  - For Linux:

startWebLogic.sh

- Execute the following command from within the **wlstapi-1.9.1/bin** directory:

  For Windows:

wlstapi.cmd ../scripts/import.py --property ../WeblogicCluster.properties

  For Linux:

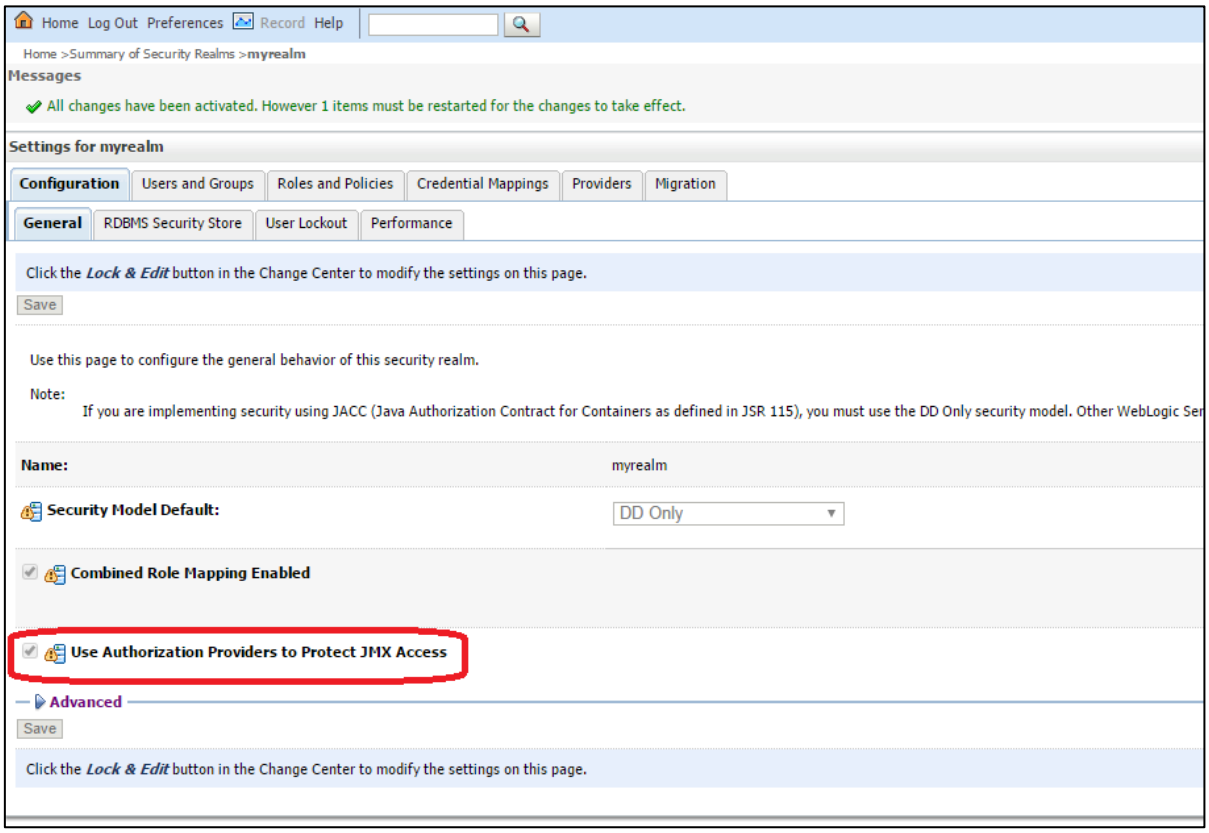
wlstapi.sh ../scripts/import.py --property ../WeblogicCluster.properties

Expected result:

```
Saving all your changes ...
Saved all your changes successfully.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Location changed to serverRuntime tree. This is a read-only tree with DomainMBean as the root.
For more help, use help('domainConfig')

Disconnected from weblogic server: AdminServer
```

6. Activate the use of the authorization providers to protect the JMX access:



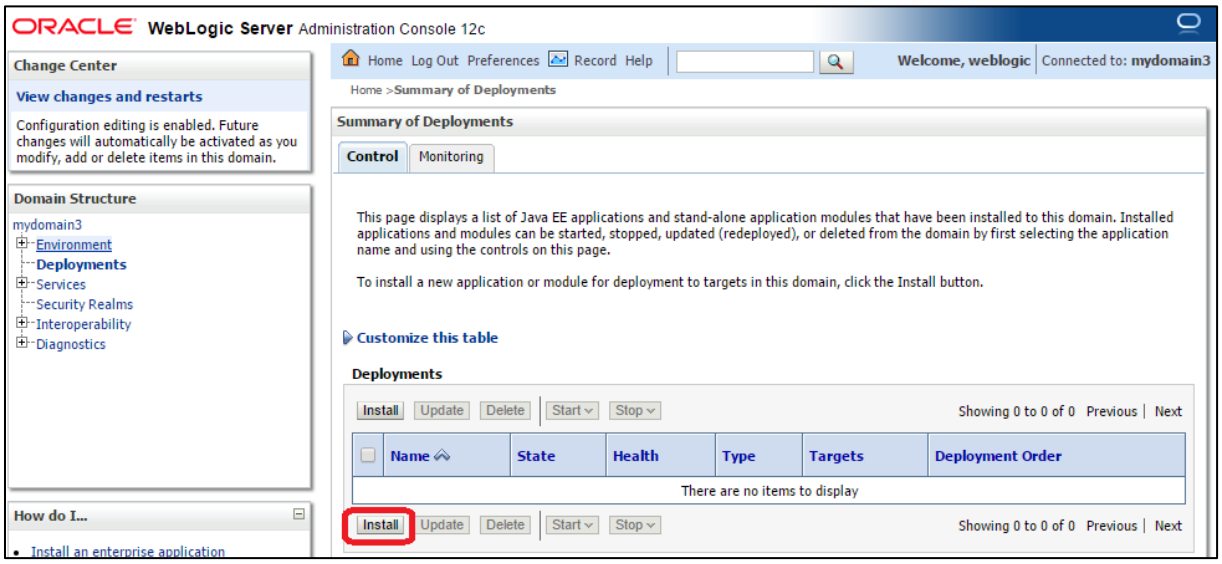
7. The database dialect is pre-configured to use the Oracle database. If you are using the MySQL database you should adapt the dialect as highlighted in the text below in <*DOMAIN_HOME*>**/conf/domibus/domibus.properties** file:

```
#EntityManagerFactory
domibus.entityManagerFactory.jpaProperty.hibernate.connection.driver_class=com.mysql.cj.jdbc.MysqlXADataSource
domibus.entityManagerFactory.jpaProperty.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

8. Install the WS plugin. For more details, refer to chapter §6.2.1.2 – *"WebLogic"*.

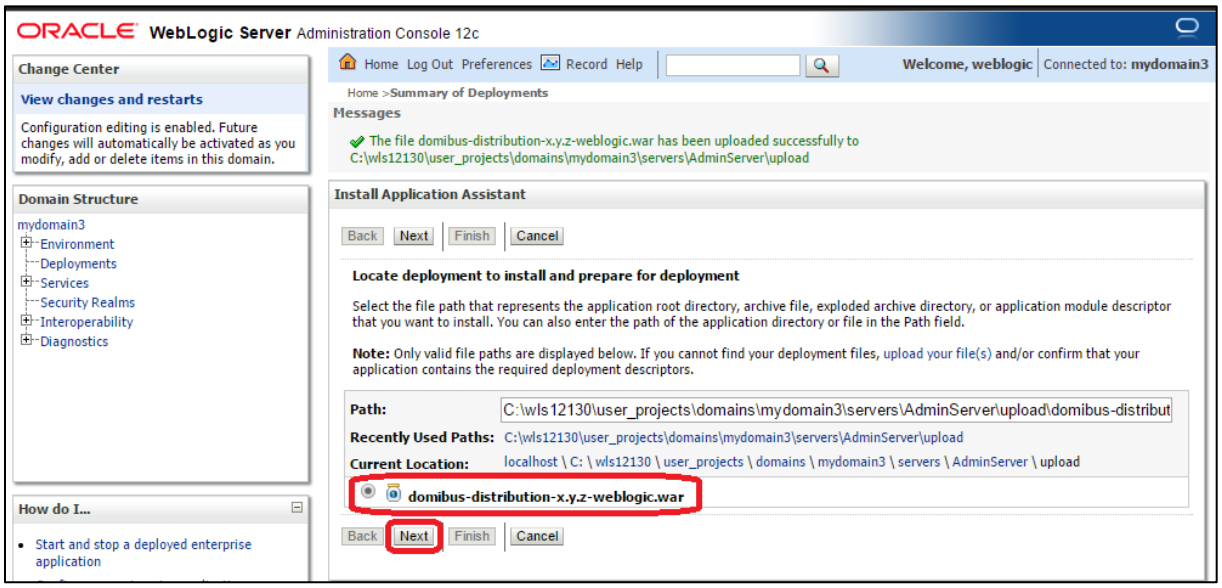
9. Deploy **domibus-msh-distribution-X.Y.Z-weblogic.war**.

- Click **Install**

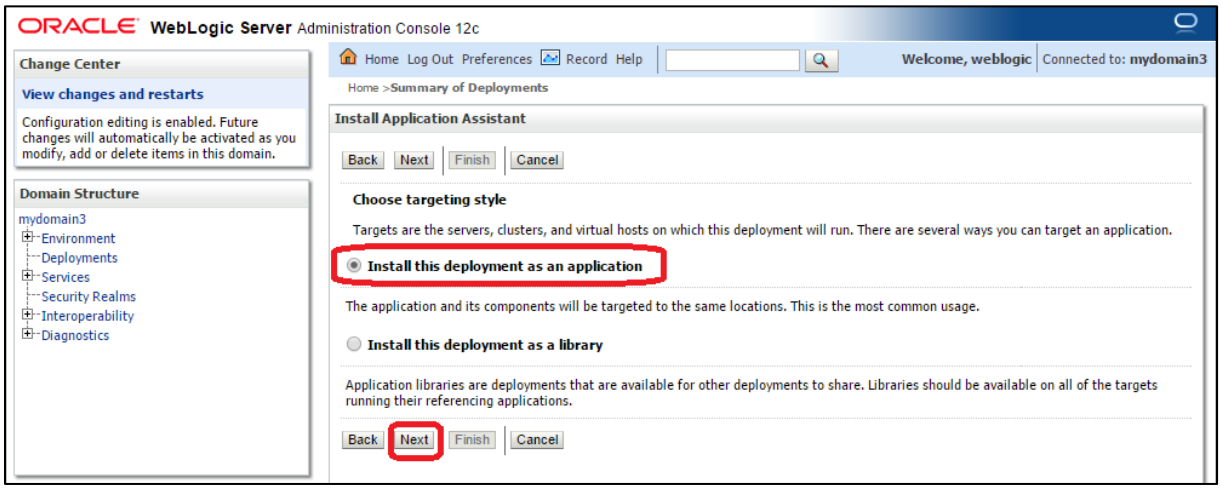


- Navigate to location DOMAIN_HOME**/conf/domibus** where the **domibus-msh-distribution-X.Y.Z-weblogic.war** file has been previously copied.

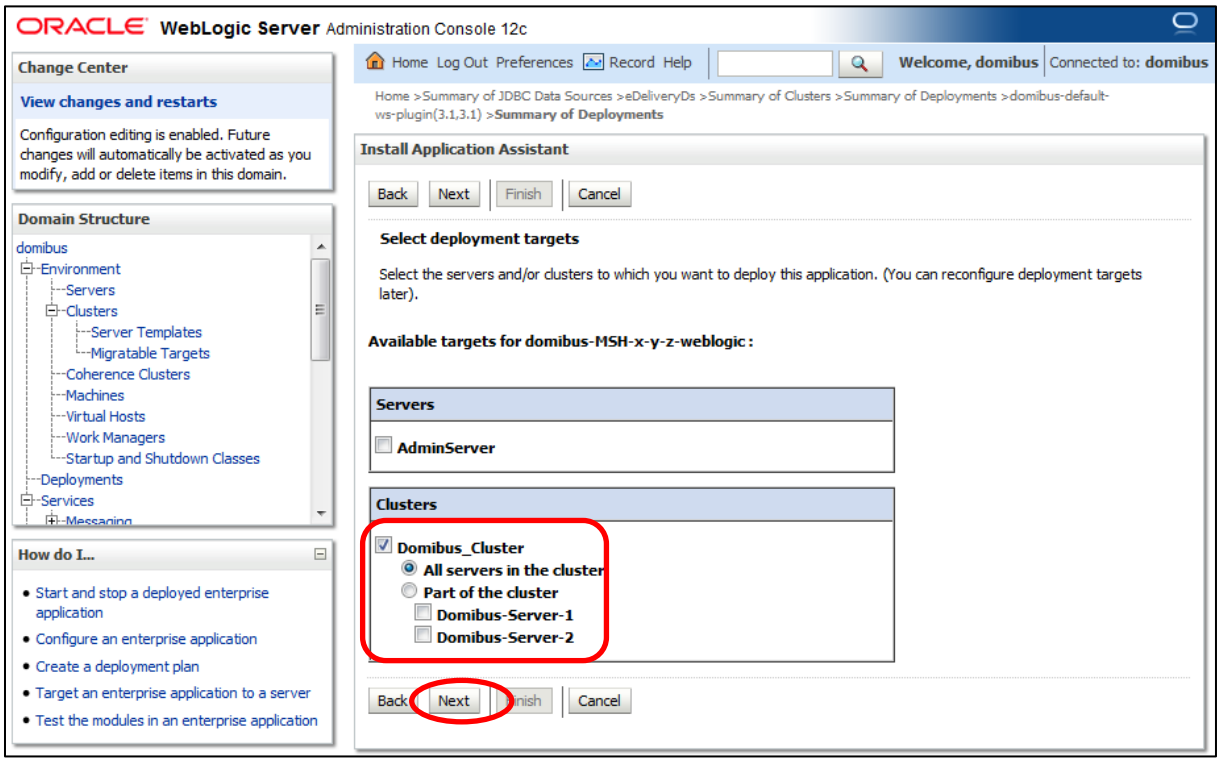
- Select the **domibus-msh-distribution-X.Y.Z-weblogic.war** file and click **Next**:



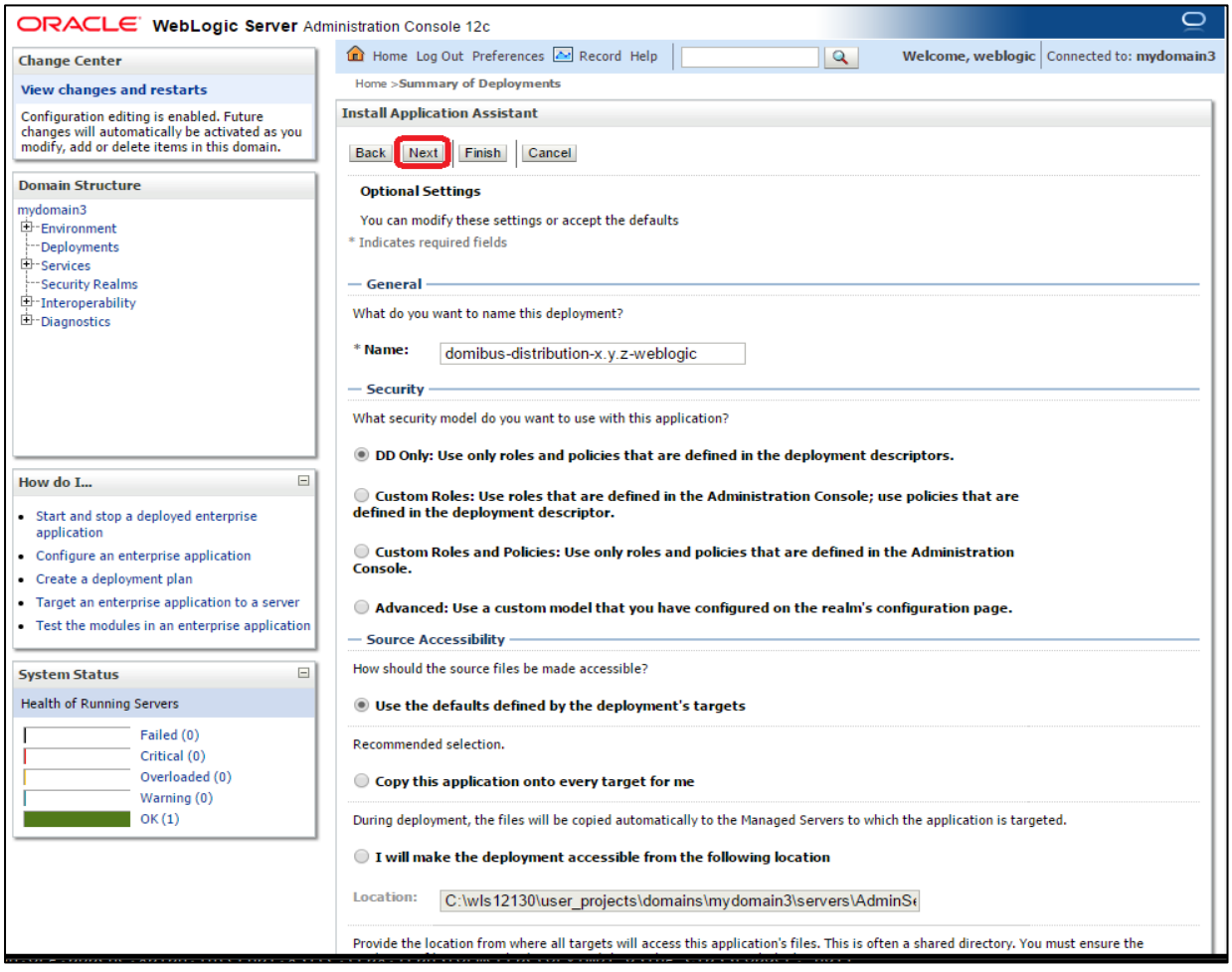
- Choose **Install this deployment as an application** and click **Next**:



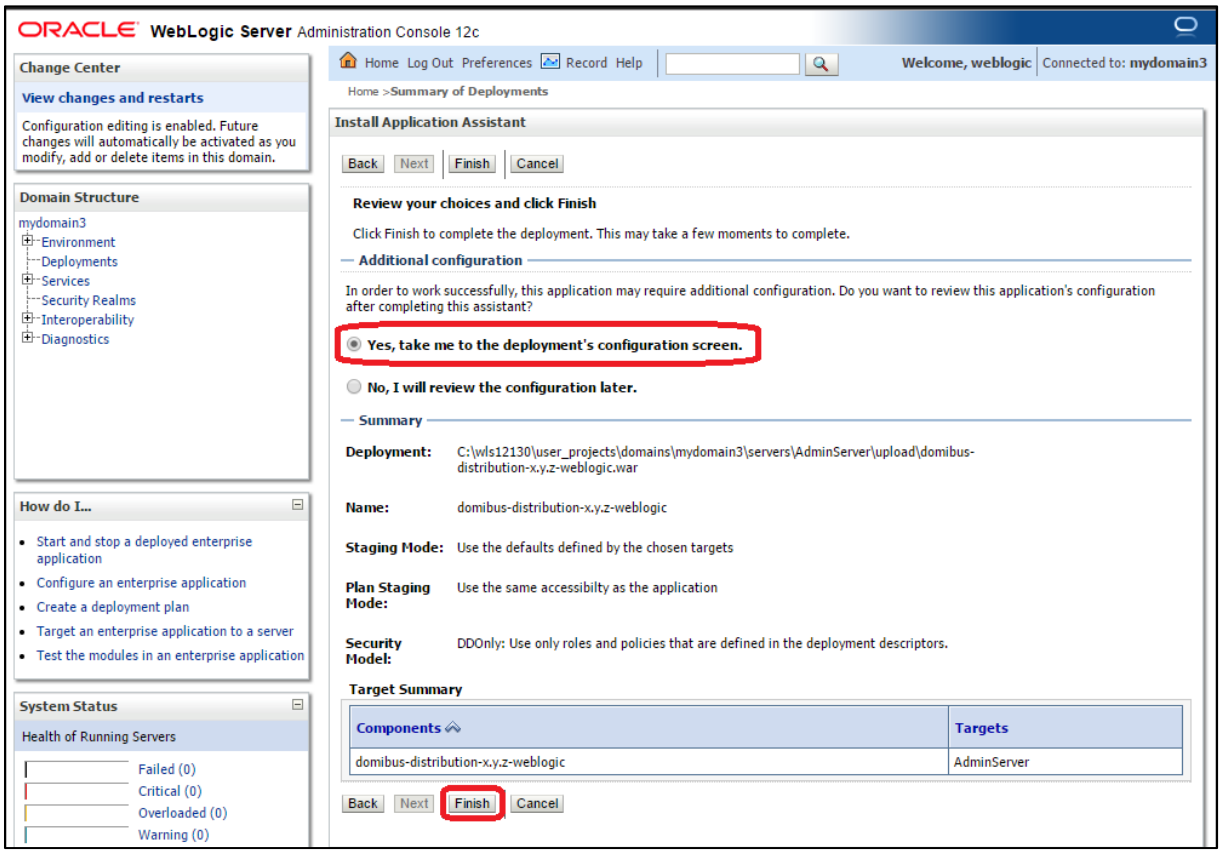
- Select your cluster for the deployment target and click **Next**:



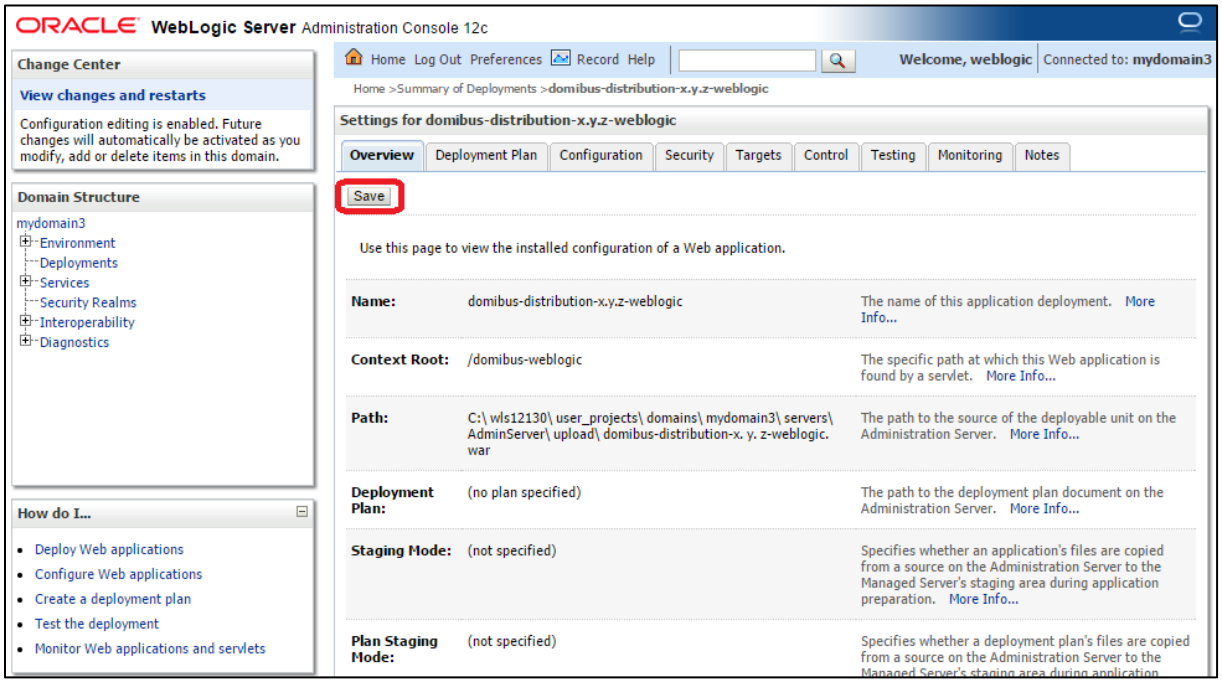
- Select the following options and click **Next**:



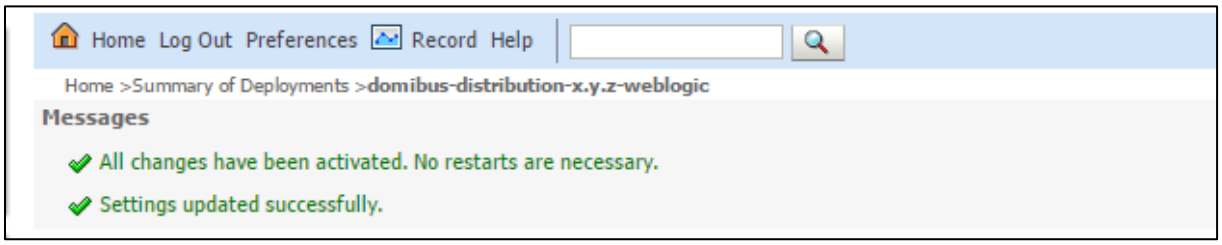
- Select the following option and click **Finish**:



- Here is an overview of the resulting settings, you can now click on the **Save** button:



The expected positive response to the deployment request should be the following:



10. Verify the installation by navigating with your browser to http://*localhost*:*7001*/domibus

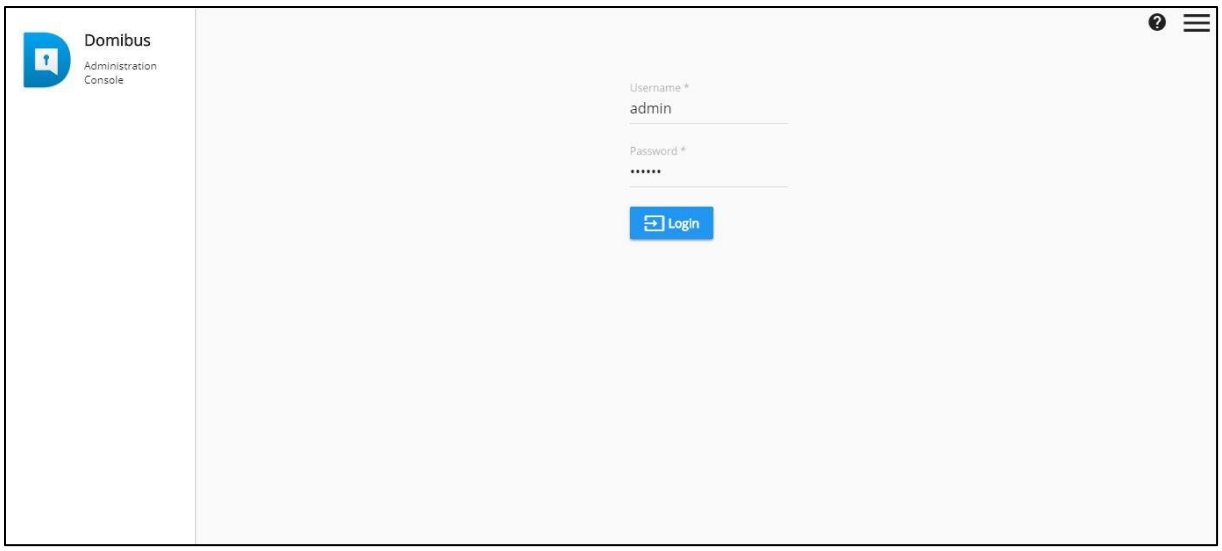
    If you can access the page, it means the deployment was successful.

    (By default: user = ***admin***; for the password, look in the logs for the phrase: "Default password for user admin is"*).*

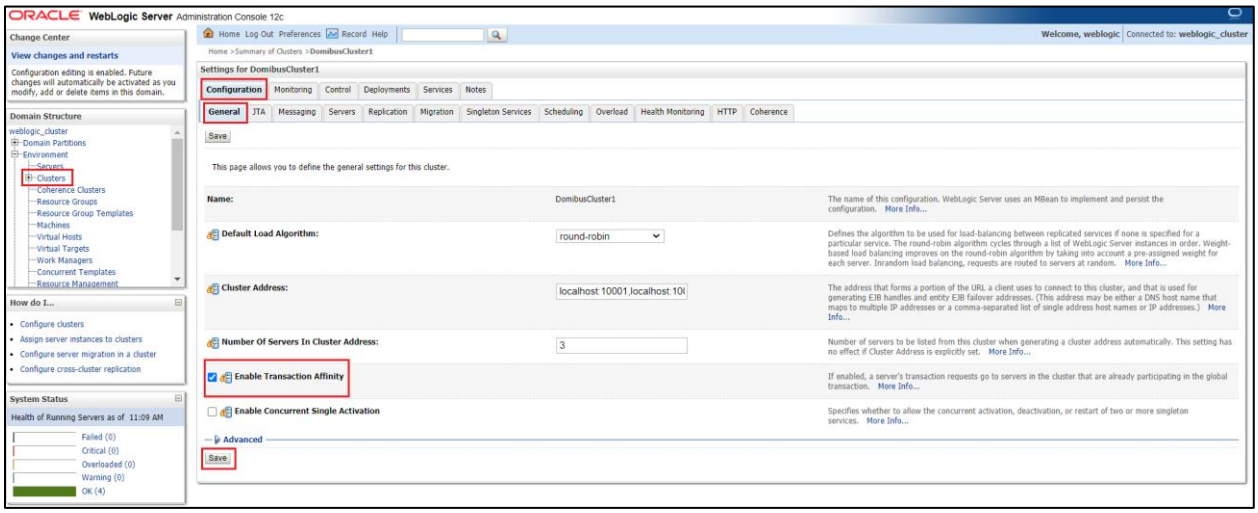
    ***Remark**:*

    It is recommended to change the passwords for the default users (See §10.1 – *"Administration " for further information).*

Expected result:



11. For performance improvement in a Weblogic cluster, enable transaction cluster affinity (see: https://docs.oracle.com/middleware/12213/wls/WLJTA/trxcon.htm#WLJTA398) and click on **Save**.



REMARK:
In order to send messages containing bodyload payloads you must ensure the Weblogic server is started with the following extra parameter:
-
Dorg.apache.cxf.binding.soap.messageFactoryClassName=com.sun.xml.internal.messaging.saaj.soap.ver1_2.SOAPMessageFactory1_2Impl

4.3. Domibus on Tomcat

Remarks:

- *As Tomcat is not a full Java EE application server and does not offer JMS capabilities by default, Domibus uses ActiveMQ as an in-memory JMS broker when deployed on a Tomcat servlet container. The configuration for the ActiveMQ JMS broker can be found in edelivery_path**/conf/domibus/internal/activemq.xml**.*

- *The Apache CXF library referred by Domibus, internally uses the environment variable java.io.tmpdir to buffer large attachments received. If the property java.io.tmpdir is not specified, then by default this points to the <CATALINA_BASE directory/temp>. It is recommended to point this to a local directory '_tmp' on each managed server and accessible by the Tomcat server. The disk space allocated for '_tmp' directory would depend on the size of attachments received. On production environment it is recommended to provide 100GB for '_tmp'.*

- *CXF has a limitation of being able to validate signatures of only 28 payload attachments at a time. As a result, Domibus cannot send/receive more than 28 attachments in a single AS4 message.*

4.3.1. Pre-Configured Single Server Deployment

For this step, you will have to use the following resources (see section §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-tomcat-full.zip**

1. Unzip the archive:

    - Unzip **domibus-msh-distribution-X.Y.Z-tomcat-full.zip** to a location on your physical machine: *edelivery_path*.

2. Prepare the database:

    - For MySQL database:

Add MySQL JDBC driver (available on MySQL official web site cf. [REF2]) in the folder *edelivery_path***/lib**.

Remark:

The version of the JDBC driver has to be mysql-connector-j-8.0.32.jar or higher.

Edit the properties file *edelivery_path***/domibus/conf/domibus/domibus.properties** and adjust the highlighted parts in the text below according to your environment. The properties associated to the database configuration are pre-configured for the MySQL database:

```
# ---------------------------------- Database ----------------------------------
#Database server name
domibus.database.serverName=localhost

#Database port
domibus.database.port=3306

#Datasource
domibus.datasource.driverClassName=com.mysql.cj.jdbc.Driver
domibus.datasource.url=jdbc:mysql://${domibus.database.serverName}:${domibus.database.port}/domibus_schema?
useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
domibus.datasource.user=edelivery_user
domibus.datasource.password=edelivery_password
domibus.datasource.maxLifetime=1800
domibus.datasource.connectionTimeout=30
domibus.datasource.idleTimeout=600
domibus.datasource.maxPoolSize=10
domibus.datasource.minimumIdle=10
domibus.datasource.poolName=
```

    - For Oracle database:

Add the Oracle JDBC driver (e.g. ***ojdbc8-21.1.0.0.jar***) (available on the Oracle official web site cf.[REF3]) in the *edelivery_path***/lib folder**.

Edit the properties file *edelivery_path***/conf/domibus/domibus.properties** and adjust the highlighted parts in the text below according to your environment:

```
# ---------------------------------- Database ----------------------------------
#Database server name
domibus.database.serverName=localhost

#Database port
domibus.database.port=1521

#General schema. Mandatory only if Domibus is configured in multi-tenancy mode.
#domibus.database.general.schema=general_schema

#set domibus.database.schema=oracle_username
domibus.database.schema= oracle_username

#Datasource
#Oracle
domibus.datasource.driverClassName=oracle.jdbc.OracleDriver
domibus.datasource.url=jdbc:oracle:thin:@${domibus.database.serverName}:${domibus.database.port}/domibus

domibus.datasource.user= oracle_username
domibus.datasource.password=edelivery_password
domibus.datasource.maxLifetime=1800
domibus.datasource.connectionTimeout=30
domibus.datasource.idleTimeout=600
domibus.datasource.maxPoolSize=10
domibus.datasource.minimumIdle=10
domibus.datasource.poolName=
```

Remark:

Configure the database dialect as it is pre-configured for MySQL by default.

```
#EntityManagerFactory
domibus.entityManagerFactory.jpaProperty.hibernate.connection.driver_class=oracle.jdbc.driver.OracleDriver
domibus.entityManagerFactory.jpaProperty.hibernate.dialect=org.hibernate.dialect.OracleDialect
```

3. Configure your Keystore based on section §5.1.3 – *"Certificates"*.

4. Set JVM parameters:

Domibus expects a single environment variable **domibus.config.location**, pointing towards the *edelivery_path***/conf/domibus** folder.

You can do this by editing the first command lines of *edelivery_path***\bin\setenv.bat** (Windows) or *edelivery_path***/bin/setenv.sh** (Linux). Set **CATALINA_HOME** equal to the absolute path of the installation ***edelivery_path*.**

- **For Windows**: edit *edelivery_path***\bin\setenv.bat** by adding the following:

```
...
set CATALINA_HOME=edelivery_path
set CATALINA_TMPDIR=<path to _tmp directory>
set  JAVA_OPTS=%JAVA_OPTS% -Dfile.encoding=UTF-8  -Xms128m -Xmx1024m -XX:PermSize=64m
set JAVA_OPTS=%JAVA_OPTS% -Ddomibus.config.location=%CATALINA_HOME%\conf\domibus
...
```

- **For Linux**: edit *edelivery_path***/domibus/bin/setenv.sh** by adding the following:

```
...
export CATALINA_HOME=edelivery_path
export CATALINA_TMPDIR=<path to _tmp directory>
```

```
export JAVA_OPTS="$JAVA_OPTS -Xms128m -Xmx1024m"
export JAVA_OPTS="$JAVA_OPTS -Ddomibus.config.location=$CATALINA_HOME/conf/domibus"
…
```

5. Launch the Domibus application:

   - For Windows:

```
cd edelivery_path\bin\
startup.bat
```

   - For Linux:

```
cd edelivery_path/bin
chmod  u+x *.sh
./startup.sh
```

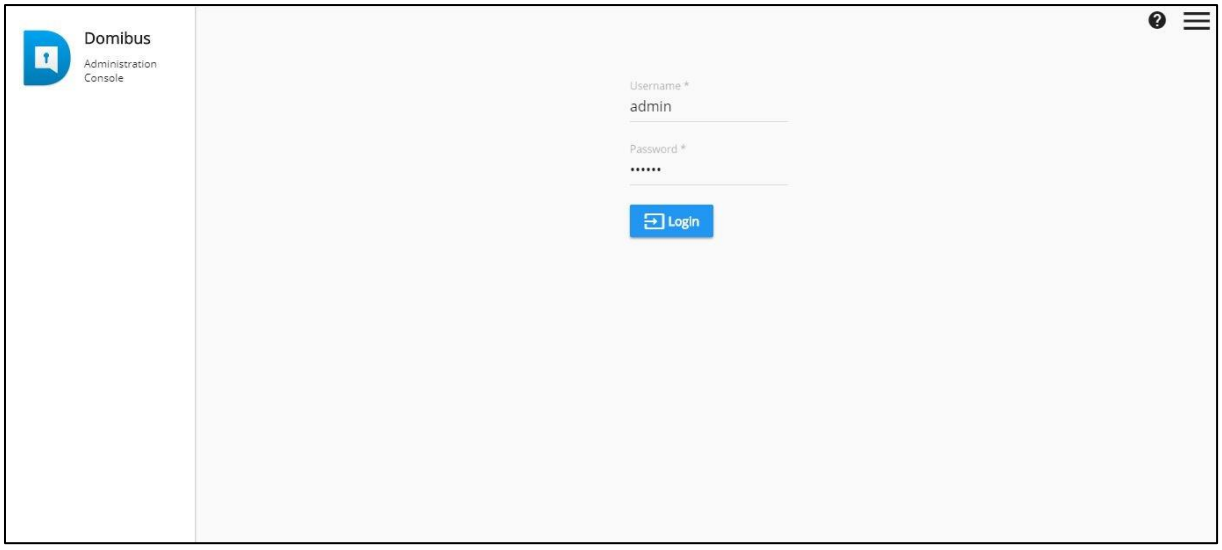
6. Display the Domibus home page on your browser: http://localhost:8080/domibus.
   (By default: User = ***admin***; for the password, look in the logs for the phrase: "Default password for user admin is"*).*

   ***Remark**:*

   It is recommended to change the passwords for the default users. See §10.1 – "Administration " for further information.

   If you can access the page, it means the deployment was successful.

Expected result:



4.3.2. Single Server Deployment

For this step, you will have to use the following resources (see §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-tomcat-configuration.zip**
- **domibus-msh-distribution-X.Y.Z-tomcat-war.zip**
- **domibus-msh-distribution-X.Y.Z-sql-scripts.zip**
- **domibus-msh-distribution-X.Y.Z-default-ws-plugin.zip**
- **domibus-msh-distribution-X.Y.Z-default-jms-plugin.zip (optional)**

- **domibus-msh-distribution-X.Y.Z-default-fs-plugin.zip (optional)**
- **domibus-msh-distribution-X.Y.Z-sample-configuration-and-testing.zip**
- **Mysql-connector-java-X.Y.Z driver (e.g.: mysql-connector-java-8.0.23.jar) [REF2])**

We assume that an Apache Tomcat 9.x is already installed and the installation location is now considered as your *edelivery_path***.**

1. Download and unzip the artefact **domibus-msh-distribution-X.Y.Z-tomcat-configuration.zip** into the directory *edelivery_path/conf/domibus.*
2. Configure the MySQL or Oracle datasource as indicated in §4.3.1 – *"Pre-Configured Single Server Deployment"*
3. Configure your Keystore based on §5.1.3 – *"Certificates"*.
4. Execute *step 4* from §4.3.1 – *"Pre-Configured Single Server Deployment"*.
5. If not already present, create a folder and name it **temp** under *edelivery_path/conf/Domibus*.
6. Rename **domibus-MSH-X.Y.Z-tomcat.war** to **domibus.war** and deploy it to *edelivery_path***/webapps**.

7. Copy Plugins subfolders to the conf/Domibus/plugins folder
8. Add the conf/dombuis path (to catalina.sh or setenv.sh). Add the following highlighted lines:
   - *JAVA_OPTS="$JAVA_OPTS -Djava.protocol.handler.pkgs=org.apache.catalina.webresources"*
   - export JAVA_OPTS="$JAVA_OPTS -Xms128m -Xmx1024m "
   - export JAVA_OPTS="$JAVA_OPTS -domibus.config.location=$CATALINA_HOME/conf/domibus"
   - *#Check for the deprecated LOGGING_CONFIG*
9. . Copy the Mysql connector (e.g.: mysql-connector-java-8.0.23.jar) to the lib folder.
10. From domibus-msh-distribution-**X.Y.Z**-sample-configuration-and-testing.zip, copy the keystores folder to …/conf/Domibus
11. Rename the domibus-MSH-tomcat-**X.Y.Z**.war to Domibus.war and copy it to webapps
12. Launch the Domibus application:
    - For Windows:

```
cd edelivery_path\bin\
startup.bat
```

    - For Linux:

```
cd edelivery_path/bin/
chmod +x *.sh
./startup.sh
```

13. Display the Domibus home page on your browser: http://localhost:8080/domibus

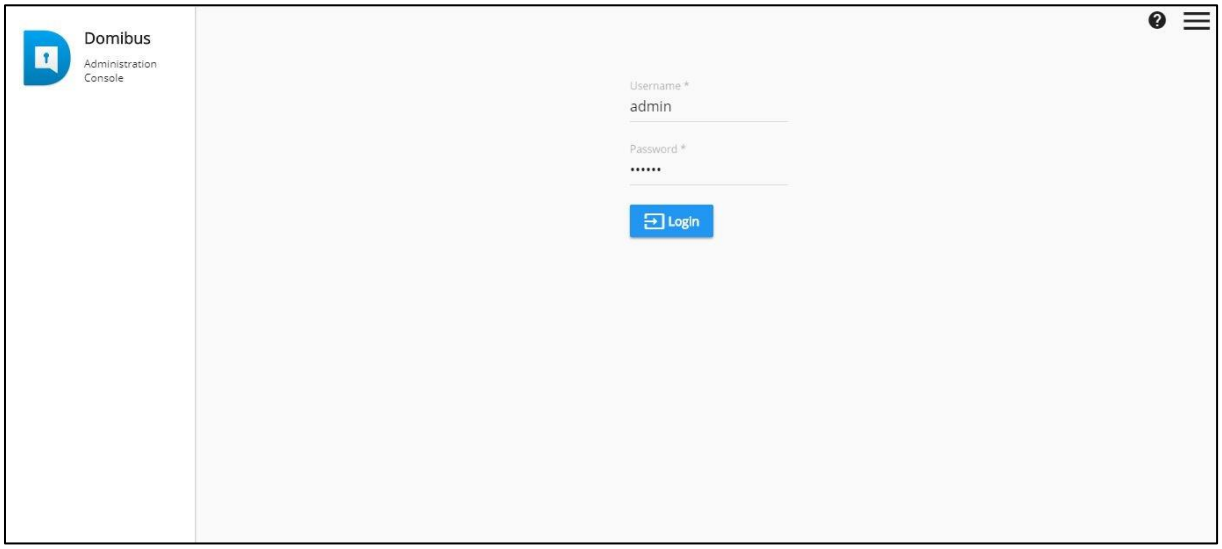
(By default: User = ***admin***; for the password, look in the logs for the phrase: “Default password for user admin is”*)*

***Remark**:*

It is recommended to change the passwords for the default users. See §10.1 – "Administration " for further information.

Accessing the page is an indication of a successful deployment.

Expected result:



4.3.3. Clustered Deployment

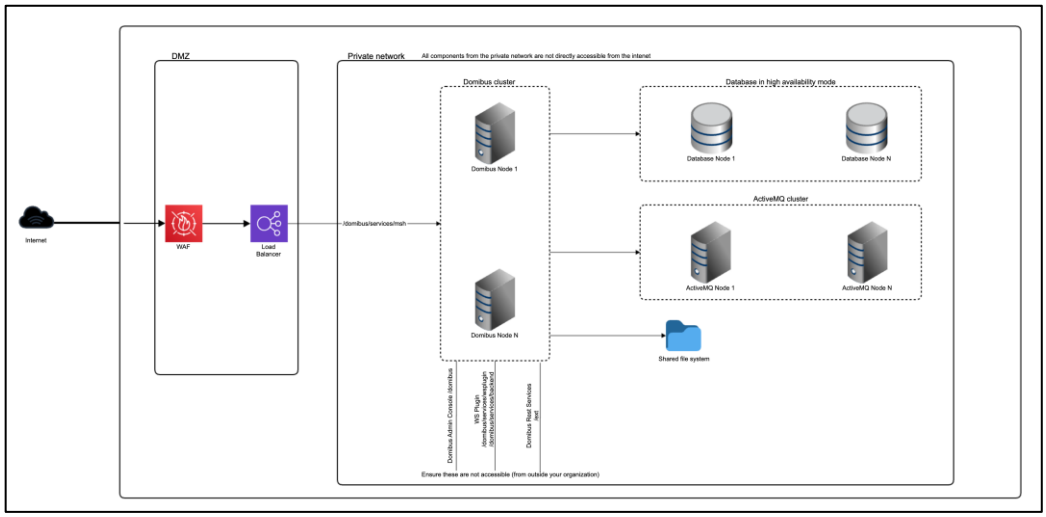


Figure 2 - Diagram representing the Deployment of Domibus in a Cluster on Tomcat

Remark:

In this section we assume that a JMS Broker and a Loadbalancer are configured separately (e.g. httpd).

For this step, you will have to use the following resources (see §3.1–" *Binaries repository*" for the download location):

- **domibus-msh-distribution-X.Y.Z-tomcat-full.zip**
- **domibus-msh-distribution-X.Y.Z-tomcat-war.zip**

1. Follow steps **1**, **2, 3, 4** and **5** from the §4.3.2 – *"Single Server Deployment"*

2. Set the JVM parameters:

Domibus expects a single JVM parameter **$domibus.config.location**, pointing towards the *edelivery_path* folder.

You can do this by editing *edelivery_path***\bin\setenv.bat** (Windows) or *edelivery_path***/bin/setenv.sh** (Linux). Set **CATALINA_HOME** equal to the absolute path of the installation *edelivery_path*.

- For Windows**:** edit *edelivery_path***\bin\setenv.bat** by adding the following:

Remark:

your_node_id refers to the installed node in the cluster which starts normally at 01(then 02, etc.).

```
…
set CATALINA_HOME=edelivery_path
set CATALINA_TMPDIR=<path to _tmp directory>
set  JAVA_OPTS=%JAVA_OPTS% -Dfile.encoding=UTF-8 -Xms128m -Xmx1024m -XX:PermSize=64m
set JAVA_OPTS=%JAVA_OPTS% -Ddomibus.config.location=%CATALINA_HOME%\conf\domibus
set JAVA_OPTS=%JAVA_OPTS% -Ddomibus.node.id=your_node_id
…
```

- For Linux: edit *edelivery_path***/bin/setenv.sh** by adding the following:

```
…
export CATALINA_HOME=edelivery_path
export CATALINA_TMPDIR=<path to _tmp directory>
export JAVA_OPTS=$JAVA_OPTS –Xms128m –Xmx1024m
export JAVA_OPTS="$JAVA_OPTS -Ddomibus.config.location=$CATALINA_HOME/conf/domibus"
export JAVA_OPTS="$JAVA_OPTS -Ddomibus.node.id=your_node_id"
…
```

3. Integrate the external JMS Broker with Domibus by adapting the following properties in *edelivery_path***/conf/domibus/domibus.properties**.
   - Please note that the **activeMQ.embedded.configurationFile** property should be deleted as the JMS broker is external.

```
#ActiveMQ
activeMQ.broker.host=localhost
activeMQ.brokerName=localhost
#activeMQ.embedded.configurationFile=file:///${domibus.config.location}/internal/activemq.xml
activeMQ.connectorPort=1199
activeMQ.rmiServerPort=1200
activeMQ.transportConnector.uri=tcp://${activeMQ.broker.host}:61616
activeMQ.username=domibus
activeMQ.password=changeit
```

4. Change the following properties related to the **Atomikos** configuration in parameters in *edelivery_path***/conf/domibus/domibus.properties**:

For clustered deployment:

Uncomment the following lines:

#com.atomikos.icatch.output_dir=*${domibus.work.location:${domibus.config.location}}/work/transactions/${domibus.node.id}*
#com.atomikos.icatch.log_base_dir=*${domibus.work.location:${domibus.config.location}}/work/transactions/${domibus.node.id}/log*

Comment the following lines:
com.atomikos.icatch.output_dir=${domibus.work.location:${domibus.config.location}}/work/transactions
com.atomikos.icatch.log_base_dir=${domibus.work.location:${domibus.config.location}}/work/transactions/log
Set the *domibus.deployment.clustered* option to true:
domibus.deployment.clustered=true

5. Follow step **6** and **7** from the §4.3.2 – *"Single Server Deployment".*

4.4. Domibus on WildFly

Remark:

- *The Apache CXF library referred to by Domibus, uses the environment variable java.io.tmpdir to buffer large attachments received, internally. If the java.io.tmpdir property is not specified, then this defaults to values provided by the operating system to the JRE. On Unix/Linux systems this usually defaults to '/tmp'. On Windows systems this usually defaults to %TEMP% folder. It is recommended to point this to a local directory '_tmp' on each managed server and accessible by the WildFly server. The disk space allocated for '_tmp' directory would depend on the size of attachments received. On production environment it is recommended to provide 100GB for '_tmp'.*

- *CXF has a limitation of being able to validate signatures of only 28 payload attachments at a time. As a result, the Domibus cannot send/receive more than 28 attachments in a single AS4 message.*

4.4.1. *Pre-Configured Single Server Deployment*

For this step, you will have to use the following resources (see section §3.1–" *Binaries repository*" for the download location):

- **domibus-msh-distribution-X.Y.Z-wildfly-full.zip (WildFly 26.1.x version)**

Remark: below steps apply for both distributions of Domibus.

1. Download and unzip the **domibus-msh-distribution-X.Y.Z-wildfly-full.zip** archive in your *edelivery_path* location.

2. Configure the MySQL database (Option 1).

   - Drivers:

     Create the directory *edelivery_path***/modules/system/layers/base/com/mysql/main** if it does not exist**.**
     Under this directory:

     - Download the MySQL JDBC driver available on MySQL official web site (cf.[REF2]) and copy it in the folder.

       Remark:

       The version of the driver has to be mysql-connector-java-8.0.23.jar or higher.

     - Create or edit the file *edelivery_path***/modules/system/layers/base/com/mysql/main/module.xml** and copy the following module configuration. Make sure to type the name of the driver you use as an argument of **resource-root** element. e.g. ***mysql-connector-java-8.0.23.jar***:

```
<module xmlns="urn:jboss:module:1.3" name="com.mysql">
        <resources>
                <resource-root path="mysql-connector-java-8.0.23.jar"/>
        </resources>
        <dependencies>
                <module name="javax.api"/>
                <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

     - Add your DBMS driver metadata to the Drivers section of the *edelivery_path***/standalone/configuration/standalone-full.xml**.

```
<subsystem xmlns="urn:jboss:domain:datasources:6.0">
  ..............................
  <datasources>
    ..............................
    <drivers>
        <driver name="h2" module="com.h2database.h2">
                <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>

           <driver name="com.mysql" module="com.mysql">
       <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
       <xa-datasource-class>com.mysql.cj.jdbc. MysqlXADataSource</xa-datasource-class>
        -->
     </driver>
```

```
            <!--Oracle

            <driver name="com.oracle" module="com.oracle">

              <driver-class>oracle.jdbc.driver.OracleDriver</driver-class>

              <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource </xa-datasource-class>

            </driver>
            -->
    <drivers>
   ……
  </datasources>
  ……
</subsystem>
```

- Datasources:
  - Add the datasources as indicated below to *edelivery_path***/standalone/configuration/standalone-full.xml**.

Remark:

- *Please make sure you modify the connection details for the **edeliveryMysqlDS** datasource for MySQL according to your environment.*
- *See ORACLE related changes in **option 2** below when Oracle is used instead of MySQL.*

```
<subsystem xmlns="urn:jboss:domain:datasources:6.0">
<datasources>
……
<xa-datasource jndi-name="java:/jdbc/cipaeDeliveryDs" pool-
name="eDeliveryMysqlXADS" enabled="true" use-ccm="true" statistics-enabled="true">
        <connection-
url>jdbc:mysql://localhost:3306/domibus_schema?autoReconnect=true&useSSL-
false&useLegacyDatetimeCode=false&serverTimezone=UTC</connection-url>

        <!--Connector/J 8.0.x ->
        <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
        <driver>com.mysql</driver>
         <pool>
                <min-pool-size>20</min-pool-size>
                <initial-pool-size>5</initial-pool-size>
                <max-pool-size>100</max-pool-size>
        </pool>             <security>
            <user-name>edelivery_user</user-name>
            <password>edelivery_password</password>
          </security>
<validation>
<valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
<background-validation>true</background-validation>
<exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
                </validation>
```

```
        </datasource>
<datasource jndi-name="java:/jdbc/cipaeDeliveryNonXADs" pool-name="eDeliveryMysqlNonXADS"
enabled="true" use-ccm="true">
  <connection-url>jdbc:mysql://localhost:3306/domibus_schema?autoReconnect=true&useSSL-
false&useLegacyDatetimeCode=false&serverTimezone=UTC </connection-url>
  <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
  <!--Connector/J 8.0.x ->

        <driver-class>com.mysql.cj.jdbc.Driver</driver-class>   <driver>com.mysql</driver>
        <pool>
                <min-pool-size>20</min-pool-size>
                <initial-pool-size>5</initial-pool-size>
                <max-pool-size>100</max-pool-size>
        </pool>
  <security>
    <user-name>edelivery_user</user-name>
    <password>edelivery_password</password>
  </security>
  <validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
    <background-validation>true</background-validation>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
  </validation>
</datasource>
……..
        </datasources>
</subsystem>
```

3. Configure the Oracle Database (option 2):
    - Drivers:

Create the directory *edelivery_path***/modules/system/layers/base/com/oracle/main** if it does not exist. Under this directory:

- Download and copy the Oracle JDBC driver (e.g. ***ojdbc8-21.1.0.0.jar***, available on the Oracle official web site cf.[REF3]) in the folder.
    - Create or edit the file *edelivery_path***/modules/system/layers/base/com/oracle/main/module.xml** in the recently created folder.

      Add the following module configuration. Make sure to type the name of the driver you use as an argument of **resource-root** element. e.g. ***ojdbc8-21.1.0.0.jar***:

```
<module xmlns="urn:jboss:module:1.3" name="com.oracle">
        <resources>
                <resource-root path="ojdbc8-21.1.0.0.jar "/>
        </resources>
        <dependencies>
                <module name="javax.api"/>
                <module name="javax.transaction.api"/>
        </dependencies>
</module>
```

- Uncomment Oracle paragraph from the Drivers section in *edelivery_path***/standalone/configuration/standalone-full.xml** .

```
<subsystem xmlns="urn:jboss:domain:datasources:6.0">
  ……….
  <datasources>
    ……….
    <drivers>
        <driver name="h2" module="com.h2database.h2">
                <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>

             <!--<driver name="com.mysql" module="com.mysql">-->
       <!--Connector/J 8.0.x ->
       <!--<driver-class>com.mysql.cj.jdbc.Driver</driver-class>
       <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>

      </driver>-->
            <!—Oracle ->

            <driver name="com.oracle" module="com.oracle">

              <driver-class>oracle.jdbc.driver.OracleDriver</driver-class>

              <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource </xa-datasource-class>

            </driver>
            -->
    <drivers>
    ……..
  </datasources>
  ……….
</subsystem>
```

- Datasources:
  - Uncomment the Oracle paragraph from the datasources section of *edelivery_path***/standalone/configuration/standalone-full.xml**.

Remark:

Please make sure you modify the connection details for both ***eDeliveryOracleNonXADS*** *and* ***eDeliveryOracleDS*** *datasources for Oracle according to your environment.*

```
<!-- Oracle
<datasource jta="true" jndi-name="java:/jdbc/cipaeDeliveryNonXADs" pool-
name="eDeliveryOracleNonXADS" enabled="true" use-ccm="true">
   <connection-url>jdbc:oracle:thin:@localhost:1521[:SID|/Service]</connection-url>
   <driver-class>oracle.jdbc.OracleDriver</driver-class>
   <driver>com.oracle</driver>
      <pool>
               <min-pool-size>20</min-pool-size>
               <initial-pool-size>5</initial-pool-size>
               <max-pool-size>100</max-pool-size>
     </pool>
   <security>
      <user-name>edelivery_user</user-name>
      <password>edelivery_password</password>
   </security>
   <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
      <background-validation>true</background-validation>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"/>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
   </validation>
</datasource>
-->

…
<!-- Oracle
<datasource jndi-name="java:/jdbc/cipaeDeliveryDs" pool-name="eDeliveryOracleDS"
enabled="true" use-ccm="true">
   <connection-url>jdbc:oracle:thin:@localhost:1521[:SID|/Service]</connection-url>
   <driver-class>oracle.jdbc.OracleDriver</driver-class>
   <driver>com.oracle</driver>
     <pool>
               <min-pool-size>20</min-pool-size>
               <initial-pool-size>5</initial-pool-size>
               <max-pool-size>100</max-pool-size>
     </pool>
   <security>
      <user-name>edelivery_user</user-name>
      <password>edelivery_password</password>
   </security>
   <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
   </validation>
</datasource>
-->
```

- Edit the configuration file *edelivery_path***/conf/domibus/domibus.properties** and configure the datasources as indicated below.

Remark:

Configure the database dialect as it is pre-configured for MySQL by default.

```
#EntityManagerFactory
domibus.entityManagerFactory.jpaProperty.hibernate.connection.driver_class=
oracle.jdbc.driver.OracleDriver
domibus.entityManagerFactory.jpaProperty.hibernate.dialect=org.hibernate.dialect.Oracle10gDialect
```

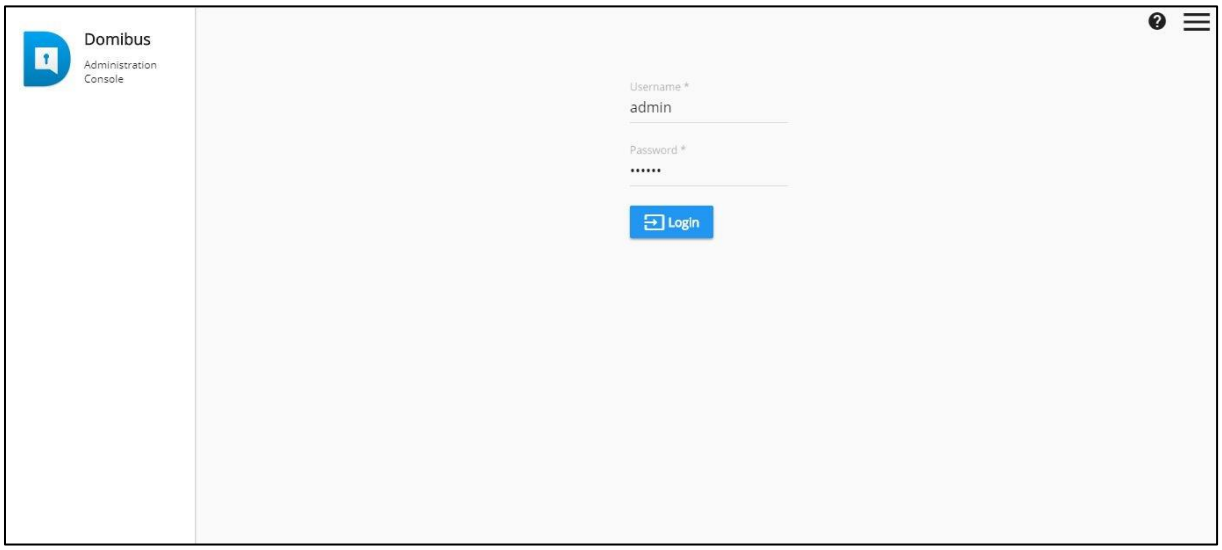
4. Configure your Keystore based on §5.1.3 – *"Certificates"*.
5. Run the standalone server:
   - For Windows under *edelivery_path***\bin**
   - **standalone.bat --server-config=standalone-full.xml**
   - For Linux under *edelivery_path***/bin/**
   - **standalone.sh --server-config=standalone-full.xml**
6. Display the Domibus home page in your browser: http://*localhost*:*8080*/domibus
   (by default: User = ***admin***; for the password, look in the logs for the phrase: "Default password for user admin is"*).*

   ***Remark**:*

   It is recommended to change the passwords for the default users. See §10.1 – *"Administration " for further information.*

   If you can access the page it means the deployment was successful.

Expected result:



4.4.2. Single Server Deployment

4.4.2.1. WildFly 26.1.x Configuration

In this section we assume that WildFly version 26.1.x is installed at location *edelivery_path*.

For this step, you will have to use the following resources (see section §3.1-"*Binaries repository* " for the download location):

- **domibus-msh-distribution-X.Y.Z-wildfly-war.zip**
- **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip**

1. Run the WildFly 26.1 JBOSS CLI to configure *edelivery_path***/standalone/configuration/standalone-full.xml** from the command line:
    - Extract the configuration scripts from the **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip** file under the scripts directory
    - Configure the JBOSS CLI tool
        i. For Windows:
        configure.bat
        ii. For Linux:
        configure.sh
    - Extract the script **configure.[bat|sh]** from **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip** under the scripts directory and adapt the following properties:
        i. For Windows:

Remark:

*The **configure.bat** script uses Windows Powershell present on machines running Windows 7 SP1 or later.*

1. Common to Oracle and MySQL:

```
SET JBOSS_HOME=C:\path\to\wildfly
SET SERVER_CONFIG=standalone-full.xml
```

2. For Oracle database:

```
SET DB_TYPE=Oracle
SET DB_HOST=localhost
SET DB_PORT=1521
SET DB_USER=edelivery_user
SET DB_PASS=edelivery_password
SET
JDBC_CONNECTION_URL="jdbc:oracle:thin:@%DB_HOST%:%DB_PORT%[:SID/Service
]"
SET
ORACLE_JDBC_DRIVER_DIR=%JBOSS_HOME%\modules\system\layers\base\com\or
acle\main
SET ORACLE_JDBC_DRIVER_NAME=ojdbc-X.Y.Z.jar
```

Remark:

Oracle configuration is commented by default. To enable Oracle, remove the comment (::) from the lines below. Do not forget to add the comment (::) for MySQL to disable it.

3. For MySQL:

```
SET DB_TYPE=MySQL
SET DB_HOST=localhost
SET
"DB_NAME=domibus_schema?autoReconnect=true^&useSSL=false^&useLegacyDate
timeCode=false^&serverTimezone=UTC"
SET DB_PORT=3306
SET DB_USER=edelivery
SET DB_PASS=edelivery
SET JDBC_CONNECTION_URL=jdbc:mysql://%DB_HOST%:%DB_PORT%/!DB_NAME!
SET
MYSQL_JDBC_DRIVER_DIR=%JBOSS_HOME%\modules\system\layers\base\com\my
sql\main
SET MYSQL_JDBC_DRIVER_NAME=mysql-connector-java-X.Y.Z.jar
```

ii. For Linux:

1. Common to Oracle and MySQL

```
JBOSS_HOME=/path/to/wildfly
SERVER_CONFIG=standalone-full.xml
```

2. For Oracle database:

```
DB_TYPE=Oracle
DB_HOST=localhost
DB_PORT=1521
DB_USER=edelivery_user
DB_PASS=edelivery_password
```

```
JDBC_CONNECTION_URL="jdbc:oracle:thin:@${DB_HOST}:${DB_PORT}[:SID/Service]
"
ORACLE_JDBC_DRIVER_DIR=${JBOSS_HOME}/modules/system/layers/base/com/ora
cle/main
ORACLE_JDBC_DRIVER_NAME=ojdbc-X.Y.Z.jar
```

Remark:

Oracle configuration is commented by default. To enable Oracle, remove the comment (#) from the lines below. Do not forget to add the comment (#) for MySQL to disable it.

3. For MySQL:

```
DB_TYPE=MySQL
DB_HOST=localhost
DB_NAME=domibus_schema?autoReconnect=true\&useSSL=false\&useLegacyDateti
meCode=false\&serverTimezone=UTC
DB_PORT=3306
DB_USER=edelivery_user
DB_PASS=edelivery_password
JDBC_CONNECTION_URL=jdbc:mysql://${DB_HOST}:${DB_PORT}/${DB_NAME}
MYSQL_JDBC_DRIVER_DIR=${JBOSS_HOME}/modules/system/layers/base/com/mys
ql/main
MYSQL_JDBC_DRIVER_NAME=mysql-connector-java-X.Y.Z.jar
```

- Execute the following command from within the **scripts** directory:

  i. For Windows:

  configure.bat

  ii. For Linux:

  configure.sh

Expected result:

```
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{"outcome" => "success"}
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
Press any key to continue . . .
```

2. Follow steps **2** (MySQL) or **3** (Oracle) from the §4.4.1 – *"Pre-Configured Single Server Deployment",* and replace the "**%JDBC_DRIVER_DIR%\%JDBC_DRIVER_NAME%"** (Windows) or "**${JDBC_DRIVER_DIR}\${JDBC_DRIVER_NAME}"** (Linux) directory with the current JDBC file.

Remark:

*The edelivery_path***/standalone/configuration/standalone-full.xml,** *the edelivery_path***/modules/system/layers/base/com/mysql/main/module.xml** *and the edelivery_path***/modules/system/layers/base/com/oracle/main/modules.xml** *files should already have the correct details filled in.*

3. Configure the environment variables:

   For Windows: edit edelivery_path/bin/standalone.conf.bat as follows:

```
……
set "JAVA_OPTS=-Xms128m -Xmx1024m -Djava.net.preferIPv4Stack=true"
set "JAVA_OPTS=%JAVA_OPTS% -Ddomibus.config.location=<path to conf directory> -
Djava.io.tmpdir=<path to _tmp directory>"
set "JBOSS_JAVA_SIZING=-Xms1024M -Xmx4096M -XX:MetaspaceSize=96M -
XX:MaxMetaspaceSize=256m -Ddomibus.config.location=%JBOSS_HOME%/conf/domibus"
……
```

4. For Unix/Linux: edit edelivery_path/bin/standalone.conf as follows:

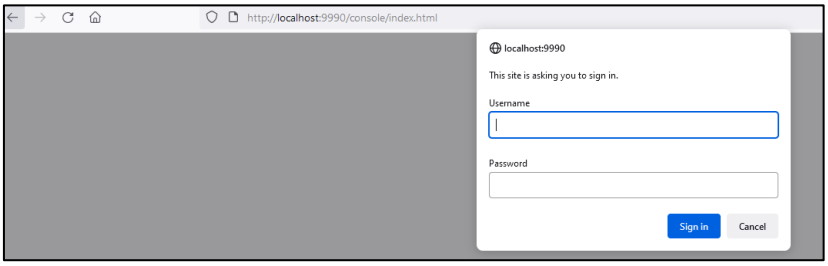
```
……
JAVA_OPTS="-Xms128m –Xmx1024m
 java.net.preferIPv4Stack=true"
JAVA_OPTS="$JAVA_OPTS -Ddomibus.config.location=$JBOSS_HOME/conf/Domibus domibus -
Djava.io.tmpdir=<path to _tmp directory>"

JAVA_OPTS="-Xms64m -Xmx6g -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -
Djava.net.preferIPv4Stack=true -Ddomibus.config.location=$JBOSS_HOME/conf/domibus"
……
```

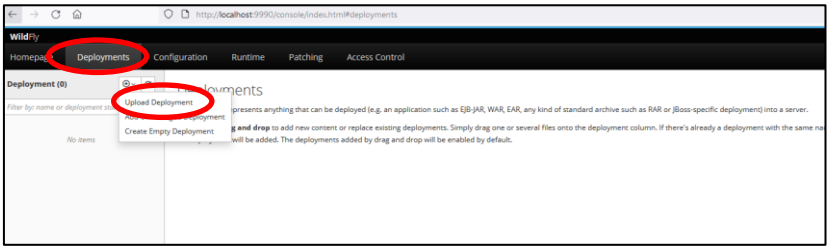
5. Download and unzip **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip** in the directory *edelivery_path***/conf/domibus,** excluding the scripts directory.

6. Configure your Keystore based on §5.1.3 – *"Certificates".*

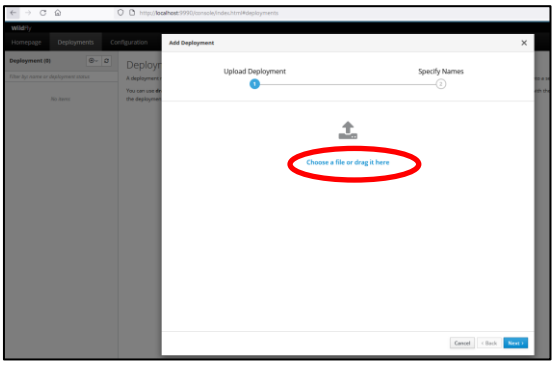
7. Connect to the Admin Console of WildFly at **http://*localhost*:*9990/console***:



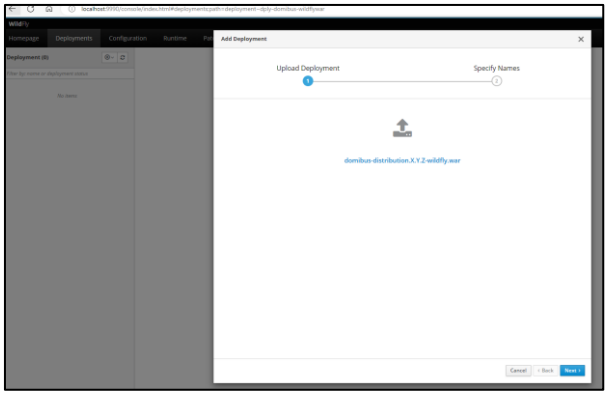
8. Click on **Deployments** in the console menu then click on **Add**:



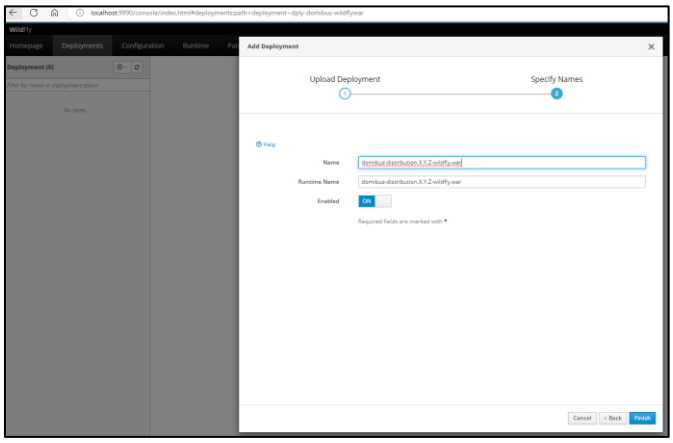
9. Select **Choose a file or drag it here**:



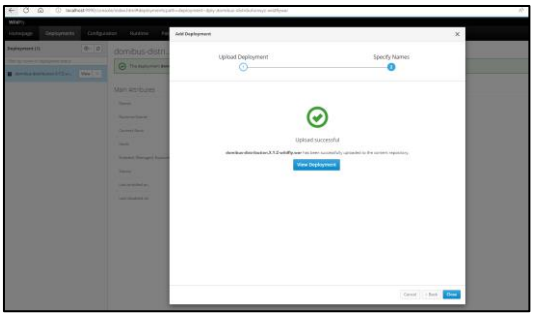
10. Browse to the location of the **domibus-msh-distribution-X.Y.Z-wildfly.war** file, select it and click **Next**:



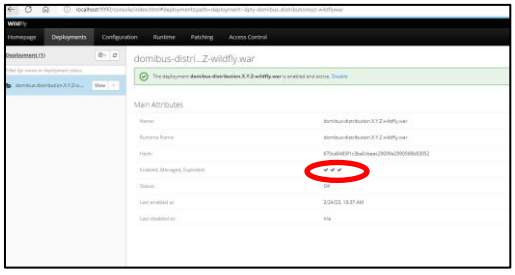
11. Click **Finish**:



12. The deployment is successful when the name of the .war file appears in the Deployment column.



13. Enabled, Managed and exploded must be ticked:



Expected result:

14. In case of WildFly upgrade of single server, you must delete the previously cached version of Domibus.
    Therefore, you must delete the following folders:
    => edelivery_path\standalone\data
    => edelivery_path\standalone\tmp

    Old deployed versions of domibus-msh-distribution-X.Y.Z-wildfly.war also have to be deleted from the path edelivery_path\standalone\deployments or they must be removed via the WildFly Admin Console.

4.4.3. Clustered Deployment

For this step, you will have to use the following resources (see section §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip**
- **domibus-msh-distribution-X.Y.Z-wildfly-war.zip**

In this section we assume that the setup of WildFly in domain mode has already been done and that the cluster has been enabled as described in the official documentation. For more details on how to perform an installation of WildFly in domain mode, please refer to the official documentation (cf.[REF4]).

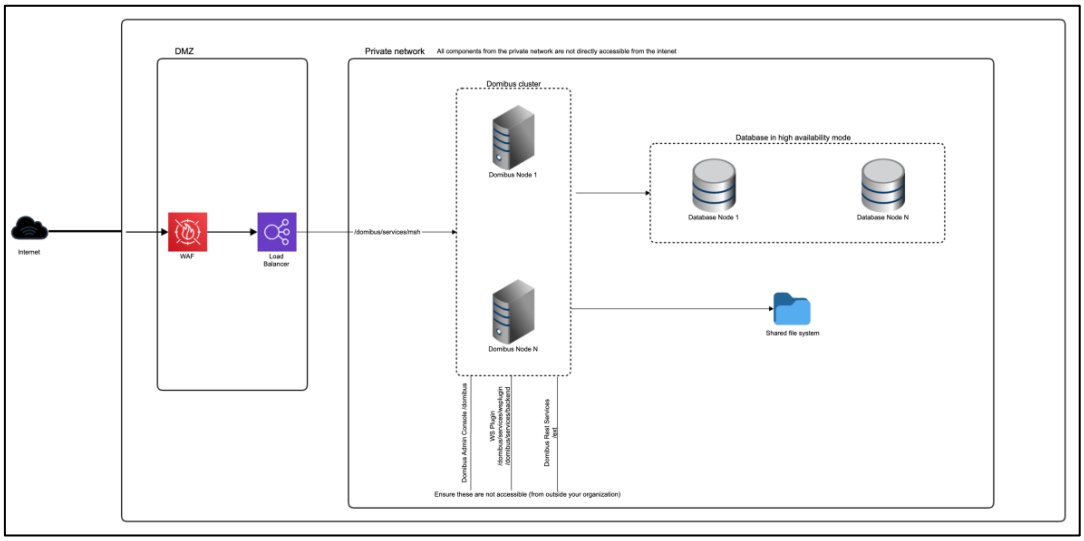


Figure 3 - Diagram representing the Deployment of Domibus in a Cluster on WildFly

In order to install Domibus in a WildFly cluster please follow the steps below:

1. Download and unzip **domibus-msh-distribution-X.Y.Z-wildfly-configuration.zip** (for WildFly 26.1) in a shared location that is accessible by all the nodes from the cluster. We will refer to this directory as *shared_edelivery_path*.

2. Follow steps **2** (MySQL) or **3** (Oracle) from the §4.4.1 – *"Pre-Configured Single Server Deployment"*.

   Remarks:

   - *This step needs to be performed on all the nodes from the cluster*
   - *In the following 2 steps we will edit the profile **full-ha** from the configuration file **domain/configuration/domain.xml** located in the master node*

3. Configure the JMS resources in the configuration file *edelivery_path***/standalone/configuration/standalone-full-ha.xml** by adding the **jms-connection-factories** and **jms-queues**.

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:3.0">
    <server name="default">
        <management jmx-enabled="true"/>

        <!--default for catch all-->
```

```
        <address-setting name="#"
                dead-letter-address="jms.queue.DLQ"
                expiry-address="jms.queue.ExpiryQueue"
                max-size-bytes="10485760"
                page-size-bytes="2097152"
                message-counter-history-day-limit="10"
                redistribution-delay="1000"/>
        <address-setting name="jms.queue.DomibusSendMessageQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="1000"
                max-delivery-attempts="1"/>
        <address-setting name="jms.queue.DomibusSendLargeMessageQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="1000"
                max-delivery-attempts="1"/>
        <address-setting name="jms.queue.DomibusSplitAndJoinQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="1000"
                max-delivery-attempts="1"/>
        <address-setting name="jms.queue.DomibusPullMessageQueue"
                expiry-address="jms.queue.ExpiryQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                redelivery-delay="1000"
                max-delivery-attempts="1"/>
        <address-setting name="jms.queue.DomibusPullReceiptQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="1000"
                max-delivery-attempts="3"/>
        <address-setting name="jms.queue.DomibusRetentionMessageQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="10000"
                max-delivery-attempts="0"/>
        <address-setting name="jms.queue.DomibusAlertMessageQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                max-delivery-attempts="1"/>
        <address-setting name="jms.queue.DomibusBusinessMessageOutQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusNotifyBackendJmsQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusErrorNotifyConsumerQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusErrorNotifyProducerQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusBusinessMessageInQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
```

```
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusPluginToBackendQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusNotifyBackendWebServiceQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusNotifyBackendFileSystemQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusUnknownReceiverQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusNotifyBackendQueue"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="300000"
                max-delivery-attempts="10"/>
        <address-setting name="jms.queue.DomibusFSPluginSendQueue"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="10000"
                max-delivery-attempts="0"/>
        <address-setting name="jms.queue.DomibusClusterCommandTopic"
                dead-letter-address="jms.queue.DomibusDLQ"
                expiry-address="jms.queue.ExpiryQueue"
                redelivery-delay="10000"
                max-delivery-attempts="3"/>
.....
        <connection-factory name="edeliveryConnectionFactory"
                entries="java:/jms/ConnectionFactory"
                discovery-group="dg-group1"
                compress-large-messages="false"
                failover-on-initial-connection="false"
                use-global-pools="true"/>
.....
        <jms-queue name="DomibusBusinessMessageOutQueue"
                entries="java:/jms/domibus.backend.jms.outQueue
java:/jms/queue/DomibusBusinessMessageOutQueue"
                durable="true"/>

        <jms-queue name="DomibusNotifyBackendJmsQueue"
                entries="java:/jms/domibus.notification.jms
java:/jms/queue/DomibusNotifyBackendJmsQueue"
                durable="true"/>

        <jms-queue name="DomibusErrorNotifyConsumerQueue"
                entries="java:/jms/domibus.backend.jms.errorNotifyConsumer
java:/jms/queue/DomibusErrorNotifyConsumerQueue"
```

```
                durable="true"/>

        <jms-queue name="DomibusErrorNotifyProducerQueue"
                entries="java:/jms/domibus.backend.jms.errorNotifyProducer
java:/jms/queue/DomibusErrorNotifyProducerQueue"
                durable="true"/>

        <jms-queue name="DomibusBusinessMessageInQueue"
                entries="java:/jms/domibus.backend.jms.inQueue
java:/jms/queue/DomibusBusinessMessageInQueue"
                durable="true"/>

        <jms-queue name="DomibusPluginToBackendQueue"
                entries="java:/jms/domibus.backend.jms.replyQueue
java:/jms/queue/DomibusPluginToBackendQueue"
                durable="true"/>

        <jms-queue name="DomibusSendMessageQueue"
                entries="java:/jms/domibus.internal.dispatch.queue
java:/jms/queue/DomibusSendMessageQueue"
                durable="true"/>

        <jms-queue name="DomibusSendLargeMessageQueue"
                entries="java:/jms/domibus.internal.largeMessage.queue
java:/jms/queue/DomibusSendLargeMessageQueue"
                durable="true"/>

        <jms-queue name="DomibusSplitAndJoinQueue"
                entries="java:/jms/domibus.internal.splitAndJoin.queue
java:/jms/queue/DomibusSplitAndJoinQueue"
                durable="true"/>

        <jms-queue name="DomibusPullMessageQueue"
                entries="java:/jms/domibus.internal.pull.queue
java:/jms/queue/DomibusPullMessageQueue"
                durable="true"/>

        <jms-queue name="DomibusPullReceiptQueue"
                entries="java:/jms/domibus.internal.pull.receipt.queue
java:/jms/queue/DomibusPullReceiptQueue"
                durable="true"/>

        <jms-queue name="DomibusRetentionMessageQueue"
                entries="java:/jms/domibus.internal.retentionMessage.queue
java:/jms/queue/DomibusRetentionMessageQueue"
                durable="true"/>

        <jms-queue name="DomibusAlertMessageQueue"
                entries="java:/jms/domibus.internal.alert.queue
java:/jms/queue/DomibusAlertMessageQueue"
                durable="true"/>

        <jms-queue name="DomibusNotifyBackendWebServiceQueue"
                entries="java:/jms/domibus.notification.webservice
java:/jms/queue/DomibusNotifyBackendWebServiceQueue"
                durable="true"/>

        <jms-queue name="DomibusNotifyBackendFileSystemQueue"
```

```
                entries="java:/jms/domibus.notification.filesystem
java:/jms/queue/DomibusNotifyBackendFileSystemQueue"
                durable="true"/>

        <jms-queue name="DomibusUnknownReceiverQueue"
                entries="java:/jms/domibus.internal.notification.unknown
java:/jms/queue/DomibusUnknownReceiverQueue"
                durable="true"/>

        <jms-queue name="DomibusNotifyBackendQueue"
                entries="java:/jms/domibus.internal.notification.queue
java:/jms/queue/DomibusNotifyBackendQueue"
                durable="true"/>

        <jms-queue name="DomibusFSPluginSendQueue"
                entries="java:/jms/domibus.fsplugin.send.queue
java:/jms/queue/DomibusFSPluginSendQueue"
                durable="true"/>

        <jms-queue name="DLQ"
                entries="java:/jms/domibus.DLQ java:/jms/queue/DomibusDLQ"
                durable="true"/>

        <jms-topic name="DomibusClusterCommandTopic"
                entries="java:/jms/domibus.internal.command
java:/jms/topic/DomibusClusterCommandTopic"/>
    </server>
</subsystem>
```

Remark:

Please note that the JMX management also has to be enabled so the JMS resources can be monitored in the JMS Monitoring screen.

4. Configure the database dialect as indicated in §4.4.1 point 3 *- Configure the Oracle Database (option 2)*.

5. Configure the environment variables in the file **bin/domain.conf**.

6. Set the domibus.deployment.clustered option to true:

```
domibus.deployment.clustered=true
```

Remark:

bin/domain.conf is located in each WildFly node. The environment variable setting needs to be performed in every node from the cluster.

```
……
JAVA_OPTS="-Xms128m –Xmx1024m
-java.net.preferIPv4Stack=true"
JAVA_OPTS="$JAVA_OPTS -Ddomibus.config.location=shared_edelivery_path/conf/Domibus -
Djava.io.tmpdir=<path to _tmp directory>"
……
```

7. Deploy the **domibus-msh-distribution-X.Y.Z-wildfly.war** (for WildFly 26.1.x) to the cluster. We will use the WildFly Administration console for performing the deployment. We will deploy the application on the **other-server-group** cluster which is configured step by step in the official documentation (cf.[REF4]).

8. For the upgrade of clustered WildFly server, you must delete the previously cached version of Domibus from the **domain** before adding the new **distribution-X.Y.Z-wildfly.war**.
   Make sure to remove all old versions of **domibus-msh-distribution-X.Y.Z-wildfly.war** if you use the WildFly administration console for the deployment.

4.5. Domibus Secure Deployment Recommendations

4.5.1. *Introduction*

This section provides essential guidelines and best practices for ensuring a secure and robust deployment of the Domibus Access Point.

These recommendations focus on key aspects of deploying Domibus securely, addressing considerations such as system architecture, network configurations, access controls. By adhering to these guidelines, organizations can enhance the overall security posture of their Domibus deployments.

Starting on 1 May 2024, the eDelivery team will assume that these recommendations are followed when assessing the impact of security vulnerabilities on a Domibus deployment.

4.5.2. *Recommendations*

An overview is provided for each server for a typical deployment topology, please check the specific server sections in this chapter.

- All the components from the private network should not be accessible directly from the internet.
- Restrict access to each component and allow access only to specific ports only from know components. For example Domibus instances should allow incoming traffic only from the load balancer on port 7001 for WebLogic or 8080 for Tomcat, the database should allow incoming traffic only from the Domibus instances on port 1521 for Oracle or 3306 for MySQL, etc.
- Implement a web application firewall before the load balancer to filter and monitor HTTPS from the internet.

- Implement a load balancer to balance the traffic between Domibus instances.
- Restrict access from the internet or at most only allow traffic from your organization to the following resources:
    - Domibus Admin Console: /domibus;
    - the plugin interfaces should be restricted if the plugin is not used or else it should be only be accessible from the backend system(s) connected to Domibus instance:
        - new WS plugin: /domibus/services/wsplugin
        - old WS plugin: /domibus/services/backend
        - JMS plugin: the JMS broker port(s)
        - file system plugin: shared file system
        - custom plugins: plugin dependent
        - the Domibus REST services: /ext
- Only the MSH endpoint /domibus/services/msh should be accessible from the Internet.

5. DOMIBUS CONFIGURATION

Domibus exposes the Message Service Handler endpoint as **../services/msh**. Only this endpoint has to be reachable by the other AS4 Access Points and it is typically exposed on the Internet.

If the Default WS Plugin (§6.1.2 – *"WS Plugin"*) is deployed, Domibus exposes the Default WS Plugin endpoint as **../services/backend**. This endpoint should ONLY be exposed to the backend client(s) within the trusted zone and it should not be exposed to the Internet.

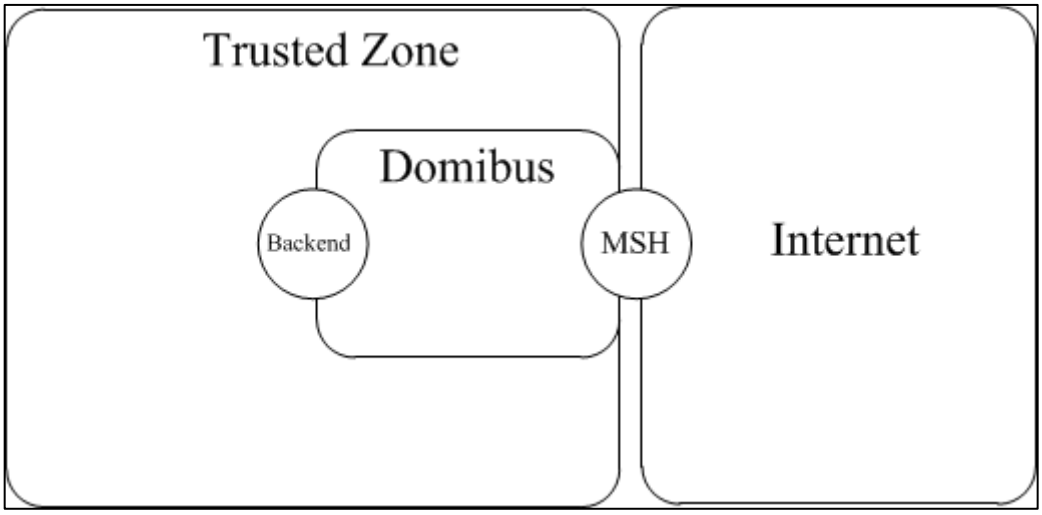


Figure 4 - Message Service Handler diagram

5.1. Security Configuration

5.1.1. Security Policies

The WS-Security policy used by Domibus when exchanging messages can be specified in the PMode configuration file (§7 – *"PMode Configuration"*).

Security policy assertions are based on the ***WS-Policy framework***.

As requested by the eDelivery AS4 profile, Domibus supports all three mechanisms to reference a security token, as described below.

Domibus distribution includes one policy file for each mechanism (edelivery_path/conf/domibus/policies/):

eDeliveryAS4Policy.xml - Reference to a Subject Key Identifier

The <wsse:SecurityTokenReference> element contains a <wsse:KeyIdentifier> element that specifies the token data by means of a X.509 SubjectKeyIdentifier reference. A subject key identifier MAY only be used to reference an X.509v3 certificate.

eDeliveryAS4Policy_BST.xml - Reference to a Binary Security Token

The <wsse:SecurityTokenReference> element contains a wsse:Reference> element that references a local <wsse:BinarySecurityToken> element or a remote data source that contains the token data itself.

eDeliveryAS4Policy_IS.xml - Reference to an Issuer and Serial Number

The <wsse:SecurityTokenReference> element contains a <ds:X509Data> element that contains a <ds:X509IssuerSerial> element that uniquely identifies an end entity certificate by its X.509 Issuer and Serial Number.

With the **eDeliveryAS4Policy.xml**, Domibus is able to receive messages with **all 3 referencing methods**. When eDeliveryAS4Policy_BST.xml or eDeliveryAS4Policy_IS.xml are used, the specific reference method becomes mandatory on both APs involved in the exchange.

For the connectivity with other APs, the three policies may be combined to obtain the required references for initiator/responder and signing/encryption tokens.

In order to validate a certificate chain contained in incoming messages with DSS (see § 21 - DSS extension configuration), Domibus also supports:

eDeliveryAS4Policy_BST_PKIP.xml - Reference to a Pki Path Binary Security Token

The <wsse:SecurityTokenReference> element contains a wsse:Reference> element that references a local <wsse:BinarySecurityToken> element or a remote data source that contains the token data itself.

With the above policy the entire certificate chain is added to the the Ws-Security header of the AS4 message.

5.1.2. Default authorization

When a message is received by Domibus MSH, the default authorization service performs authorization checks on the signing certificate: the certificate that was used to sign either the UserMessage or the SignalMessage (for PullRequests).

On other words, the validations are performed by the receiving AP on the sender's certificate for a UserMessage and on the initiator's certificate for a PullRequest.

There are 3 checks that can be enabled/disabled independently:

domibus.sender.trust.validation.truststore_alias: this check verifies that the sender's certificate matches the certificate stored in the truststore. The certificate is loaded from the truststore based on the alias (party name). By default it is set to true.

With this check, it is ensured that when Domibus is configured to receive from multiple parties, these parties cannot impersonate each other.

Example: red_gw is configured to receive from both blue_gw and green_gw. Without this check enabled, blue_gw can sign with its own certificate (which is accepted by the receiving AP) but pretend it is green_gw.

domibus.sender.trust.validation.expression: when this property is not empty, Domibus will verify, before receiving a message, if the subject of the sender's certificate matches the regular expression. By default it is empty, therefore no check is performed.

This property is mainly meant for chain of certificates, where sender's certificate is signed by a certificate authority and the leaf certificate is not present in the truststore of the receiving AP.

domibus.sender.certificate.subject.check: this check verifies that the subject of the sender's certificate contains the alias (party name). Because this check is very restrictive, it is set by default to false.

In addition to these 3 properties, the property **domibus.sender.trust.validation.onreceiving**, when set to false, completely disables the authorization (as well as the certificate validation – valid/expired/revoked).

5.1.3. Certificates

The certificates that are used for signing and encrypting the messages when communicating with the other Access Points can be configured in the property file located under *edelivery_path***/conf/domibus*/domibus.properties.***
By default Domibus is pre-configured to use self-signed certificates. Please note that self-signed certificates should be used only for testing purposes and are not intended for production use.

In order to configure Domibus to use custom certificates the following properties need to be modified:

```
#The location of the keystore
domibus.security.keystore.location=${domibus.config.location}/keystores/gateway_keystore.jks
#Type of the used keystore
domibus.security.keystore.type=jks
#The password used to load the keystore
domibus.security.keystore.password=test123

#Private key
#The alias from the keystore of the private key
domibus.security.key.private.alias=blue_gw
#The private key password
domibus.security.key.private.password=test123

#Truststore
#The location of the truststore
domibus.security.truststore.location=${domibus.config.location}/keystores/gateway_truststore.jks
#Type of the used truststore
domibus.security.truststore.type=jks
#The password used to load the trustStore
domibus.security.truststore.password=test123
```

1. Create, if not present, a folder *edelivery_path***/conf/domibus/keystores**.
2. Get your key pair from an external provider.
   - Self-signed certificates should only be used for testing purposes, not production.
   - If you are interested in using the eDelivery Public Key Infrastructure Solution (cf.[REF5]).
   - For other certificate providers, please refer to the "Guidance on digital certificates used in eDelivery" document (cf.[REF14]).

3. Create, if not present, the public and private keys containers (e.g. *truststore***.jks** *and keystore***.jks**).
4. Import your private key into your keystore.

Remarks:

- *Your private key and your keystore should always stay secret. Please never share them.*
- *The keystore alias has to be the same as the party*
- *It is strongly recommended to use your key pair (private and public key) and the public key of the other participants you trust in two separate containers.*
- *Note that as from Oracle JAVA 11, the KeyStore PKCS12 is the default keystore format and its implementation was moved from the SunJSSE provider. Therefore, if the keystore is in PCKS12 and the Java version is 11, the following error can occur when Domibus tries to load the keystore.: "Could not load key store: keystore password was incorrect" (the password is in fact correct). If such a scenario occurs, the keystore must be recreated from the problematic keystore with a legacy format. To do this, run the following command:*

```
/opt/java/jdk1.8.0_301/bin/keytool -J-Dkeystore.pkcs12.legacy -importkeystore -
srckeystore test-jdk1.8.0_301.p12 -destkeystore test-jdk1.8.0_301-legacy.p12 -
srcstoretype PKCS12 -deststoretype PKCS12
```

5.1.4. Security Profiles

In order to support additional types of cryptography algorithms (e.g. Ellyptic Curve Certificates), besides the already supported RSA algorithms, Domibus supports security profiles (cryptographic profiles). Domibus supports different private keys for signature and decryption which are predefined inside these security profiles.

This allows a choice among several predefined security profiles for the cryptography algorithms, namely ECC and RSA. The user has the option to choose between one of these predefined profiles but cannot create a custom one.

The previous cryptography specification is grouped under the “RSA” security profile.

The security profiles are configured by the administrator on Domibus for both static and dynamic discovery message exchange. Each of these scenarios requires to setup specific settings, as described in the following sections.

The result of the security profiles extension is that when loading a Keystore “.jks” file containing different types of certificates, they are read, validated and used according to the configured security profile.

The settings that need to be configured to activate the security profiles, for both static and dynamic discovery message exchanges, are described below.

Inside the domibus.properties file, the fields corresponding to a specific security profile must be uncommented and the values setup accordingly. There are two available options for each security profile (RSA and ECC):

- Use the same alias and password for both signing and decrypting
- Use different aliases for signing and decrypting

The user must choose one of the security profiles. He must activate one profile and leave the other one commented out. In the example below, the user chooses to use different certificates for signing and decrypting for the ECC profile:

```
# --- ECC Profile ---
# Same alias (and password) for sign and decrypt
#domibus.security.key.private.ecc.alias=blue_gw_ecc
#domibus.security.key.private.ecc.password=test123

#Sign
domibus.security.key.private.ecc.sign.alias=blue_gw_ecc_sign
domibus.security.key.private.ecc.sign.password=test123

#Decrypt
domibus.security.key.private.ecc.decrypt.alias=blue_gw_ecc_decrypt
domibus.security.key.private.ecc.decrypt.password=test123
```

The complete set of options available for the RSA and ECC security profiles are:

```
#---------- Legacy security section (no security profiles) --------

#The alias from the keystore of the private key
#domibus.security.key.private.alias=blue_gw

#The private key password
#domibus.security.key.private.password=test123


# ------------- Security Profiles --------------------

#For enabling Security Profiles for signing and encryption uncomment the following
parameters #accordingly

# --- RSA Profile ---

# Same alias (and password) for sign and decrypt
#domibus.security.key.private.rsa.alias=blue_gw_rsa
#domibus.security.key.private.rsa.password=test123

#Sign
#domibus.security.key.private.rsa.sign.alias=blue_gw_rsa_sign
#domibus.security.key.private.rsa.sign.password=test123

#Decrypt
#domibus.security.key.private.rsa.decrypt.alias=blue_gw_rsa_decrypt
#domibus.security.key.private.rsa.decrypt.password=test123

# --- ECC Profile ---
# Same alias (and password) for sign and decrypt
#domibus.security.key.private.ecc.alias=blue_gw_ecc
#domibus.security.key.private.ecc.password=test123

#Sign

#domibus.security.key.private.ecc.sign.alias=blue_gw_ecc_sign
#domibus.security.key.private.ecc.sign.password=test123
```

```
#Decrypt
#domibus.security.key.private.ecc.decrypt.alias=blue_gw_ecc_decrypt
#domibus.security.key.private.ecc.decrypt.password=test123
```

If one or more security profiles are used, the following legacy fields must be commented out:

```
#The alias from the keystore of the private key
#domibus.security.key.private.alias=blue_gw
#The private key password
#domibus.security.key.private.password=test123
```

In case both legacy single alias keystore and at least one of the security profiles are uncommented, an exception will be raised, and the application will not start correctly. The user must choose either one option or the other.

5.1.4.1. Security Profiles in static discovery message exchange configuration

For the static configuration message exchange pattern, the security of the message exchange is governed by the PMode configuration.

Before setting up the PMode, the Domibus administrator must know the security profile that will be employed for the selected leg.

The security profile needs to be defined as an attribute in the <security> tag from the PMode xml file. The aliases for the corresponding certificates are defined in the domibus.properties files.

The currently available security profiles are: RSA and ECC. The user must choose between one of these predefined values. A sample of two leg security configuration definitions, one using security profile RSA and the other using security profile ECC can be seen below:

```
<securities>
            <security name="eDeliveryAS4PolicyRSA"
              policy="eDeliveryAS4Policy.xml"
              profile = "RSA "/>
     <security name="eDeliveryAS4PolicyECC"
              policy="eDeliveryAS4Policy.xml"
              profile = "ECC"/>
</securities>
```

The signatureMethod field from the old <security> definition is no longer needed when using security profiles. However, for backward compatibility reasons, in applications that do not use security profiles, the old definition can still be used, as in the example below:

```
<security name="eDeliveryAS4Policy"
policy="eDeliveryAS4Policy.xml"
signatureMethod="RSA_SHA256"/>
```

5.1.4.2. Security Profiles in Dynamic Discovery Message Exchange configuration

In the case of dynamic discovery message exchange configuration, an additional setting must be configured, namely the security profiles order list.

This field is defined inside the domibus.properties file. It must be uncommented and setup accordingly:

```
#Priority order of Security Profiles used in Dynamic Discovery to set the transport protocol
domibus.security.profile.order=ECC,RSA
```

The use of this parameter is described next. When setting up a secured connection between two access points in a dynamic discovery message exchange scenario, a request is sent to the SMP to obtain the endpoint information of the receiver. The SMP will return a service metadata response that contains amongst others, a list of transport profiles supported by the receiver. An algorithm will try to match the highest ranking security profile from the priority list defined in domibus.security.profile.order, with a transport profile supported by the receiver. If a match is found, the security profile is successfully selected, and it will be further used for the secured connection. In case a transport profile that corresponds to the first security profile from the priority list is not found, a match is attempted for the next security profile in the list, and this continues until a match is found between the security profile and a transport profile.

The current matching between the security profiles and transport profiles is defined internally in Domibus:

- OASIS:
    - RSA - "bdxr-transport-ebms3-as4-v1p0"
    - ECC - "bdxr-transport-ebms3-as4-EC-sample"
- PEPPOL:
    - RSA - "peppol-transport-as4-v2_0"

For backward compatibility reasons, the following property must be kept and must be uncommented:

```
#The AS4 transport profile by which the endpoint is identified in the SMP response
domibus.dynamicdiscovery.transportprofileas4=bdxr-transport-ebms3-as4-v1p0
```

In case the domibus.security.profile.order is not defined, the above defined transport protocol will be used. In case both the domibus.security.profile.order and the domibus.dynamicdiscovery.transportprofileas4 properties are defined, the priority order will be used.

5.2. Domibus Properties

The table below details the properties defined in the property file *edelivery_path***/conf/domibus/domibus.properties** that can be used to configure Domibus.

Note: All the properties which are commented are considered as default values. Domibus takes into account all the commented default values on start up. If you want to modify the default value of a property, then you must uncomment it and change it to the desired value.

Configuration Property	Default value	Purpose
Mandatory configuration start		

Configuration Property	Default value	Purpose
domibus.alert.sender.smtp.url		SMTP server URL for sending alert.
domibus.alert.sender.smtp.port		SMTP server port.
domibus.alert.sender.smtp.user		SMTP server user.
domibus.alert.sender.smtp.password		SMTP server user password.
domibus.alert.receiver.email		Alert email receiver. You can specify multiple recipients by using semicolon separated email addresses: name1@gmail.com;name2@gmail.com.
Mandatory configuration end		
domibus.alert.cleaner.cron	0 0 0/1 * * ?	Cron configuration for cleaning alerts.
domibus.alert.cleaner.alert.lifetime	20	Lifetime in days of alerts before cleaning.
domibus.alert.queue.concurrency	1	Concurrency to process the alerts.
domibus.alert.retry.cron	0 0/10 * * * ?	Frequency of failed alerts retry.
domibus.alert.retry.time	10	Elapsed time in minutes between alert retry.

Configuration Property	Default value	Purpose
domibus.alert.retry.max_attempts	2	Maximum number of attempts for failed alerts.
Message		
domibus.msh.messageid.suffix	domibus.eu	This property is used to generate the random messageID with a fixed suffix which is set by default to "domibus.eu". The resulting format will be UUID@$domibus.msh.messageid.suffix. This property is mandatory.
domibus.message.resend.cron	0 0/1 * * * ?	Messages resend job execution interval as a cron expression.
domibus.message.download.maxSize	10000000	The maximum size of message in bytes that can be downloaded via admin console.
domibus.file.upload.maxSize	52428800	The maximum file size in bytes that can be uploaded through REST (PMode, trustStore). The default is 50MB.
domibus.httpSecurity.httpStrictTransportSecurity.maxAge	31536000	How long(in seconds) HSTS should last in the browser's cache. The default is one year.
domibus.party.finalRecipient.cleanup.cron	0 0 0/1 * * ?	Interval as a cron expression at which to delete the final recipients older than the number of days from the previous property.
domibus.message.test.notification	false	Activates the plugin notification of test messages.
Retry		
domibus.msh.retry.messageExpirationDelay	30000	The retry strategy grants a few extra seconds to avoid not sending the last attempt (value in milliseconds, default 30000).

Configuration Property	Default value	Purpose
domibus.msh.retry.cron	0/30 * * * * ?	It is the retry worker cron job to send the messages. It is set by default to every 5 seconds. This property is mandatory.
domibus.smart.retry.enabled		List of party names (comma separated) for which the smart retry mechanism is enabled. Case sensitive. Example: domibus-red,domibus-blue,domibus-green. The smart retry mechanism prevents message retry to be sent to a party already identified as not reachable by the monitoring feature.
domibus.msh.retry.timeoutDelay	10	Retry strategy adds these extra minutes to the interval used to search back for messages in WAITING_FOR_RETRY status. For performance reasons, the interval defaults to 10 minutes. When there are older messages in WAITING_FOR_RETRY (e.g. restored messages), the interval should be increased to capture those messages as well.
Dynamic Discovery		
domibus.smlzone	acc.edelivery.tech.ec.europa.eu	Set the SMLZone if Domibus needs to be used under Dynamic discovery model. This property is only mandatory if an SML is used.
domibus.dynamicdiscovery.useDynamicDiscovery	false	Whether dynamic discovery is used or not.
domibus.dynamicdiscovery.client.specification	OASIS	The property specifies the dynamic discovery client to be used for the dynamic process. Possible values: OASIS and PEPPOL.

Configuration Property	Default value	Purpose
domibus.dynamicdiscovery.peppolclient.mode	TEST	This information is passed to the PEPPOL client that needs to know whether the usage is for production or testing mode. Possible values: PRODUCTION and TEST.
domibus.dynamicdiscovery.oasisclient.regexCertificateSubjectValidation	^.*EHEALTH_SMP.*$	Apart from validating esponse of signer certificates against the truststore, the Oasis Dynamic Discovery Client gives the possibility to add (optional) a regular expression to validate any certificate metadata related to the subject of the signer certificate. Example: domibus.dynamicdiscovery.oasisclient.regexCertificateSubjectValidation=”^.*$”.or"^.*EHEALTH_SMP.*$"
domibus.dynamicdiscovery.peppolclient.regexCertificateSubjectValidation	.*	Apart from validating the response of the signer certificates against the truststore, the Peppol Dynamic Discovery Client gives the possibility to add (optional) a regular expression to validate the subject of the SMP signer certificate when only the issuer chain is added to the truststore.
domibus.dynamicdiscovery.client.allowedCertificatePolicyOIDs		List of certificate policy OIDs separated by comma. To trust/allow certificate, at least one must be in the service metadata signer’s certificate policy extension (and certificate chain). Empty value disables the certificate policy validation. Example: 1.3.6.1.4.1.7879.13.25
domibus.dynamicdiscovery.peppolclient.partyid.responder.role	urn:fdc:peppol.eu:2017:roles:ap:as4	The role of the responder PartyId may be defined here for PEPPOL.
domibus.dynamicdiscovery.oasisclient.partyid.responder.role	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder	The role of the responder PartyId may be defined here for OASIS.

Configuration Property	Default value	Purpose
domibus.dynamicdiscovery.peppolclient.partyid.type	urn:fdc:peppol.eu:2017:identifiers:ap	The type of the PartyId may be defined here for PEPPOL.
domibus.dynamicdiscovery.oasisclient.partyid.type	urn:oasis:names:tc:ebcore:partyid-type:unregistered	The type of the PartyId may be defined here for OASIS.
domibus.dynamicdiscovery.transportprofileas4	bdxr-transport-ebms3-as4-v1p0	The AS4 transport profile by which the endpoint is identified in the SMP response. In PEPPOL the latest value is peppol-transport-as4-v2_0.
domibus.security.profile.order	ECC, RSA	Priority order of security profiles used in dynamic discovery to set the matching transport protocol.
domibus.dynamicdiscovery.certificate.retention.cron	0 0 23 ? * * *	Retention cron expression for the job responsible for deleting from the truststore the dynamically discovered certificates.
domibus.dynamicdiscovery.certificate.retention.hours	24	#Retention in hours for deleting from the truststore the dynamically discovered certificates. Certificates will be deleted from the truststore if they are not lookup up during the configured value #If the property value is empty, the certificates will not be deleted.
domibus.dynamicdiscovery.lookup.cache.ttl	3600	Global cache property for dynamic discovery lookup caching in seconds.
JMS		
domibus.jms.queue.pull	domibus.internal.pull.queue	Domibus internal queue used for dispatching the pull requests.
domibus.jms.internal.command.concurrency	1-1	Concurrency configured for executing internal commands.
Cluster		
domibus.deployment.clustered	false	If true the quartz scheduler jobs are clustered. This property is mandatory, it should be set to true if the deployment of Domibus is done in a cluster.

Configuration Property	Default value	Purpose
domibus.scheduler.bootstrap.synchronized	true	In cluster mode, specifies if the database synchronization mechanism should be used when initializing the Quartz scheduler
domibus.synchronization.timeout	10000	Timeout for the database synchronization mechanism, in milliseconds
Dispatcher		
domibus.dispatcher.allowChunking	false	Allows chunking when sending messages to other Access Points.
domibus.dispatcher.chunkingThreshold	104857600	If domibus.dispatcher.allowChunking is true, this property sets the threshold at which messages start getting chunked (in bytes). Messages under this limit do not get chunked. Defaults to 100 MB.
domibus.dispatcher.concurency	5-20	Specify concurrency limits via a "lower-upper" String, e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case) when sending messages to other Access Points.
domibus.dispatcher.largeFiles.concurrency	1	Specify concurrency limits via a "lower-upper" String, e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case) when sending large messages (SplitAndJoin) to other Access Points.
domibus.dispatcher.connection.keepAlive	true	Specifies if the connection will be kept alive between C2 and C3. Default value is true.
domibus.dispatcher.connectionTimeout	60000	For connection between the access points – C2 & C3. Specifies the amount of time, in milliseconds, that the consumer will attempt to establish a connection before it times out. 0 is infinite.

Configuration Property	Default value	Purpose
domibus.dispatcher.receiveTimeout	60000	For connection between the access points – C2 & C3. Specifies the amount of time, in milliseconds, that the consumer will wait for a response before it times out. 0 is infinite.
domibus.dispatcher.timeout	300	The timeout of the dispatcher JMS queue transaction in seconds.
domibus.dispatcher.cacheable	true	Cache the dispatcher clients used for communication between the Access Points.
domibus.dispatcher.priority.rule1		Priority rule name. The rule name will be further used to specify additional rule properties.
domibus.dispatcher.priority.rule1.service		Service value to be matched against the sent message.
domibus.dispatcher.priority.rule1.action		List of actions separated by comma to be matched against the sent message.
domibus.dispatcher.priority.rule1.value		Priority value assigned to the JMS message. Accepted priority values must be between 1-9 included.
Pulling		
domibus.msh.pull.cron	0 0 0/1 * * ?	Cron expression used for configuring the message puller scheduling. Format: Sec Min Hour Day Month weekday Year. The example shown is running every hour.
domibus.pull.queue.concurency	1-1	Number of threads used to parallelize the pull requests.
domibus.pull.request.send.per.job.cycle	1	Number of pull requests executed every cron cycle.
domibus.pull.receipt.queue.concurrency	5-20	Number of threads used to parallelize the sending of pull receipts.

Configuration Property	Default value	Purpose
domibus.pull.request.frequency.recovery.time	0	Time in second for the system to recover its full pull capacity when job schedule is one execution per second. If configured to 0, no incremental frequency is executed and the pull pace is executed at its maximum.
domibus.pull.retry.cron	0/10 * * * * ?	Pull Retry Worker execution interval as a cron expression.
domibus.pull.dynamic.initiator	false	Allow dynamic initiator on pull requests - 0 or multiple initiators are allowed in the PMode process. **This property is experimental and should be used only in very specific scenarios.**
domibus.pull.multiple_legs	false	Allow multiple legs configured on the same pull process (with the same security policy).
domibus.pull.force_by_mpc	False	Force message into READY_TO_PULL when mpc attribute is present. **This property is experimental and should be used only in very specific scenarios.**
domibus.pull.mpc_initiator_separator	PID	MPC initiator separator. This is used when the MPC provides information on the initiator: baseMpc/SEPARATOR/partyName. **This property is experimental and should be used only in very specific scenarios.**
Retention		
domibus.retentionWorker.cronExpression	0 0/1 * * * ?	Cron expression used for configuring the retention worker scheduling. The retention worker deletes the expired messages (downloaded and not-downloaded).
domibus.retentionWorker.message.retention.downloaded.max.delete	50	This property is used to tweak the maximum downloaded messages to be deleted by the retention worker.

Configuration Property	Default value	Purpose
domibus.retentionWorker.message.retention.not_downloaded.max.delete	50	This property is used to tweak the maximum not-downloaded messages to be deleted by the retention worker.
Domibus.rentetionWorker.deletion.strategy	DEFAULT	This property defines the message deletion strategy. The possible values are DEFAULT and PARTITIONS. The deletion strategy PARTITIONS is only available on Oracle when partitioning was performed on the tables.
domibus.retention.jms.concurrency	5-10	Specify concurrency limits via a "lower-upper" string, e.g. "5-10", or a simple upper limit string, e.g. "10" (the lower limit will be 1 in this case), when deleting messages. This property is only used when Deletion Strategy is DEFAULT.
domibus.retentionWorker.message.retention.batch.delete	1000	Maximum number of messages to be deleted by the retention worker in a bulk delete (when not specified in the PMode MPC). Default set to 1000, maximum allowed when using Oracle database. This property is only used when the deletion strategy is DEFAULT.
domibus.ongoingMessagesSanitizing.worker.cron	0 0 0/2 ? * * *	Cron expression used for configuring the worker sanitizing messages not in a final state that are not processed anymore. This worker verifies if there are messages in a non-final state.
domibus.ongoingMessagesSanitizing.alert.level	MEDIUM	The priority level of the alerts that report messages in a non-final state that are not processed anymore (LOW, MEDIUM or HIGH).
domibus.ongoingMessagesSanitizing.alert.email.subject	Ongoing messages	Subject of the email alerts that report messages in a non-final state that are not processed anymore.

Configuration Property	Default value	Purpose
domibus.ongoingMessagesSanitizing.alert.email.body	There are messages currently in ongoing statuses that are not processed anymore. Here is the list of IDs and statuses of these messages: [{messages}].	Body of the email alerts that report messages in a non-final state that are not processed anymore containing the string *{messages}* that gets replaced with the list of message IDs and their respective statuses.
Task Executor		
domibus.taskExecutor.threadCount	200	Tomcat only: customize the msh endpoint task executor threads count. Defaults to 200.
domibus.mshTaskExecutor.threadCount	200	Property to customize the msh endpoint task executor threads count. Defaults to 100.
Validation		
domibus.datetime.pattern.onreceiving	yyyy-MM-dd'T'HH:mm:ss[.SSSSSSSSS][.SSSSSS][.SSS][z]	Pattern accepted by Domibus for AS4 connection for the type dateTime. Possible values: # 2020-06-02T20:12:34.5678901 # 2020-06-02T20:12:34.5678901.234 # 2020-06-02T20:12:34.5678901.234567 # 2020-06-02T20:12:34.5678901.234567890 # 2020-06-02T20:12:34.5678901Z # 2020-06-02T20:12:34.5678901.234Z # 2020-06-02T20:12:34.5678901.234567Z # 2020-06-02T20:12:34.5678901.234567890Z
domibus.datetime.pattern.onsending	yyyy-MM-dd'T'HH:mm:ss.SSS'Z'	Pattern accepted by Domibus for AS4 receipts for the type dateTime.
domibus.sendMessage.messageIdPattern	^[\\x20-\\x7E]*$	When an initiator backend client submits messages to Domibus for transmission, with the message id field populated, then the message id should be RFC2822 compliant. The pattern specified here ensures this validation. This field is optional. In case the existing client does not match this message id pattern during submission, then this property can be omitted to skip the validation.

Configuration Property	Default value	Purpose
domibus.receiver.selfsending.validation.active	true	If activated, Domibus will check if the received message is self sent.
domibus.receiver.certificate.validation.onsending	true	If activated Domibus will verify before sending a User Message if the receiver's certificate is valid and not revoked. If the receiver's certificate is not valid or it has been revoked, Domibus will not send the message and it will mark it as SEND_FAILURE.
domibus.sender.certificate.validation.onsending	true	If activated, Domibus will verify before sending a User Message if his own certificate is valid and not revoked. If the certificate is not valid or it has been revoked, Domibus will not send the message and it will mark it as SEND_FAILURE (default is true).
domibus.sender.certificate.validation.onreceiving	true	If activated, Domibus will verify before receiving a User Message if the sender's certificate is valid and not revoked. If the certificate is not valid or it has been revoked, Domibus will not accept the message (default is true).
domibus.sender.trust.validation.onreceiving	true	Enable/disable both the authorization and the validation checks on the sender’s certificate. When set to false, none of the other checks on the sender’s certificate are performed.
domibus.sender.trust.validation.truststore_alias	true	Check that sender’s certificate matches the certificate stored in the truststore. The certificate is loaded from the truststore based on the alias (party name).
domibus.sender.trust.validation.expression		When this property is not empty, Domibus will verify, before receiving a message, if the subject of the sender’s certificate matches the regular expression.

Configuration Property	Default value	Purpose
domibus.sender.trust.validation.allowedCertificatePolicyOIDs		List of certificate policy OIDs separated by comma. When this property is not empty, Domibus will verify before receiving a message that the certificate contains at least one certificate policy OID in certificatePolicy extension.
domibus.sender.certificate.subject.check	false	Check that the subject of the sender's certificate contains the alias (party name). Because this check is very restrictive, it is set by default to false.
JMS		
domibus.listPendingMessages.maxCount	10000 for Tomcat 500 for WildFly and Weblogic	This property specifies the maximum number of messages that would be served when the 'listPendingMessages' operation is invoked. Setting this property is expected to avoid timeouts due to huge result sets being served. A value of 0 would return all the pending messages. This property is optional. Omitting this property would default the result set size to 500. Note: For Tomcat server, the maximum number of shown messages in queue monitoring is defined by the 'domibus.listPendingMessages.maxCount' property.
domibus.jms.queue.maxBrowseSize	10000	The maximum number of messages to be listed from the JMS queues. Setting this property is expected to avoid timeouts due to huge results being served. Setting this property to zero returns all messages.
domibus.jms.queue.alert	domibus.internal.alert.queue	Domibus internal queue used for alerts.

Configuration Property	Default value	Purpose
domibus.jms.internalQueue.expression	.*domibus\.(internal|DLQ|backend\.jms|notification\.jms|notification\.webservice|notification\.kerkovi|notification\.filesystem).*	Regular expression used for identifying the internal queues in the Admin Page.
domibus.jms.connectionFactory.session.cache.size	1	Desired size for the JMS Session cache.
domibus.jms.connectionFactory.maxPoolSize	100	Tomcat only: The max pool size of the JMS connection factory.
Security		
domibus.auth.unsecureLoginAllowed	true	The property specifies if authentication is required or not.
domibus.console.login.maximum.attempt	5	Maximum connection attempts before the account gets locked (suspended).
domibus.console.login.suspension.time	3600	Property defining how many seconds the account remains locked (suspended) before it is automatically unlocked by the system.
domibus.account.unlock.cron	0 0/1 * * * ?	Cron job that determines the interval at which the system checks for account to be reactivated.
domibus.certificate.revocation.offset	15	When a certificate is about to expire, the system will log a warning. The warning will appear as from the expiration date minus the offset in days.
domibus.certificate.check.cron	0 0 0/1 * * ?	Cron expression that specifies the frequence of the certificate revocation check.
domibus.certificate.crl.excludedProtocols		The list of protocols to be excluded from CRL list (possible values: http, https, ftp, file, ldap, etc).
domibus.certificate.crl.http.timeout	10	Configure http timeout (http.connection.timeout, http.socket.timeout, http.connection-manager.timeout) in seconds.
domibus.certificate.crlByUrl.cache.enabled	false	Enable caching of CRLs by URL. Note that, while a CRL is cached, any certificates that were revoked since it was cached would still be accepted.

Configuration Property	Default value	Purpose
domibus.certificate.crlByCert.cache.enabled	true	Enable caching of CRLs by certificate.
domibus.password.encryption.active	false	Domibus encrypts the configured passwords if activated.
domibus.password.encryption.properties	Depends on the server	Enable this property if the password encryption is activated. Add the list of configured passwords to be encrypted.
domibus.password.encryption.key.location	${domibus.config.location}/internal/encrypt	The location where the encrypted key is stored.
domibus.certificate.crlByUrl.cache.enabled	false	Enable caching of CRLs by URL. To enable the property, you should also uncomment the cache in the xml. Note that, while a CRL is cached, any certificates that were revoked since it was cached would still be accepted.
domibus.certificate.crlByCert.cache.enabled	true	Enable caching of CRLs by certificate.
Keystore/Truststore		
domibus.security.keystore.location	${domibus.config.location}/keystores/gateway_keystore.jks	The location of the keystore.
domibus.security.keystore.type	jks	The type of the used keystore.
domibus.security.keystore.password	test123	The password used to load the keystore. Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" are not allowed in domibus.properties file.

Configuration Property	Default value	Purpose
domibus.security.key.private.alias	blue_gw	The alias from the keystore of the private key. Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" must be escaped in domibus.properties file.
domibus.security.key.private.password	test123	The private key password.
domibus.security.truststore.location	${domibus.config.location}/keystores/gateway_truststore.jks	The location of the truststore.
domibus.security.truststore.type	jks	The type of the used truststore.
domibus.security.truststore.password	test123	The password used to load the trustStore. Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" must be escaped in domibus.properties file.
domibus.security.key.private.rsa.alias	blue_gw_rsa	Alias for the RSA security profile, when the same alias is used for signing and decryption.
domibus.security.key.private.rsa.password	test123	Password for the RSA security profile, when the same alias is used for signing and decryption.
domibus.security.key.private.rsa.sign.alias	blue_gw_rsa_sign	Alias for the signing certificate in the RSA security profile.
domibus.security.key.private.rsa.sign.password	test123	Password for the signing certificate in the RSA security profile.
domibus.security.key.private.rsa.decrypt.alias	blue_gw_rsa_decrypt	Alias for the decrypting certificate in the RSA security profile.
domibus.security.key.private.rsa.decrypt.password	test123	Password for the decrypting certificate in the RSA security profile.

Configuration Property	Default value	Purpose
domibus.security.key.private.ecc.alias	blue_gw_ecc	Alias for the ECC security profile when the same alias is used for signing and decryption.
domibus.security.key.private.ecc.password	test123	Password for the ECC security profile when the same alias is used for signing and decryption.
domibus.security.key.private.ecc.sign.alias	blue_gw_ecc_sign	Alias for the signing certificate in the ECC security profile.
domibus.security.key.private.ecc.sign.password	test123	Password for the signing certificate in the ECC security profile.
domibus.security.key.private.ecc.decrypt.alias	blue_gw_ecc_decrypt	Alias for the decrypting certificate in the ECC security profile.
domibus.security.key.private.ecc.decrypt.password	test123	Password for the decrypting certificate in the ECC security profile.
EntityManagerFactory		
domibus.entityManagerFactory.packagesToScan	eu.domibus	Packages to be scanned (comma-separated) by the EntityManagerFactory.
domibus.entityManagerFactory.jpaProperty.hibernate.connection.driver_class		The JDBC driver class used for connecting to the database.
domibus.entityManagerFactory.jpaProperty.hibernate.dialect		This property makes Hibernate generate the appropriate SQL for the chosen database.
domibus.entityManagerFactory.jpaProperty.hibernate.id.new_generator_mappings	false	
domibus.entityManagerFactory.jpaProperty.hibernate.format_sql	true	Pretty print the SQL in the log and console.
domibus.entityManagerFactory.jpaProperty.hibernate.order_inserts	true	
domibus.entityManagerFactory.jpaProperty.hibernate.order_updates	true	
domibus.entityManagerFactory.jpaProperty.hibernate.jdbc.batch_size	30	Batch size for hibernate update and insert. Best practice is to keep this value less than 100.
domibus.entityManagerFactory.jpaProperty.hibernate.connection.provider_disables_autocommit=true		

Configuration Property	Default value	Purpose
domibus.entityManagerFactory.jpaProperty.hibernate.cache.use_query_cache	true	
domibus.entityManagerFactory.jpaProperty.hibernate.cache.use_second_level_cache	true	
domibus.entityManagerFactory.jpaProperty.hibernate.cache.region.factory_class	eu.domibus.core.cache.DomibusCacheRegionFactory	
domibus.entityManagerFactory.jpaProperty.hibernate.cache.generate_statistics	false	
domibus.entityManagerFactory.jpaProperty.hibernate.cache.default_cache_concurrency_strategy	read-only	
domibus.entityManagerFactory.jpaProperty.hibernate.javax.cache.missing_cache_strategy	create	
ActiveMQ		
activeMQ.broker.host	localhost	Tomcat only: the host of the JMS broker.
activeMQ.brokerName	localhost	Tomcat only: the name of the JMS broker.
activeMQ.embedded.configurationFile	file:///${domibus.config.location}/internal/activemq.xml	Tomcat only: the configuration file of the embedded ActiveMQ broker. In case an external broker is used this property is not needed and it should be deleted from the property file.
activeMQ.JMXURL	service:jmx:rmi:///jndi/rmi://${activeMQ.broker.host}:${activeMQ.connectorPort}/jmxrmi	Tomcat only: the service URL of the MBeanServer.
activeMQ.connectorPort	1199	Tomcat only: the port that the JMX connector will use for connecting to ActiveMQ.
activeMQ.transportConnector.uri	tcp://${activeMQ.broker.host}:61616	Tomcat only: the connection URI that the clients can use to connect to an ActiveMQ broker using a TCP socket.
activeMQ.username	domibus	Tomcat only: the username that is allowed to connect to the ActiveMQ broker.
activeMQ.password	changeit	Tomcat only: the password of the username defined in the **activeMQ.username** property. It is recommended to change the password value.

Configuration Property	Default value	Purpose
activeMQ.persistent	true	Tomcat only: the persistence enabled flag.
activeMQ.connection.closeTimeout	15000	Tomcat only: the timeout before a close is considered complete
activeMQ.connection.connectResponseTimeout	0	Tomcat only: the connection response timeout
Database		
domibus.datasource.maxPoolSize	10	Tomcat only: sets the maximum pool size. The amount of pooled connections, including both idle and in-use connections, will not go above this value.
domibus.database.serverName	localhost	Tomcat only: the host name or the IP address of the database server.
domibus.database.port	3306	Tomcat only: the port number of the database server.
domibus.database.schema	domibus_schema	The Domibus database schema.
domibus.datasource.driverClassName	com.mysql.jdbc.cj.Driver	Tomcat only: the JDBC driver class name.
domibus.datasource.url	jdbc:mysql://localhost:3306/domibus_schema?useSSL=false &useLegacyDatetimeCode=false&serverTimezone=UTC	Tomcat only: the JDBC URL connection.
domibus.datasource.user	edelivery_user	Tomcat only: a user who has access to the Domibus database schema.
domibus.datasource.password	edelivery_password	Tomcat only: the password of the user defined in the **domibus.datasource.user** property.
domibus.database.general.schema	general_schema	Multitenancy only: schema used by Domibus to configure the association of users to domains, the super users and other things that are not related to a specific domain. This property is mandatory for Multitenancy mode.
Plugin User Security		

Configuration Property	Default value	Purpose
domibus.plugin.login.maximum.attempt	5	Plugin user security property: number of console login attempts before the user is deactivated (default 5).
domibus.plugin.login.suspension.time	3600	Plugin user security property: time in second for a suspended plugin user to be reactivated. (1 hour per default if property is not set, if 0 the user will not be reactivated).
domibus.plugin.account.unlock.cron	0 0/1 * * * ?	Plugin user security property: cron job that determines the interval at which the system checks for plugin account to be reactivated.
GUI		
domibus.UI.title.name	Domibus	Property where you can specify the title in the Tab of Admin Console.
domibus.ui.pages.messageLogs.countLimit	50000	The limit when calculating the number of message logs (disabled when 0).
domibus.ui.pages.messageLogs.interval.default	24	The initial/default interval (in hours) for filtering messages
domain_name.domibus.ui.pages.messageLogs	true	Defines whether the Messages page has all the advanced capabilities (defaults to true, should be set to false in high load environments).
domibus.ui.csv.rows.max	10000	Max rows for CSV export
domibus.ui.support.team.name	EDELIVERY Support Team	Support team name
domibus.ui.support.team.email	CEF-EDELIVERY-SUPPORT@ec.europa.eu	Support team email
domibus.ui.resend.action.enabled.received.minutes	5	How many minutes after message's received date the Resend button will become enabled for messages having SEND_ENQUEUED status
domibus.ui.session.secure	false	Used to secure the session of admin console; set to true only in conjunction with https protocol

Configuration Property	Default value	Purpose
domibus.ui.session.timeout	30	Used to set the the session timeout of admin console (in minutes)
domibus.ui.session.sameSite	Strict	Used to set the SameSite attribute of cookies (session and all others) used by the admin console. Possible values are: Strict, None, Lax
Proxy Settings		In case your Access Point has to use a proxy server you can configure it with these properties.
domibus.proxy.enabled	false	Values “true”/”false”, depending on whether you need to use proxy or not.
domibus.proxy.http.host	-	Host name of the proxy server.
domibus.proxy.http.port	-	Port of Proxy server.
domibus.proxy.user	-	Username for authentication on the proxy server.
domibus.proxy.password	-	Password for authentication on the proxy server.
domibus.proxy.nonProxyHosts	-	Indicates the hosts that should be accessed without going through the proxy.
Alert management		
domibus.alert.active	true	Enable/disable the entire alert module.
domibus.alert.mail.sending.active	false	Allow to disable alert mail sending.
domibus.alert.mail.smtp.timeout	5000	SMTP Socket I/O timeout value in milliseconds.
domibus.alert.msg.communication_failure.active	true	Enable/disable the messaging alert module.
domibus.alert.msg.communication_failure.states	SEND_FAILURE	Message status change that should be notified by the messaging alert module. Comma-separated.

Configuration Property	Default value	Purpose
domibus.alert.msg.communication_failure.level	HIGH	Alert levels corresponding to message status defined in domibus.alert.msg.communication_failure.states. Should be: HIGH, MEDIUM OR LOW.
domibus.alert.msg.communication_failure.mail.subject	Message status change	Messaging alert module mail subject.
domibus.alert.user.login_failure.active	true	Enable/disable the login failure alert of the authentication module.
domibus.alert.user.login_failure.level	LOW	Alert level for login failure.
domibus.alert.user.login_failure.mail.subject	Login failure	Login failure mail subject.
domibus.alert.user.account_disabled.active	true	Enable/disable the account disable alert of the authentication module.
domibus.alert.user.account_disabled.level	HIGH	Alert level for account disabled.
domibus.alert.user.account_disabled.moment	WHEN_BLOCKED	Time when the account disabled alert should be triggered. 2 possible values: AT_LOGON: an alert will be triggered each time a user tries to login to a disabled account. WHEN_BLOCKED: an alert will be triggered once when the account got disabled.
domibus.alert.user.account_disabled.subject	Account disabled	Account disabled mail subject.
domibus.alert.cert.imminent_expiration.active	true	Enable/disable the imminent certificate expiration alert of certificate scanner module.
domibus.alert.cert.imminent_expiration.delay_days	60	Number of days before revocation as from when the system should start sending alerts.
domibus.alert.cert.imminent_expiration.frequency_days	14	Frequency in days between alerts.
domibus.alert.cert.imminent_expiration.level	HIGH	Certificate imminent expiration alert level.
domibus.alert.cert.imminent_expiration.mail.subject	Certificate imminent expiration	Certificate imminent expiration mail subject.

Configuration Property	Default value	Purpose
domibus.alert.cert.expired.active	true	Enable/disable the certificate expired alert of certificate scanner module.
domibus.alert.cert.expired.frequency_days	7	Frequency in days between alerts.
domibus.alert.cert.expired.duration_days	90	Number of days after the revocation when the system should trigger alerts for the expired certificate.
domibus.alert.cert.expired.level	HIGH	Certificate expired alert level.
domibus.alert.cert.expired.mail.subject	Certificate expired	Certificate expired mail subject.
domibus.alert.partition.expiration.frequency_days	1	Frequency in days between alerts sent when attempting to delete a partition that contains messages not in final state.
domibus.alert.connection.monitoring.parties	ALL	Comma separated list of parties for whom to create alerts.
domibus.alert.connection.monitoring.frequency_days	1	Connection monitoring failed alert frequency in days.
domibus.alert.connection.monitoring.level	MEDIUM	Connection monitoring failed alert level.
domibus.alert.connection.monitoring.mail.subject	Connection monitoring failed	Connection monitoring failed mail subject.
Alert management for Plugin Password policy		
domibus.alert.plugin_password.imminent_expiration.active	true	Enable/disable the imminent password expiration alert.
domibus.alert.plugin_password.imminent_expiration.delay_days	15	Number of days before expiration as for how long before expiration the system should send alerts.
domibus.alert.plugin_password.imminent_expiration.frequency_days	3	Frequency in days between alerts.
domibus.alert.plugin_password.imminent_expiration.level	LOW	Password imminent expiration alert level.
domibus.alert.plugin_password.imminent_expiration.mail.subject	Password imminent expiration	Password imminent expiration mail subject.

Configuration Property	Default value	Purpose
domibus.alert.plugin_password.expired.active	true	Enable/disable the imminent password expiration alert.
domibus.alert.plugin_password.expired.delay_days	30	Number of days after expiration as for how long the system should send alerts.
domibus.alert.plugin_password.expired.frequency_days	5	Frequency in days between alerts.
domibus.alert.plugin_password.expired.level	LOW	Password expiration alert level.
domibus.alert.plugin_password.expired.mail.subject	Password expired	Password expiration mail subject.
Alert management:authentication module for plugin users		Properties for configuring alerts for plugin user authentication.
domibus.alert.plugin.user.login_failure.active	true	Enable/disable the login failure alert of the authentication module.
domibus.alert.plugin.user.login_failure.level	LOW	Alert level for login failure.
domibus.alert.plugin.user.login_failure.mail.subject	Login failure	Login failure mail subject.
domibus.alert.plugin.user.account_disabled.active	true	Enable/disable the account disable alert of the authentication module.
domibus.alert.plugin.user.account_disabled.level	HIGH	Alert level for account disabled.
domibus.alert.plugin.user.account_disabled.moment	WHEN_BLOCKED	Time when the account disabled alert should be triggered: 2 possible values: AT_LOGON: an alert will be triggered each time a user tries to login to a disabled account. WHEN_BLOCKED: an alert will be triggered once when the account got disabled.
domibus.alert.plugin.user.account_disabled.subject	Account disabled	Account disabled mail subject.
SplitAndJoin		

Configuration Property	Default value	Purpose
domibus.attachment.temp.storage.location		SplitAndJoin only: Domibus uses a file system location for storing temporary data when processing SplitAndJoin messages. In a cluster configuration the temporary file system storage needs to be accessible by all the nodes from the cluster.
domibus.dispatcher.splitAndJoin.concurrency	1	SplitAndJoin only: specify concurrency limits via a "lower-upper" String, e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case) when sending the SourceMessage receipt (Split and Join) to other Access Points.
domibus.dispatcher.splitAndJoin.payloads.schedule.threshold	1000	SplitAndJoin only: The threshold value in MB to switch from synchronous to asynchronous saving of outgoing SourceMessage payloads.
domibus.splitAndJoin.receive.expiration.cron	0 0/5 * * * ?	SplitAndJoin only: Cron expression that specifies the frequency of the checking if the joinInterval has expired.
Metrics		Properties related to Metrics configuration.
domibus.metrics.jmx.reporter.enable	false	Enable jmx reporter for dropwizard metrics. It is not recommended to gather metrics via JMX. However, it can be helpful for development and browsing purposes.
domibus.metrics.slf4j.reporter.enable	false	Enable slf4j reporter for dropwizard metrics.
domibus.metrics.slf4j.reporter.period.time.unit	MINUTES	The time unit used to configure the frequency of writing statistics into the statistic.log file. Possible values are: SECONDS, MINUTES, HOURS.

Configuration Property	Default value	Purpose
domibus.metrics.slf4j.reporter.period.number	1	The number of period of the previously time unit used to configure the frequency of writing statistics into the statistic.log file. E.g. the default configuration will write statistics with the file every 1 MINUTE.
domibus.metrics.monitor.memory	false	Activate dropwizard memory metrics.
domibus.metrics.monitor.gc	false	Activate dropwizard GC metrics.
domibus.metrics.monitor.cached.threads	false	Activate dropwizard cached threads metrics.
domibus.metrics.monitor.jms.queues	false	Activate dropwizard JMS Queues metrics.
domibus.metrics.monitor.jms.queues.refresh.period	10	Amount of time (in seconds) the JMS count will be cached. If 0, the count is not cached.
domibus.metrics.monitor.jms.queues.show.dlq.only	true	Add metrics for only for DLQ queue count only.
Password Policy		
domibus.passwordPolicy.pattern	^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[~`!@#$%^&+=\\-_<>.,?:;*/()\|\\[\\]{}'"\\\\]).{8,32}$	Password minimum complexity rules (empty to disable password complexity enforcement).
domibus.passwordPolicy.validationMessage	Minimum length: 16 characters;Maximum length: 32 characters;At least one letter in lowercase;At least one letter in uppercase;At least one digit;At least one special character	Password validation message in case it does not meet the rules in **domibus.passwordPolicy.pattern**.
domibus.passwordPolicy.expiration	90	Password expiration policy in days (0 to disable).
domibus.passwordPolicy.defaultPasswordExpiration	15	Default password expiration policy in days (0 to disable).
domibus.passwordPolicy.warning.beforeExpiration	15	Password expiration policy: number of days before expiration when the system warns users at login.

Configuration Property	Default value	Purpose
domibus.passwordPolicy.dontReuseLast	5	Password reuse policy: do not reuse any of the last N passwords (0 to disable).
domibus.passwordPolicy.checkDefaultPassword	true	Default password validation policy enabled/disabled (by default is enabled).
domibus.passwordPolicy.defaultUser.autogeneratePassword	true	Default user password generation policy enabled/disabled (by default is enabled)
domibus.passwordPolicies.check.cron	0 0 0/1 * * ?	Cron expression that specifies the frequency of the password expiration check.
Plugin Users Password Policy		
domibus.plugin.passwordPolicy.pattern	^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[~`!@#$%^&+=\\-_<>.,?:;*/()\|\\[\\]{}'"\\\\]).{16,32}$	Password minimum complexity rules (empty to disable password complexity enforcement).
domibus.plugin.passwordPolicy.validationMessage	Minimum length: 16 characters; Maximum length: 32 characters; At least one letter in lowercase; At least one letter in uppercase; At least one digit; At least one special character	Password validation message in case it does not meet the rules stated in **domibus.plugin.passwordPolicy.pattern**.
domibus.plugin.passwordPolicy.expiration	90	Password expiration policy in days (0 to disable).
domibus.plugin.passwordPolicy.defaultPasswordExpiration	1	Default password expiration policy in days (0 to disable).
domibus.plugin.passwordPolicy.dontReuseLast	5	Password reuse policy: do not reuse any of the last N passwords (0 to disable).
Payload		
domibus.payload.encryption.active	false	Domibus encrypts the payloads stored in the database or file system if this property is active.
domibus.payload.temp.job.retention.exclude.regex	.*ehcache-sizeof-agent.*	Temporary files are excluded from deletion if this regular expression matches the file name.
domibus.payload.temp.job.retention.directories	domibus.attachment.temp.storage.location	List of directories to check for cleaning the temporary files.
domibus.payload.temp.job.retention.cron	0 0/10 * * * ?	Cron expression that specifies the frequency of checking if the temporary payloads have expired.

Configuration Property	Default value	Purpose
domibus.payload.temp.job.retention.expiration	120	The threshold in minutes for considering the temporary payloads as expired. The expired temporary payloads are scheduled to be deleted.
domibus.attachment.storage.location	-	It is possible to configure Domibus to save the message payloads on the file system instead of the database. This setting is recommended when exchanging payloads bigger than 30MB. In order to enable the file system storage please add the following property: domibus.attachment.storage.location= *your_file_system_location* where *your_file_system_location* is the location on the file system where the payloads will be saved. Remark: In a cluster configuration the file system storage needs to be accessible by all the nodes from the cluster. your_file_system_location. (User should provide absolute path of 'your_file_system_location' . Relative path of payload storage is forbidden in domibus.) Please note that \\ \' and \" must be escaped in domibus.properties file. So in 'your_file_system_location' please use '/ '.

Configuration Property	Default value	Purpose
domibus.payload.limit.28attachments.per.message	true	Limit attachments per message to 28 (count enforced by Apache Santuario library for extended XML signature validation. Ref: https://santuario.apache.org/faq.html#faq-4.SecureValidation)
domibus.payload.decompression.validation.active	false	When set to true, Domibus tries to decompress the archived payloads on receiving a message. In case it fails to decompress one payload, an error receipt is returned.
DSS		
domibus.dss.ssl.trust.store.path	${domibus.config.location}/keystores/dss-tls-truststore.p12	TLS truststore for DSS dataloader.
domibus.dss.ssl.trust.store.password	dss-tls	TLS truststore password for DSS dataloader.
domibus.dss.ssl.trust.store.type	PKCS12	TLS truststore type DSS dataloader.
domibus.dss.ssl.cacert.path		Override cacert truststore path if needed.
domibus.dss.ssl.cacert.type	JKS	Cacert truststore type.
domibus.dss.ssl.cacert.password	changeit	Cacert truststore password. It is recommended to change the password value.
domibus.dss.perform.crl.check	false	Perform CRL check within DSS. It is perfomed by Domibus.
Connection monitoring		
domibus.monitoring.connection.cron	0 0 0/2 ? * * *	Cron expression that specifies the frequency of test messages sent to monitor the C2-C3 connections.
domibus.monitoring.connection.self.cron	0 0 0/1 ? * * *	Cron expression that specifies the frequency of test messages sent to itself (e.g. C2-C2 connections)
domibus.monitoring.connection.party.enabled		Specifies the parties for which to monitor the connection (comma-separated list).

Configuration Property	Default value	Purpose
domibus.monitoring.connection.party.history.delete	ALL	Specifies the parties for which to delete the old test messages (comma separated list)
domibus.monitoring.connection.messages.received.history.delete.cron	0 0 0/1 ? * * *	Cron expression that specifies the frequency of deleting test message history sent to gateway party
Extensions		
domibus.extension.iam.authentication.identifier	DEFAULT_AUTHENTICATION_SPI	Name of the authentication extension used to verify the chain trust. Default is CXF.
domibus.extension.iam.authorization.identifier	DEFAULT_AUTHORIZATION_SPI	Name of the authorization extension used to check incoming message authorization. Default is truststore check.
Various		
messageFactoryClass	com.sun.xml.messaging.saaj.soap.ver1_2.SOAPMessageFactory1_2Impl	The factory for creating SOAPMessage objects.
domibus.javax.xml.validation.SchemaFactory	com.sun.org.apache.xerces.internal.jaxp.validation.XMLSchemaFactory	The factory for creating SchemaFactory objects.
domibus.jmx.user	jmsManager	WebLogic specific: the user that will be used to access the queues via JMX.
domibus.jmx.password	jms_Manager1	WebLogic specific: the associated password of the configured domibus.jmx.user.
domibus.plugin.notification.active	true	If disabled, Domibus will not notify the plugins when the state of the User Message changes. Defaults to true.
domibus.nonrepudiation.audit.active	true	If disabled, Domibus will not save the non-repudiation audit data. Defaults to true.
domibus.dispatch.ebms.error.unrecoverable.retry	true	This property should be set to true if Domibus needs to retry sending the failed messages. This property is mandatory
domibus.userInput.blackList	'\u0022(){}[];,+=%&*#<>/\\	Characters that are not accepted for user input in admin console.
domibus.internal.queue.concurency	3-10	Number of threads used to parallelize the dispatching of messages to the plugins.

Configuration Property	Default value	Purpose
domibus.logging.ebms3.error.print	true	Prints the raw XML response in the logs in case of EBMS3 error on receiver/sender side (if eu.domibus is put at least on ERROR).
domibus.logging.payload.print	false	Prints the AS4 payload in the logs while org.apache.cxf is set to at least INFO in logback.xml. Defaults to false.
domibus.logging.metadata.print	true	Prints the AS4 metadata in the logs when org.apache.cxf is set to at least INFO in logback.xml. Defaults to true.
domibus.logging.cxf.limit	18000	The size limit at which messages are truncated in the logs when org.apache.cxf is set to at least INFO in logback.xml Number between 0 and 1000000000 bytes. Default to limit is 6000 bytes.
domibus.connection.cxf.ssl.offload.enable	false	Enables offloading the SSL connection to another application (e.g. SSL Forward Proxy).
domibus.logging.remote.certificates.print	false	Prints the details of the remote receiver certificate used to encrypt the message when sending or the details of the remote sender certificate used to verify the trust of the message when receiving.
domibus.logging.local.certificates.print	false	Prints the details of the local sender certificate corresponding to its private key being used to sign the message when sending or the details of the local receiver certificate corresponding to its private key when decrypting the message when receiving. Please note, when enabling this property, there is an extra delay introduced needed to load the certificate corresponding to the private key.
domibus.property.length.max	10000	The maximum length accepted for a property value, in bytes. Defaults to 10000.

Configuration Property	Default value	Purpose
domibus.userInput.whiteList	^[\\w\\-\\.: @]*$	Characters that are accepted in user input
domibus.internal.queue.concurency	3-10	Number of threads used to parallelize the dispatching of messages to the plugins.
domibus.property.validation.enabled	true	Enables the validation of domibus properties values (default to true).
domibus.property.backup.period.min	24	The minimum interval of time after which a new backup file is created, in hours. Defaults to 24. 0 for creating a backup file on every change.
domibus.property.backup.history.max	10	The maximum number of backup files to keep. Defaults to 10. 0 for keeping all of them.
domibus.partyinfo.roles.validation.enabled	true	Validate PartyInfo From/To initiator and responder roles. This property helps maintaining backwards compatibility. It is recommended to be enabled. By default enabled.
domibus.pmode.legconfiguration.mpc.validation.enabled	true	Validate the qualified name of the Mpc in the received UserMessage on MSH endpoint matches the qualified name of the Mpc configured on the pMode leg configuration.
domibus.cacerts.validation.enabled	true	When enabled, the system default certificates (cacerts) are used in addition to custom certificates, to validate that a call to a https server can be trusted.
domibus.instance.name	Domibus	Domibus instance/environment name.
domibus.sendMessage.failure.delete.payload	false	Whether to delete the message payload or send failure. Defaults to false (the admin could put the message back in the send queue).

Configuration Property	Default value	Purpose
domibus.sendMessage.success.delete.payload	true	Whether to delete the message payload on send success. Default set to true (to keep backwards compatibility).
domibus.sendMessage.attempt.audit.active	true	If disabled, Domibus will not save the message attempt details when there is a failure sending a message. Defaults to true.
compressionBlacklist	application/vnd.etsi.asic-s+zip,image/jpeg	The list of mime-types that will not be compressed (in outgoing messages) even if compression is turned on for the given message.
eArchive		
domibus.earchive.active	false	Enable earchive.
domibus.earchive.export.empty	false	Allows to create empty export batches if no messages are found. If false, no batch or files are created when no messages are found.
domibus.earchive.queue.concurrency	1-1	Earchive queue concurrency.
domibus.earchive.notification.queue.concurrency	1-1	Earchive notification queue concurrency.
domibus.earchive.notification.dlq.concurrency	1-1	Earchive dead letter queue (after notification) concurrency.
domibus.earchive.cron	0 0 0/1 * * ?	Cron configuration for executing the earchiving continuous process.
domibus.earchive.sanitizer.cron	0 0 0/1 * * ?	Cron configuration for executing the earchiving sanity process.
domibus.earchive.sanitizer.messagesCheck.delay.hours	24	Number of hours to be excluded from running time (at least) from the earchive sanitizer run
domibus.earchive.retention.cron	0 0 0/2 * * ?	Cron configuration for executing the earchiving cleanup process.
domibus.earchive.batch.size	5000	Maximum messages to be archived in one batch.

Configuration Property	Default value	Purpose
domibus.earchive.batch.max	10	Maximum earchive batches to be created during one job.
domibus.earchive.batch.retry.timeout	0	Timeout used to retrieve the messages in minutes. WARNING: this value is rounded to the previous whole hour to limit the messages search scope (f.i.: runtime = 15h12 - if retry.timeout=5 -> latest time for a message to be taken into account will be 15h00 and earlier - if retry.timeout=30 -> latest time for a message to be taken into account will be 14h00 and earlier) "-1" disables this functionality and retrieve the timeout with the loaded PMode.
domibus.earchive.rest.messages.return	false	For methods: history of exports and enqueued batches, also return message list as part of batches objects. Note: in case of large batch size, returning message list in all of the batches in a list will slow down the response time of the services.
domibus.earchive.notification.timeout	5000	Timeout used when notifying the eArchiving client.
domibus.earchive.notification.useProxy	false	Specifies whether to use a proxy when notifying the e-archiving client.
domibus.earchive.retention.days	30	If a batch is not archived during this time, it is considered as expired.
domibus.earchive.retention.delete.max	5000	Maximum number of earchive batches to delete at a time.
domibus.earchive.sanitizer.messagesCheck.delay.hours	24	The creation date of the last UserMessage processed by the earchiving continuous batch minus the ['messageCheck.delay' hour] will be latest userMessage taken for the sanitizer.

Configuration Property	Default value	Purpose
domibus.earchive.start_date.stopped.allowed_hours	24	An alert is sent if the start date of the continuous job was not updated between “now” and “now minus the allowed_window (in hours)”.
domibus.alert.earchive.notification.active	true	Enable/disable the earchiving notification failed alerts.
domibus.alert.earchive.notification.level	MEDIUM	Earchive notification failed alert level.
domibus.alert.earchive.notification.mail.subject	E-Archiving client notification failed	Earchive notification failed mail subject.
domibus.alert.earchive.messages_non_final.active	true	Enable/disable the earchive non final message alert.
domibus.alert.earchive.messages_non_final.level	HIGH	Alert level for earchive non final message.
domibus.alert.earchive.messages_non_final.mail.subject	Earchive: message not in final state	Earchive non final message mail subject.
domibus.alert.earchive.start_date_stopped.active	true	Enable/disable the earchive start date stopped alert.
domibus.alert.earchive.start_date_stopped.level	HIGH	Alert level for earchive start date stopped.
domibus.alert.earchive.start_date_stopped.mail.subject	Earchive: continuous job start date stopped	Earchive start date stopped mail subject.
domibus.earchive.sanitizer.messagesCheck.delay.hours	24	The creation date of the last UserMessage processed by the earchiving continuous batch minus the **messageCheck.delay** hour will be latest userMessage taken for the sanitizer
domibus.alert.earchive.export.failed.active	true	Enable/disable the Earchive message export failed alert.
domibus.alert.earchive.export.failed.level	HIGH	Alert level for Earchive message export failed alert.
domibus.alert.earchive.export.failed.mail.subject	Earchive: sanitizer export message failed	Earchive message export failed alert mail subject.
Error logs		

Configuration Property	Default value	Purpose
domibus.errorlog.cleaner.cron	0 0 0/1 * * ?	Cron configuration for cleaning error logs without message ids.
domibus.errorlog.cleaner.older.days	100	Cron job will delete error logs without message ids older than this property days
domibus.errorlog.cleaner.batch.size	5000	How maximum error logs will be deleted for each job run
Distributed Cache		
domibus.cache.distributed.size	5	Maximum used heap size in megabytes
domibus.cache.distributed.ttl	3600	The maximum number of seconds for each entry to stay in the cache. Entries that are older than timeToLiveSeconds will be automatically evicted from the cache. Updates on the entry will change the eviction time. Any integer between 0 and Integer.MAX_VALUE. 0 means infinite. Default is 1h.
domibus.cache.distributed.idle.max	3600	Maximum number of seconds for each entry to stay idle in the cache. Entries that are idle (not touched) for more than maxIdleSeconds will get automatically evicted from the cache. Entry is touched if get(), getAll(), put() or containsKey() is called. Any integer between 0 and Integer.MAX_VALUE. 0 means infinite. Default is 3600. The time precision is limited by 1 second. The MaxIdle that less than 1 second can lead to unexpected behaviour.
domibus.cache.distributed.nearcache.size	5000	The near cache default size for the distributed cache
domibus.cache.distributed.nearcache.ttl	3600	The maximum number of seconds for each entry to stay in the near cache

Configuration Property	Default value	Purpose
domibus.cache.distributed.nearcache.idle.max	3600	Set the maximum number of seconds each entry can stay in the Near Cache as untouched (not-read). Entries that are not read (touched) more than maxIdleSeconds value will get removed from the Near Cache.
domibus.cache.distributed.port	6701	Sets the port the distributed cache member will try to bind on. A valid port value is between 0 and 65535. A port number of 0 will let the system pick up an ephemeral port.
domibus.cache.distributed.port.autoincrement	true	Sets if a distributed cache member is allowed to find a free port by incrementing the port number when it encounters an occupied port. If you explicitly want to control the port a distributed cache member is going to use, set portAutoincrement to false. In this case, the distributed cache member is going to try the port and if the port is not free, the member will not start and throw an exception. If this value is set to true, the distributed cache will start at the port specified and will try until it finds a free port, or until it runs out of ports to try
domibus.cache.distributed.port.count	5	The maximum number of ports allowed to use in case autoincrement is set to true
domibus.cache.distributed.members	127.0.0.1	Sets the well-known members of the distributed cache. Can be a comma separated string, e.g. '10.11.12.1,10.11.12.2' which indicates multiple members are going to be added.
domibus.cache.location	${java.io.tmpdir}/cache	Disk cache location, used in the ehCache configuration by all caches with a disk storage tier

Configuration Property	Default value	Purpose

Table 1 - Domibus Properties

The properties that are specific to super users (ROLE_AP_ADMIN) are defined in a separate file called **super-domibus.properties,** a file that can be found along with the others. These properties are related to password policy and alert configuration for super users (for more information see also $19.6 -Users):

Configuration Property	Default value	Purpose
Super.domibus.alert.cleaner.cron	0 0 0/1 * * ?	Cron configuration for cleaning super user alerts.
Super.domibus.alert.cleaner.alert.lifetime	20	Lifetime in days of super user alerts.
Super.domibus.alert.active	true	Enable/disable the super user alert module.
Super.domibus.alert.mail.sending.active	false	Enable/disable the super user alert mail sending.
Super.domibus.alert.retry.cron	0 0/1 * * * ?	Frequency of failed super user alert retry.
Super.domibus.alert.retry.time	1	Elapsed time in minutes between super user alert retry.
Super.domibus.alert.retry.max_attempts	2	Maximum number of attempts for failed super user alert
Super.domibus.alert.user.login_failure.active	true	Enable/disable the login failure super user alert of the authentication module.
Super.domibus.alert.user.login_failure.level	LOW	Super user alert level for login failure.
Super.domibus.alert.user.login_failure.mail.subject	Super user login failure	Super user login failure alert mail subject.
Super.domibus.alert.user.account_disabled.active	true	Enable/disable the account disabled super user alert of the authentication module.
Super.domibus.alert.user.account_disabled.level	HIGH	Super user alert level for account disabled.

Configuration Property	Default value	Purpose
Super.domibus.alert.user.account_disabled.moment	WHEN_BLOCKED	Time when the account disabled super user alert should be triggered. 2 possible values: AT_LOGON: an alert will be triggered each time a user tries to login to a disabled account. WHEN_BLOCKED: an alert will be triggered once when the account got disabled.
Super.domibus.alert.user.account_disabled.subject	Super user account disabled	Super user account disabled alert mail subject.

Table 2 – Super-domibus Properties

5.2.1. Password encryption

Passwords configured in *domibus.properties* are stored by default in clear text. The Domibus configuration file, *domibus.properties*, is not accessible for third-party users. Nevertheless, it is a good practice to encrypt the configured passwords to increase the security level.

Domibus encrypts the configured passwords using symmetric encryption with *AES/GCM/NoPadding* algorithm. In order to activate the password encryption, please set the property *domibus.password.encryption.active=true* and uncomment the domibus.password.encryption.properties to enable the list of configured passwords to be encrypted. Once activated, all the passwords configured under the property *domibus.password.encryption.properties* will be encrypted.

Domibus generates the symmetric key the first time the password encryption is activated. The generated symmetric key is stored in the file *encrypted.key*, in the location specified by the property *domibus.password.encryption.key.location*.

For instance, the property *domibus.security.keystore.password=test123* will be encrypted to *domibus.security.keystore.password=ENC(4DTXnc9zUuYqB0P/q7RtRHpG9VJLs3E=).*

6. PLUGIN MANAGEMENT

This section describes the different types of plugins and their registration process.

6.1. Default Plugins

Domibus comes with three default plugins. The three Interface Control Documents (ICD) describe these three plugins (JMS, WS and File System Plugin) (cf.[REF6]).

6.1.1. JMS Plugin

For the JMS plugin, you will have to use the following resources (see section § 3.1-"*Binaries repository*" for the download location):

- **domibus-msh-distribution-X.Y.Z-default-jms-plugin.zip**

6.1.2. WS Plugin

For the WS plugin, you will have to use the following resources (see section §3.1-"*Binaries repository*" for the download location):

- **domibus-msh-distribution-X.Y.Z-default-ws-plugin.zip**

6.1.3. File System Plugin

For the File System plugin, you will have to use the following resources (see section §3.1-"*Binaries repository*" for the download location):

- **domibus-msh-distribution-X.Y.Z-default-fs-plugin.zip**

6.2. Custom Plugin

Users can develop their own plugins. Please refer to the plugin cookbook for more details (cf.[REF6]).

6.2.1. Plugin registration

***Remark**:*

> *Please refer to section* 10.3 "Message Log" *for the routing of the specific plugin after registering the plugin on your specific Application Server.*

6.2.1.1. Tomcat

In order to install a custom plugin for Tomcat, please follow the steps below**:**

1. Stop Tomcat server
2. Copy the custom plugin jar file to the plugins folder *CATALINA_HOME***/conf/domibus/plugins/lib**

3. Copy the custom plugin XML configuration files under the Tomcat subfolder **directly** to *CATALINA_HOME***/conf/domibus/plugins/config.** . There should not be any Tomcat folder under *DOMAIN_HOME***/conf/domibus/plugins/config**
4. Start Tomcat server

Remark:

CATALINA_HOME is the folder where the Tomcat is installed.

6.2.1.2. *WebLogic*

In order to install a custom plugin for WebLogic please follow the steps below**:**

1. Stop the WebLogic server
2. Copy the custom plugin jar file to the plugins folder *DOMAIN_HOME***/conf/domibus/plugins/lib**
3. Copy the custom plugin XML configuration files under the WebLogic subfolder **directly** to *DOMAIN_HOME***/conf/domibus/plugins/config** folder. There should not be any WebLogic folder under *DOMAIN_HOME***/conf/domibus/plugins/config**
4. Start the WebLogic server

Remark:

DOMAIN_HOME is the folder corresponding to the WebLogic domain.

6.2.1.3. WildFly

In order to install a custom plugin please follow the steps below**:**

1. Stop the WildFly server
2. Copy the custom plugin jar file to the plugins folder *edelivery_path* **/conf/domibus/plugins/lib**
3. Copy the custom plugin XML configuration files under the WildFly subfolder **directly** to *edelivery_path* **/conf/domibus/plugins/config.** There should not be a WildFly folder under *DOMAIN_HOME***/conf/domibus/plugins/config**
4. Start the WildFly server

6.3. Plugin authentication

The plugins authentication is disabled by default for the default plugins. In order to enable the plugin authentication for the default plugins in Domibus the following steps must be followed:

1. Set the property "**domibus.auth.unsecureLoginAllowed**" to false in **domibus.properties**:

```
domibus.auth.unsecureLoginAllowed=false
```

2. Configure the application server to allow http(s) requests and pass the authentication credentials to Domibus.

6.4. Plugin notifications

Domibus core notifies the plugins on different events. The types of events are the following:

MESSAGE_RECEIVED, MESSAGE_FRAGMENT_RECEIVED, MESSAGE_SEND_FAILURE, MESSAGE_FRAGMENT_SEND_FAILURE, MESSAGE_RECEIVED_FAILURE, MESSAGE_FRAGMENT_RECEIVED_FAILURE, MESSAGE_SEND_SUCCESS, MESSAGE_FRAGMENT_SEND_SUCCESS, MESSAGE_STATUS_CHANGE, MESSAGE_FRAGMENT_STATUS_CHANGE

It is possible to specify in the properties configuration file of each Domibus default plugin the list of events received.

More details can be found in the Plugin Cookbok (cf.[REF6]) and the Interface Control Document (ICD) of each plugin (WS, FS and JMS) (cf.[REF6]).

6.5. Plugin Ehcache files

Pluging are able to use their own Ehcache files for defining caches:

- Default Classpath file (into the .jar of the plugin) – must be of name *-plugin-default-ehcache.xml and must be situated in a folder named /ehcache

- External file which overrides the caches defined in the default file: must have the name *plugin-ehcache.xml and should be situated in the Domibus configuration folder at location/plugins/config along with the other configuration files for plugins

More details could be found in the Plugin Cookbok (cf.[REF6]).

7. PMODE CONFIGURATION

Processing Modes (PModes) are used to configure Access Points. The PMode parameters are loaded into the Access Point via an XML file.

The features described in the PMode file are: Security, Reliability, Transport, Business Collaborations, Error Reporting, Message Exchange Patterns (MEPs) and Message Partition Channels (MPCs).

As different messages may be subject to various types of processing or, as different business domains may have several requirements, Access Points commonly support several PModes. Some PMode parameters are mandatory, others are optional. For more information, please refer to the Access Point Component Offering Document.

7.1. Configuration

In Domibus, PModes are XML files that you can create or edit. You can configure the two files given: *edelivery_path**/conf/pmodes/domibus-gw-sample-pmode-**party_id_name1**.xml* and *edelivery_path**/conf/pmodes/domibus-gw-sample-pmode-**party_id_name2**.xml.*

The "*party_id_name1" value must be replaced* with your own party name and *the "party_id_name2"* with your corresponding party name.

The party_id must match the alias of the certificate in the keystore and the endpoint must be the external access link to your instance.

***Remark**:*

This step could be managed by a PMode Configuration Manager, known to your Business Owner.

```
<party name="party_id_name2"
          endpoint="http:// party_id_name2_hostname:8080/domibus/services/msh">
   <identifier partyId="party_id_name2_1"
                    partyIdType="partyTypeUrn"/>
</party>
<party name="party_id_name1"
          endpoint="http:// party_id_name1_hostname:8080/domibus/services/msh">
   <identifier partyId="party_id_name1_1" partyIdType="partyTypeUrn"/>
</party>
```

7.1.1. *Adding a new participant*

If a new participant Access Point is joining your network, you need to configure your PMode accordingly and re-upload it as mentioned in §7.1.4 – *"Upload new Configuration"*.

- Add a "new_party" element:

```
<party name="new_party_name"
                        endpoint="http://new_party_msh" >
        <identifier partyId="new_party_id" partyIdType="partyTypeUrn"/>
</party>
```

- Add your "new_party_name" as initiator:

The party with the role of initiator will be the sender of the messages:

```
<initiatorParties>
                ...
                <initiatorParty name="new_party_name"/>
        </initiatorParties>
```

- Add your "new_party_name" as responder:

The party with the role of responder will be the receiver of the messages:

```
<responderParties>
                ...
                <responderParty name="new_party_name"/>
        </responderParties>
```

7.1.2. Sample PMode file

Processing modes (PModes) describe how messages are exchanged between AS4 partners (in this case *Access Points blue_gw and red_gw*). These files contain the identifiers of each AS4 Access Point (identified as *parties* in the PMode file below).

Sender and Receiver Identifiers represent the organizations that send and receive the business documents. They are both used in the authorization process (PMode). Therefore, adding, modifying or deleting a participant implies modifying the corresponding PMode files.

An an example of a PMode XML file is shown below:

***Remark**:*

In this setup, we have allowed each party (blue_gw or red_gw) to initiate the process. If only blue_gw is supposed to send messages, then put only blue_gw in <initiatorParties> and red_gw in <responderParties>.

```
<?xml version="1.0" encoding="UTF-8"?>
<db:configuration xmlns:db="http://domibus.eu/configuration" party="blue_gw">

        <mpcs>
                <mpc name="defaultMpc"
                        qualifiedName="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultMPC"
                        enabled="true"
                        default="true"
                        retention_downloaded="0"
                        retention_undownloaded="14400"/>
                        retention_sent="14400"
                        delete_message_metadata="false"
                        max_batch_delete="1000"/>
```

```
        </mpcs>
        <businessProcesses>
                <roles>
                        <role name="defaultInitiatorRole"
                                        value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/initiator"/>
                        <role name="defaultResponderRole"
                                        value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/responder"/>
                </roles>
                <parties>
                        <partyIdTypes>
                                <partyIdType name="partyTypeUrn"
value="urn:oasis:names:tc:ebcore:partyid-type:unregistered"/>
                        </partyIdTypes>
                        <party name="red_gw"
                                   endpoint="http://<red_hostname>:8080/domibus/services/msh">
                                <identifier partyId="domibus-red" partyIdType="partyTypeUrn"/>
                        </party>
                        <party name="blue_gw"

        endpoint="http://<blue_hostname>:8080/domibus/services/msh">
                                <identifier partyId="domibus-blue" partyIdType="partyTypeUrn"/>
                        </party>
                </parties>
                <meps>
                        <mep name="oneway" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/oneWay"/>
                        <mep name="twoway" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/twoWay"/>
                        <binding name="push" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/push"/>
                        <binding name="pull" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/pull"/>
                        <binding name="pushAndPush" value=http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/pushAndPush"/>
                </meps>
                <properties>
                        <property name="originalSenderProperty"
                                        key="originalSender"
                                        datatype="string"
                                        required="true"/>
                        <property name="finalRecipientProperty"
                                        key="finalRecipient"
                                        datatype="string"
                                        required="true"/>
                        <propertySet name="eDeliveryPropertySet">
                                <propertyRef property="finalRecipientProperty"/>
                                <propertyRef property="originalSenderProperty"/>
                        </propertySet>
                </properties>
                <payloadProfiles>
                        <payload name="businessContentPayload"
```

```
                                        cid="cid:message"
                                        required="true"
                                        mimeType="text/xml"/>
                        <payload name="businessContentAttachment"
                                        cid="cid:attachment"
                                        required="false"
                                        mimeType="application/octet-stream"/>
                        <payloadProfile name="MessageProfile" maxSize="2147483647">
                                <attachment name="businessContentPayload"/>
                                <attachment name="businessContentAttachment"/>
                        </payloadProfile>
                </payloadProfiles>
                <securities>
                        <security name="eDeliveryAS4Policy"
                                        policy="eDeliveryAS4Policy.xml"
                                        signatureMethod="RSA_SHA256" />
                </securities>
                <errorHandlings>
                        <errorHandling name="demoErrorHandling"
                                        errorAsResponse="true"
                                        businessErrorNotifyProducer="true"
                                        businessErrorNotifyConsumer="true"
                                        deliveryFailureNotifyProducer="true"/>
                </errorHandlings>
                <agreements>
                        <agreement name="agreement1" value="A1" type="T1"/>
                </agreements>
                <services>
                        <service name="testService1" value="bdx:noprocess" type="tc1"/>
                        <service name="testService" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/service"/>
                </services>
                <actions>
                        <action name="tc1Action" value="TC1Leg1"/>
                        <action name="testAction" value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/test"/>
                </actions>
                <as4>
                        <receptionAwareness name="receptionAwareness" retry="12;4;CONSTANT"
duplicateDetection="true"/>
                        <reliability name="AS4Reliability" nonRepudiation="true"
replyPattern="response"/>
                </as4>
                <legConfigurations>
                        <legConfiguration name="pushTestcase1tc1Action"
                                        service="testService1"
                                        action="tc1Action"
                                        defaultMpc="defaultMpc"
                                        reliability="AS4Reliability"
                                        security="eDeliveryAS4Policy"
                                        receptionAwareness="receptionAwareness"
                                        propertySet="eDeliveryPropertySet"
                                        payloadProfile="MessageProfile"
```

```
                                        errorHandling="demoErrorHandling"
                                        compressPayloads="true"/>
                        <legConfiguration name="testServiceCase"
                                                                service="testService"
                                                                action="testAction"
                                                                defaultMpc="defaultMpc"
                                                                reliability="AS4Reliability"
                                                                security="eDeliveryAS4Policy"

receptionAwareness="receptionAwareness"
                                                                propertySet="eDeliveryPropertySet"
                                                                payloadProfile="MessageProfile"
                                                                errorHandling="demoErrorHandling"
                                                                compressPayloads="true"/>
                </legConfigurations>
<process name="tc1Process"
      mep="oneway"
      binding="push"
      initiatorRole="defaultInitiatorRole"
      responderRole="defaultResponderRole">
         <initiatorParties>
                <initiatorParty name="blue_gw"/>
                <initiatorParty name="red_gw"/>
         </initiatorParties>
         <responderParties>
                <responderParty name="blue_gw"/>
                <responderParty name="red_gw"/>
         </responderParties>
         <legs>
                <leg name="pushTestcase1tc1Action"/>
                <leg name="testServiceCase"/>
         </legs>
</process>
         </businessProcesses>
</db:configuration>
```

7.1.3. Domibus PMode configuration to ebMS3 PMode Mapping

The following table provides additional information about the Domibus PMode configuration files.

Domibus PMode Configuration	EbMS3 Specification [ebMS3CORE] [AS4-Profile]	Description
MPCs	-	Container which defines the different MPCs (Message Partition Channels).
MPC	PMode[1].BusinessInfo.MPC: The value of this parameter is the identifier of the MPC (Message Partition Channel) to which the message is assigned. It maps to the attribute **Messaging** / **UserMessage**	Message Partition Channel allows the partition of the flow of messages from a *Sending MSH* to a *Receiving MSH* into several flows, each of which is controlled separately. An MPC also allows merging flows from several *Sending MSHs* into a unique flow that will be treated as such by a *Receiving MSH*. The value of this parameter is the identifier of the MPC to which the message is assigned.
MessageRetentionDownloaded	-	Retention interval for messages already delivered to the backend.
MessageRetentionUnDownloaded	-	Retention interval for messages not yet delivered to the backend.
MessageRetentionSent		Retention interval for messages already sent with success or failed to the other MSH.
DeleteMessageMetadata		When true, message metadata is deleted, not only the payload. The interval between the two is configured by MetadataRetentionOffset.
MaxBatch		Sets the maximum batch to be used when deleting messages in bulk. When there are multiple expired messages, they will be deleted in batches until all consumed.
MetadataRetentionOffset		When DeleteMessageMetadata is also true, the message metadata is deleted later than the payload.
Parties	-	Container which defines the different PartyIdTypes, Party and Endpoint.
PartyIdTypes	maps to the attribute **Messaging/UserMessage/ PartyInfo**	Message Unit bundling happens when the Messaging element contains multiple child elements or Units (either User Message Units or Signal Message Units).

Party ID	maps to the element **Messaging/UserMessage/ PartyInfo**	The ebCore Party ID type can simply be used as an identifier format and therefore as a convention for values to be used in configuration and – as such – does not require any specific solution building block.
Endpoint	maps to **PMode[1].Protocol.Address**	The endpoint is a party attribute that contains the link to the MSH. The value of this parameter represents the address (endpoint URL) of the *Receiver MSH* (or *Receiver Party*) to which Messages under this PMode leg are to be sent. Note that a URL generally determines the transport protocol (e.g. if the endpoint is an email address, then the transport protocol must be SMTP; if the address scheme is "http", then the transport protocol must be HTTP).
AS4	-	Container.
Reliability [@Nonrepudiation] [@ReplyPattern]	Nonrepudiation maps to **PMode[1].Security.SendReceipt.NonRepudiation** ReplyPattern maps to **PMode[1].Security.SendReceipt.ReplyPattern**	PMode[1].Security.SendReceipt.NonRepudiation: value = 'true' (to be used for non-repudiation of receipt), value = 'false' (to be used simply for reception awareness). PMode[1].Security.SendReceipt.ReplyPattern: value = 'Response' (sending receipts on the HTTP response or back-channel). PMode[1].Security.SendReceipt.ReplyPattern: value = 'Callback' (sending receipts use a separate connection).
ReceptionAwareness [@retryTimeout] [@retryCount] [@strategy] [@duplicateDetection]	retryTimeout maps to **PMode[1].ReceptionAwareness.Retry=true** PMode[1].ReceptionAwareness.Retry.Parameters retryCount maps to **PMode[1].ReceptionAwareness.Retry.Parameters** strategy maps to **PMode[1].ReceptionAwareness.Retry.Parameters** duplicateDetection maps to **PMode[1].ReceptionAwareness.DuplicateDetection**	These parameters are stored in a composite string. • *retryTimeout* defines timeout in minutes. • *retryCount* is the total number of retries. • *strategy* defines the frequency of retries. The only *strategy* available as of now is *CONSTANT*. • *duplicateDetection* allows to check duplicates when receiving twice the same message. The only *duplicateDetection* available as of now is *TRUE*.

ReceptionAwareness [@retryTimeout] [@initialInterval] [@multiplyingFactor] [@strategy] [@duplicateDetection]	retryTimeout maps to **PMode[1].ReceptionAwareness.Retry=true** PMode[1].ReceptionAwareness.Retry.Parameters initialInterval maps to **PMode[1].ReceptionAwareness.Retry.Parameters** multiplyingFactor maps to **PMode[1].ReceptionAwareness.Retry.Parameters** strategy maps to **PMode[1].ReceptionAwareness.Retry.Parameters** duplicateDetection maps to **PMode[1].ReceptionAwareness.DuplicateDetection**	These parameters are used only by the PROGRESSIVE strategy. • *retryTimeout* defines timeout in minutes. • *initialInterval* is the initial retry interval in minutes, after the first failed attempt. • *multiplyingFactor* is the factor of multiplication for the series of intervals (e.g. initialInterval=1, multiplyingFactor=2 results in the following retry series: 1, 2, 4, 8, … until retryTimeout is reached. • *strategy* defines the frequency of retries. The available strategies as of now are PROGRESSIVE and *CONSTANT*. • *duplicateDetection* allows to check duplicates when receiving twice the same message. The only *duplicateDetection* available as of now is *TRUE*.
Securities	-	Container.
Security	-	Container.
Policy	PMode[1].Security.* NOT including PMode[1].Security.X509.Signature.Algorithm	The parameter defines the name of a WS-SecurityPolicy file.
SignatureMethod	PMode[1].Security.X509.Signature.Algorithm	This parameter is not supported by WS-SecurityPolicy and therefore it is defined separately.
BusinessProcessConfiguration	-	Container.
Agreements	maps to eb:Messaging/ UserMessage/ CollaborationInfo/ AgreementRef	This OPTIONAL element occurs zero times or once. The *AgreementRef* element is a string that identifies the entity or artifact governing the exchange of messages between the parties.
Actions	-	Container.
Action	maps to **Messaging/ UserMessage/ CollaborationInfo/Action**	This REQUIRED element occurs once. The element is a string identifying an operation or an activity within a Service that may support several of these.
Services	-	Container.
ServiceTypes Type	maps to **Messaging/ UserMessage/ CollaborationInfo/ Service[@type]**	This REQUIRED element occurs once. It is a string identifying the service that acts on the message and it is specified by the designer of the service.

MEP [@Legs]	-	An ebMS MEP defines a typical choreography of ebMS User Messages which are all related through the use of the referencing feature (RefToMessageId). Each message of an MEP Access Point refers to a previous message of the same Access Point, unless it is the first one to occur. Messages are associated with a label (e.g. *request*, *reply*) that precisely identifies their direction between the parties involved and their role in the choreography.
Bindings	-	Container.
Binding	-	The previous definition of ebMS MEP is quite abstract and ignores any binding consideration to the transport protocol. This is intentional, so that application level MEPs can be mapped to ebMS MEPs independently from the transport protocol to be used.
Roles	-	Container.
Role	Maps to **PMode.Initiator.Role or PMode.Responder.Role** depending on where this is used. In ebMS3 message this defines the content of the following element: • For Initiator: **Messaging/UserMessage/PartyInfo/From/Role** • For Responder: **Messaging/UserMessage/PartyInfo/To/Role**	The required role element occurs once, and identifies the authorized role (*fromAuthorizedRole* or *toAuthorizedRole*) of the Party sending the message (when present as a child of the *From* element), or receiving the message (when present as a child of the *To* element). The value of the role element is a non-empty string, with a default value of *http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultRole* Other possible values are subject to partner agreement.
Processes	-	Container.
PayloadProfiles	-	Container.
Payloads	-	Container.

Payload	maps to **PMode[1].BusinessInfo.PayloadProfile**	This parameter allows specifying some constraint or profile on the payload. It specifies a list of payload parts. A payload part is a data structure that consists of five properties: 1. **name** (or Content-ID) that is the **part identifier**, and can be used as an index in the notation PayloadProfile; 2. **MIME data type** (text/xml, application/pdf, etc.); 3. **name of the applicable XML Schema file** if the MIME data type is text/xml; 4. **maximum size in kilobytes;** 5. **Boolean** string indicating whether the part is **expected** or **optional**, within the User message. The message payload(s) must match this profile.
ErrorHandlings	-	Container.
ErrorHandling	-	Container.
ErrorAsResponse	maps to **PMode[1].ErrorHandling.Report.AsResponse**	This Boolean parameter indicates (if *true*) that errors generated from receiving a message in error are sent over the back-channel of the underlying protocol associated with the message in error. If *false*, such errors are not sent over the back-channel.
ProcessErrorNotifyProducer	maps to **PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer**	This Boolean parameter indicates whether (if *true*) the Producer (application/party) of a User Message matching this PMode should be notified when an error occurs in the Sending MSH, during processing of the *User Message to be sent*.
ProcessErrorNotifyConsumer	maps to **PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer**	This Boolean parameter indicates whether (if *true*) the Consumer (application/party) of a User Message matching this PMode should be notified when an error occurs in the Receiving MSH, during processing of the *received User message*.

DeliveryFailureNotifyProducer	maps to **PMode[1].ErrorHandling.Report.DeliveryFailuresNotifyProducer**	When sending a message with this reliability requirement (*Submit* invocation), one of the two following outcomes shall occur: - The Receiving MSH successfully delivers (*Deliver* invocation) the message to the Consumer. - The Sending MSH notifies (*Notify* invocation) the Producer of a delivery failure.
Legs	-	Container.
Leg	-	Because messages in the same MEP may be subject to different requirements - e.g. the reliability, security and error reporting of a response may not be the same as for a request – the PMode will be divided into *legs*. Each user message label in an ebMS MEP is associated with a PMode leg. Each PMode leg has a full set of parameters for the six categories above (except for *General Parameters*), even though in many cases parameters will have the same value across the MEP legs. Signal messages that implement transport channel bindings (such as PullRequest) are also controlled by the same categories of parameters, except for *BusinessInfo group*.
Process	-	In *Process* everything is plugged together.

Table 3 - Domibus PMode configuration to ebMS3 mapping

7.1.4. Upload new Configuration

7.1.4.1. Upload the PMode file

Remark:

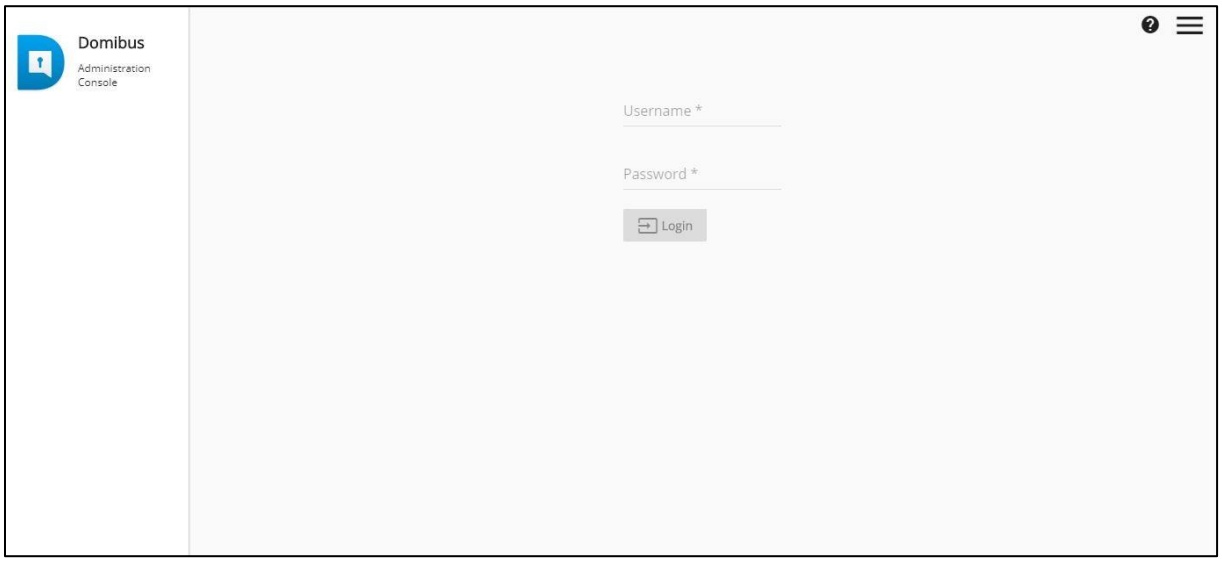
In case of a cluster environment, the PMode configuration is replicated automatically on all the nodes.

1. To update the PMode configuration and/or Truststore, connect to the Administration Console using the administrator's credentials (by default: User = ***admin***; for the password, look in the logs for the phrase: “Default password for user admin is”) to http://*localhost*:*8080*/domibus.

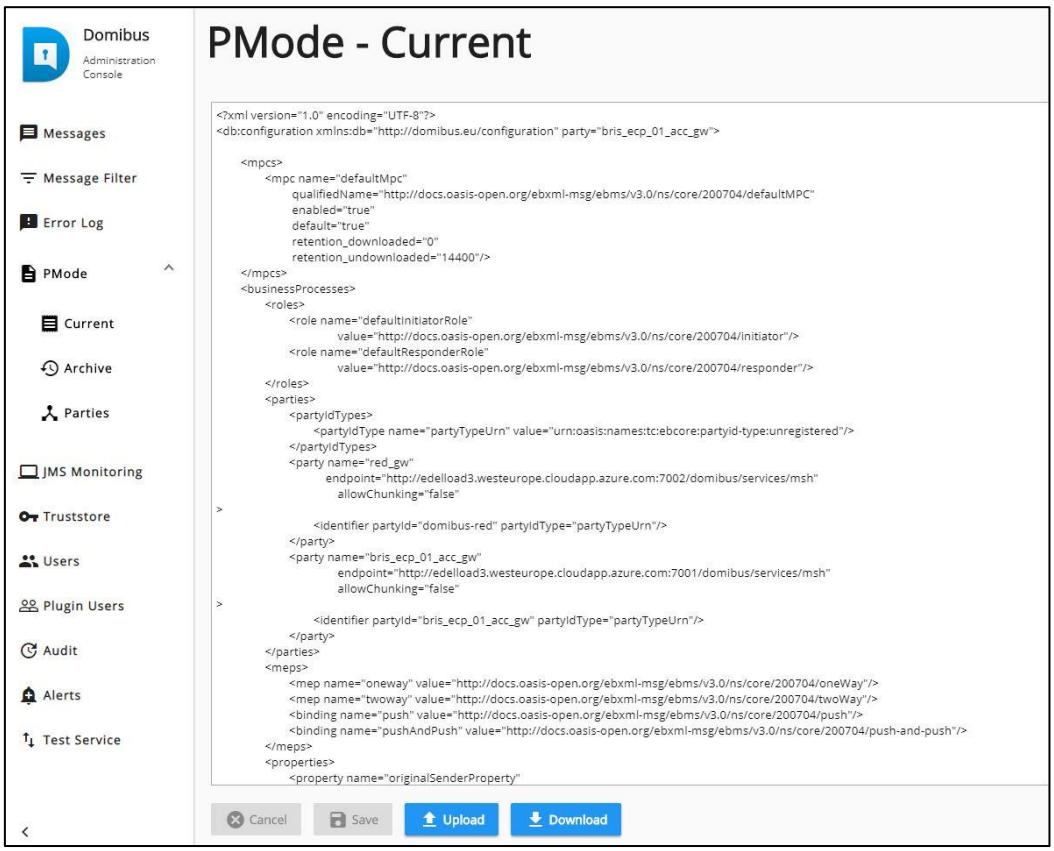
Remark*:*

Duplicate parameters/entities are not allowed in PMode. XSD validation is used to find the duplicate entities.

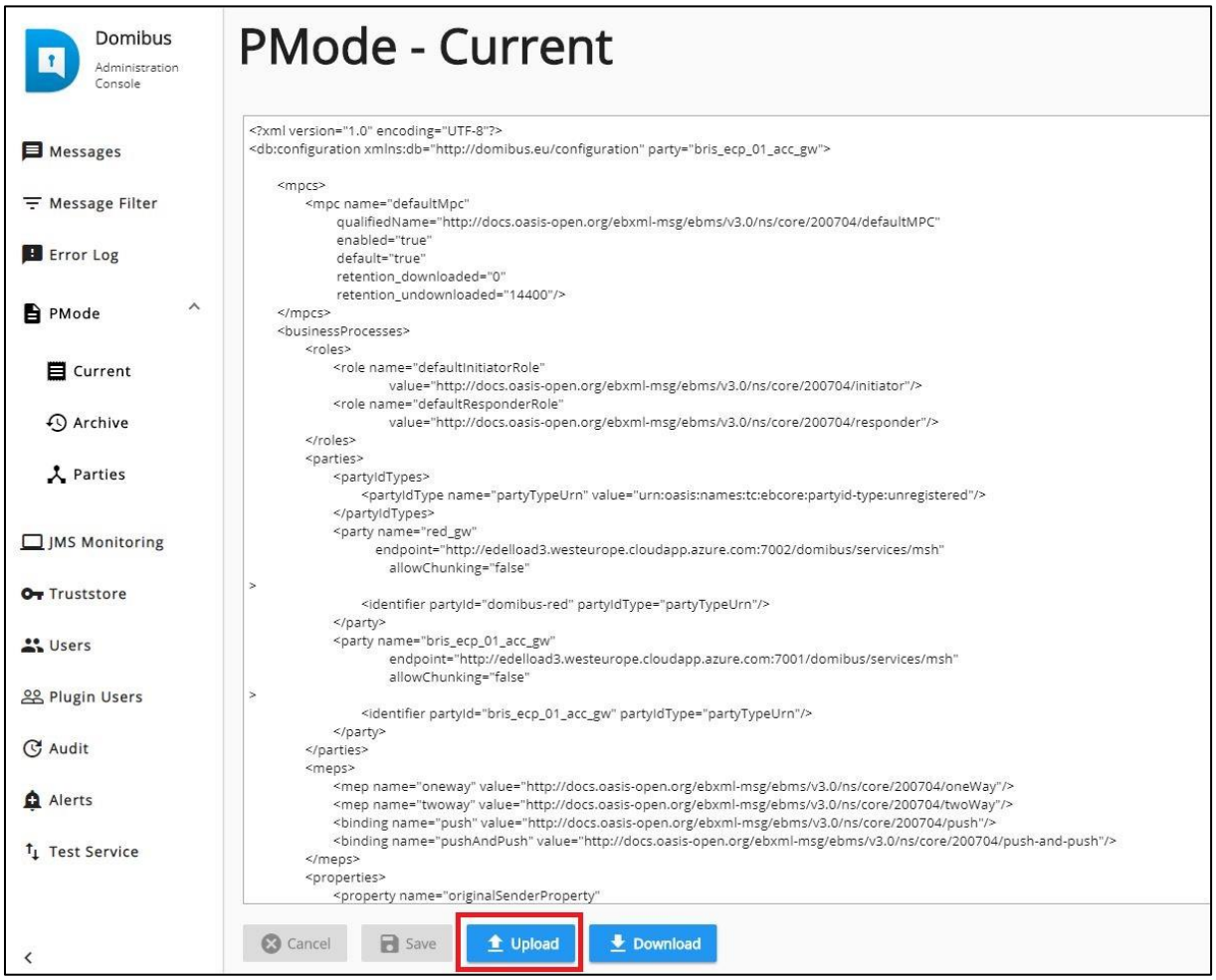
It is recommended to change the passwords for the default users.
See §10.1 – "Administration " for further information.



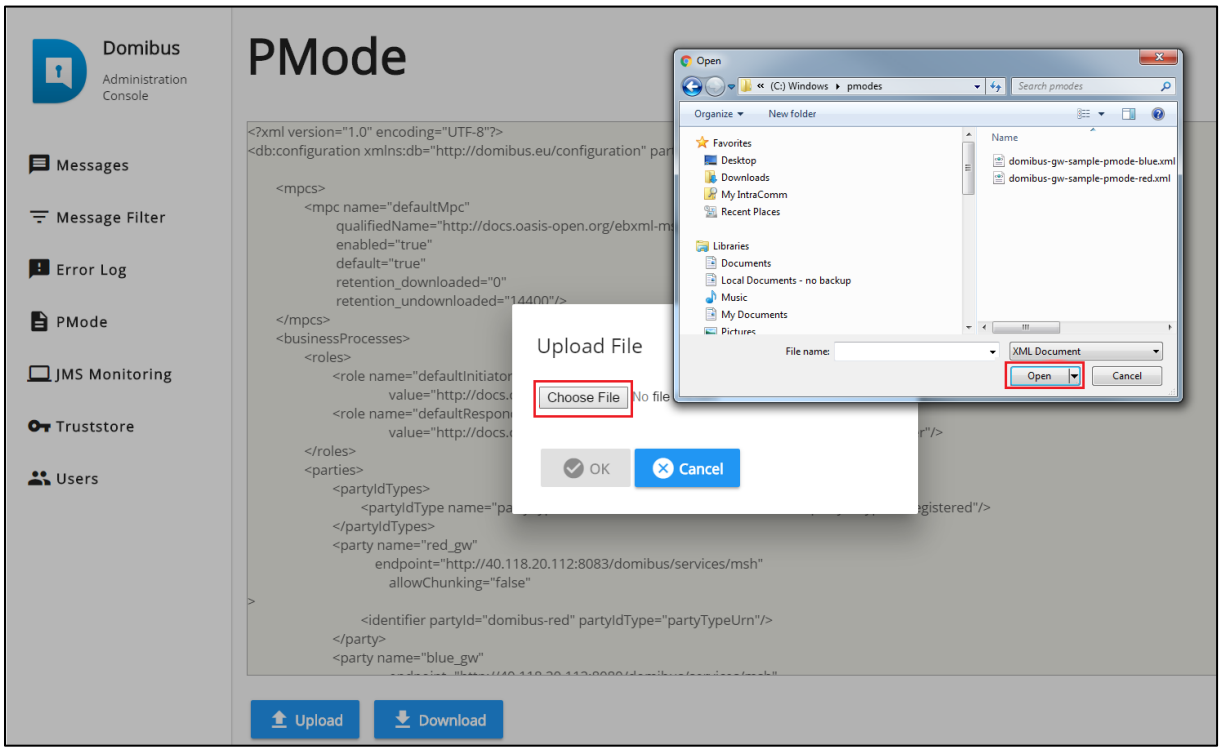
2. Click on the **PMode menu**:



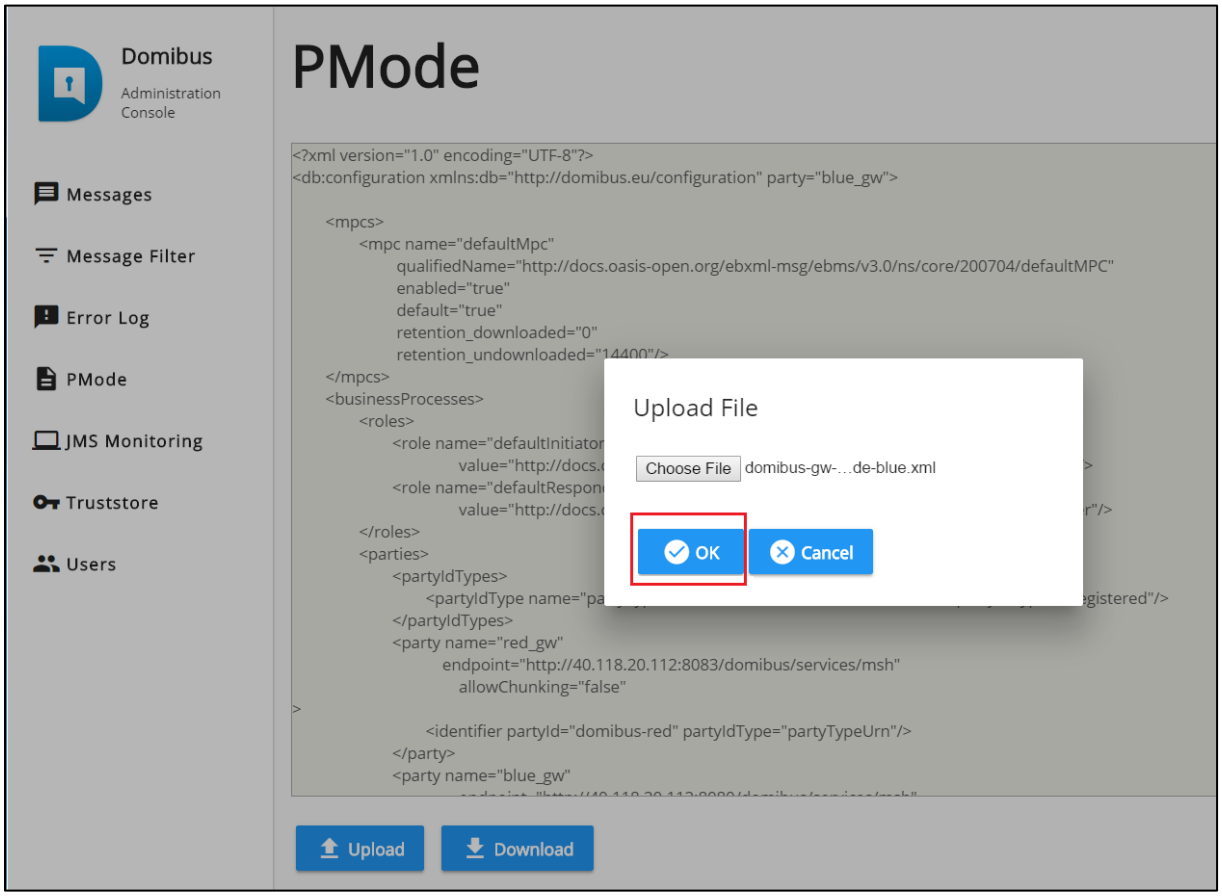
3. Press the **Upload** button:



4. Press the **Choose File** button, and navigate to the PMode file, select it and click on the **Open** button (or equivalent) in the standard dialog box:



5. Once the file has been selected, click "**OK**" to upload the PMode xml file:

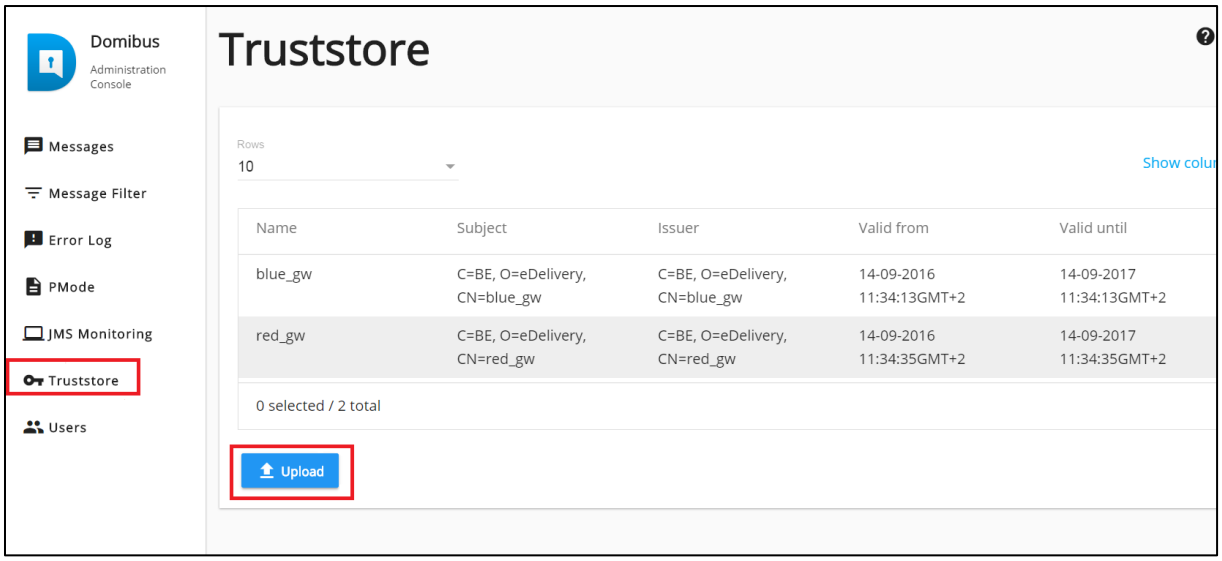


Remark:

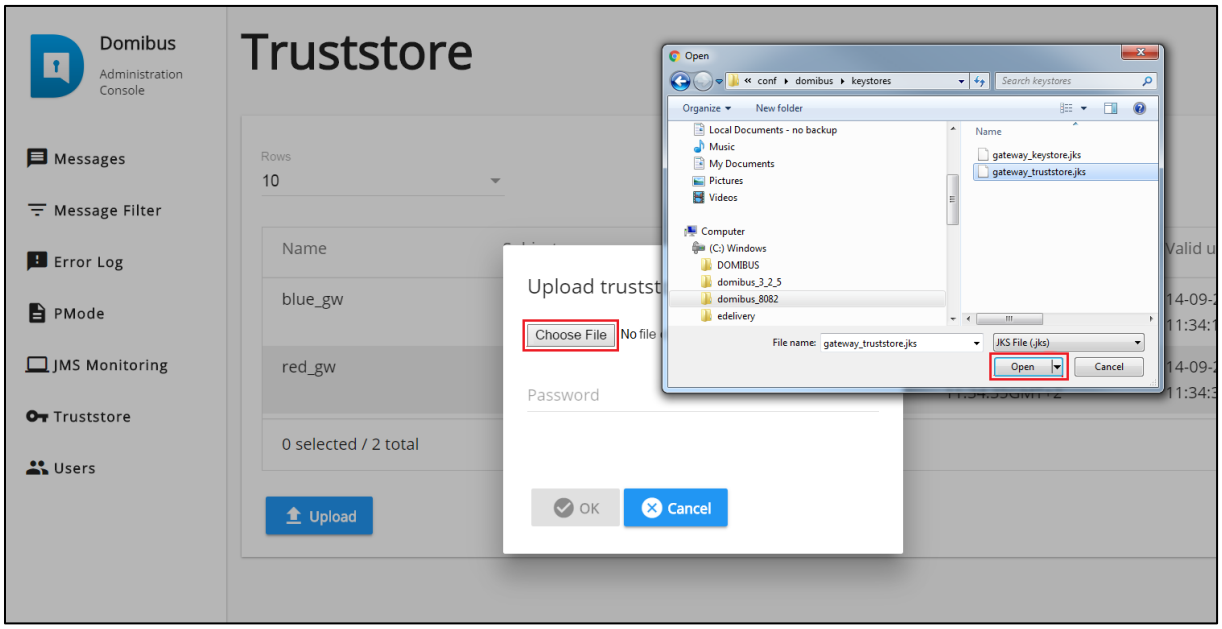
Every time a PMode is updated, the truststore is also reloaded.

7.1.4.2. Upload the Truststore

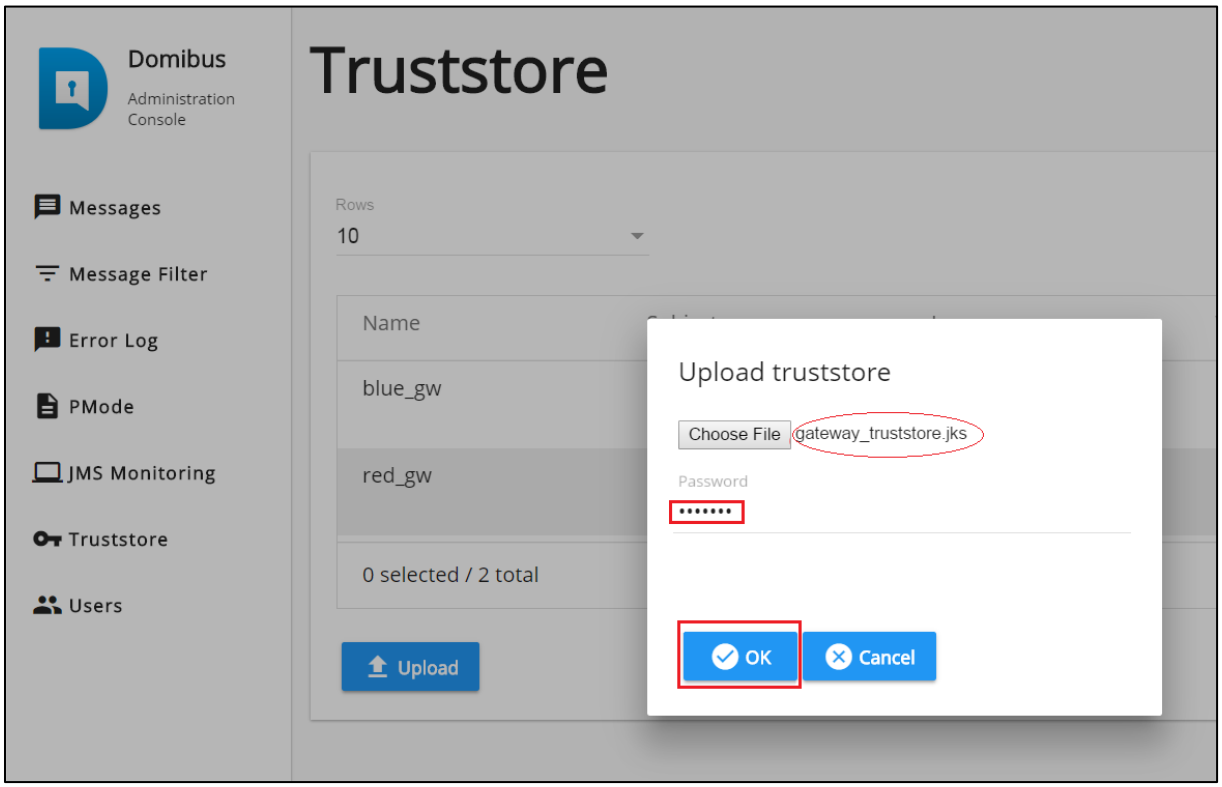
1. Select the "Truststore" menu and press the **Upload** button:



2. Navigate to the Truststore and select it by clicking on the **Open** button (or equivalent) of the standard file open dialog:



3. Once the file has been selected, enter the keystore password and click on the **OK** button to activate the new **truststore jks file:**



7.1.5. *Message properties validation*

While exchanging AS4 messages using PMode configuration, a user could define Message Properties as in the example below:

```
<ns:UserMessage>
....
   <ns:MessageProperties>
      <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</ns:Property>
       <ns:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C4</ns:Property>
        </ns:MessageProperties>
...
</ns:UserMessage>
```

Domibus has a limitation of 1024 characters for the value of a Message Property. If this value is exceeded, an EbMS3Exception is thrown on both sending (C2) and receiving (C3) side and the message is not submitted/accepted.

```
                <properties>
                        <property name="originalSenderProperty"
                                        key="originalSender"
                                        datatype="string"
                                        required="true"/>
                        <property name="finalRecipientProperty"
                                        key="finalRecipient"
                                        datatype="string"
                                        required="true"/>
```

```
                <propertySet name="eDeliveryPropertySet">
                    <propertyRef property="finalRecipientProperty"/>
                    <propertyRef property="originalSenderProperty"/>
                </propertySet>
            </properties>
```

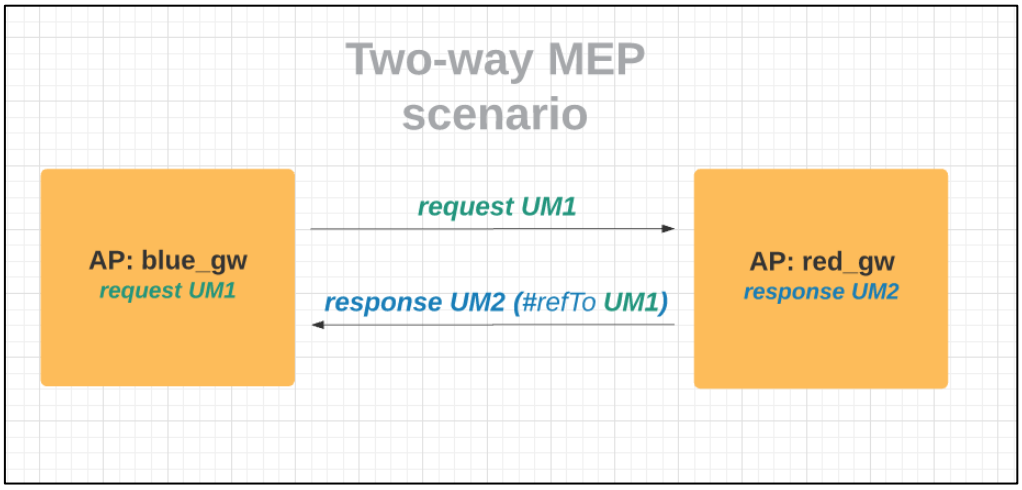
8. Two-way MEP Scenario

The **Two-Way MEP** governs the exchange of two User Messages in opposite directions, the first one being the request, and the second one being the response. The response must reference the request using eb:RefToMessageId.

A two-way scenario is presented below, including the PMode configuration for all 3 possible bindings for two-way exchanges: PushAndPush, PushAndPull, PullAndPush.

The scenario is the following: blue_gw wants to place an order to red_gw and expects a response from red_gw.

Blue_gw has the 'request' UserMessage that needs to be exchanged with the red_gw, and red_gw has the 'response' UserMessage that needs to be exchanged with blue_gw.



Processes described below simulate the 3 possible bindings for Two-Way mep.

Two legs are used: leg1 for the exchange of the request UM1 and leg2 for the exchange of response UM2. The legs are reused in all 3 bindings.

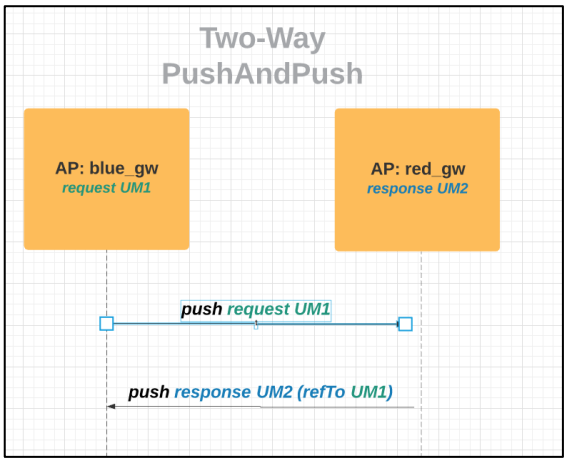
```
<legConfiguration name="leg1"
                  service="serviceA"
                  action="action1"
                  defaultMpc="mpcA"
                  reliability="AS4Reliability"
                  security="eDeliveryAS4Policy"
                  receptionAwareness="receptionAwareness"
                  propertySet="eDeliveryPropertySet"
                  payloadProfile="MessageProfile"
                  errorHandling="demoErrorHandling"
                  compressPayloads="true"/>
<legConfiguration name="leg2"
                  service="serviceA"
                  action="action2"
                  defaultMpc="mpcA"
                  reliability="AS4Reliability"
                  security="eDeliveryAS4Policy"
```

```
                                    receptionAwareness="receptionAwareness"
                                    propertySet="eDeliveryPropertySet"
                                    payloadProfile="MessageProfile"
                                    errorHandling="demoErrorHandling"
                                    compressPayloads="false"/>
```

8.1. PushAndPush binding

pushLeg1: blue_gw pushes the request UM1 on leg1

pushLeg2: red_gw pushes the response UM2 on leg2 - requires RefToMessageId: UM1



PMode configuration:

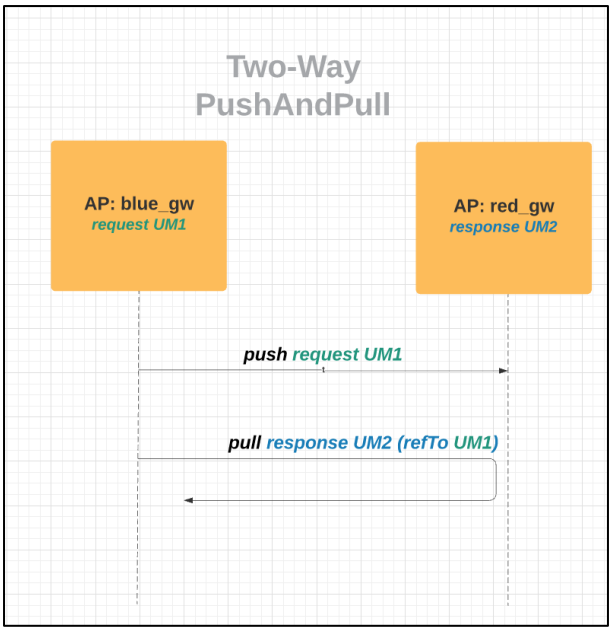
```
<process name="pushLeg1"
      mep="oneway"
      binding="push"
      initiatorRole="defaultInitiatorRole"
      responderRole="defaultResponderRole">
  <initiatorParties>
          <initiatorParty name="blue_gw"/>
  </initiatorParties>
  <responderParties>
          <responderParty name="red_gw"/>
  </responderParties>
  <legs>
          <leg name="leg1"/>
  </legs>
</process>
<process name="pushLeg2"
      mep="oneway"
      binding="push"
      initiatorRole=" defaultResponderRole "
      responderRole="defaultInitiatorRole">
  <initiatorParties>
          <initiatorParty name="red_gw"/>
  </initiatorParties>
  <responderParties>
```

```
        <responderParty name="blue_gw"/>
    </responderParties>
    <legs>
        <leg name="leg2"/>
    </legs>
</process>
```

8.2. PushAndPull binding

pushLeg1: blue_gw pushes the request UM1 on leg1

pullLeg2: blue_gw pulls the response UM2 on leg2 - requires RefToMessageId: UM1



PMode configuration:

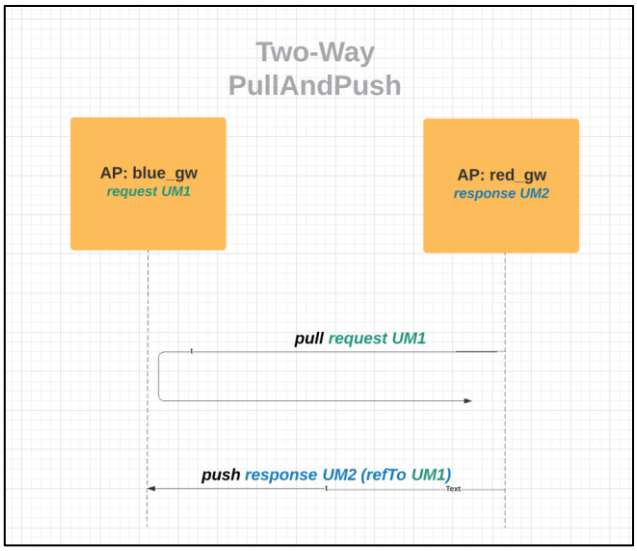
```
<process name="pushLeg1"
      mep="oneway"
      binding="push"
      initiatorRole="defaultInitiatorRole"
      responderRole="defaultResponderRole">
    <initiatorParties>
        <initiatorParty name="blue_gw"/>
    </initiatorParties>
    <responderParties>
        <responderParty name="red_gw"/>
    </responderParties>
    <legs>
        <leg name="leg1"/>
    </legs>
</process>
<process name="pullLeg2"
      mep="oneway"
      binding="pull"
```

```
        initiatorRole="defaultResponderRole"
        responderRole="defaultInitiatorRole">
    <initiatorParties>
            <initiatorParty name="blue_gw"/>
    </initiatorParties>
    <responderParties>
            <responderParty name="red_gw"/>
    </responderParties>
    <legs>
            <leg name="leg2"/>
    </legs>
</process>
```

8.3. PullAndPush binding

pullLeg1: red_gw pulls the request UM1 on leg1

pushLeg2: red_gw pushes the response UM2 on leg2 - requires RefToMessageId: UM1




```
<process name="pullLeg1"
      mep="oneway"
      binding="pull"
      initiatorRole="defaultInitiatorRole"
      responderRole="defaultResponderRole">
        <initiatorParties>
            <initiatorParty name="red_gw"/>
        </initiatorParties>
        <responderParties>
            <responderParty name="blue_gw"/>
        </responderParties>
        <legs>
            <leg name="leg1"/>
        </legs>
</process>
<process name="pushLeg2"
      mep="oneway"
```

```
    binding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
        <initiatorParties>
                <initiatorParty name="red_gw"/>
        </initiatorParties>
        <responderParties>
                <responderParty name="blue_gw"/>
        </responderParties>
        <legs>
                <leg name="leg2"/>
        </legs>
</process>
```

9. Special Scenario: Sender and Receiver are the same

In this special scenario, the Sender Access Point acts also as the Receiver Access Point. Multiple backends can exchange messages via the same Access Point using the same or different plugins.

9.1. PMode Configuration

A party (e.g. **blue_gw**) which is Sender and Receiver must be defined in both the <initiatorParties> and <responderParties> sections as shown below:

```
.....
        <initiatorParties>
                ……
                <initiatorParty name="blue_gw"/>
                ……
        </initiatorParties>
        <responderParties>
                ……
                <responderParty name="blue_gw"/>
                ……
        </responderParties>
.....
```

9.2. Message structure

A message that is sent to the same Access Point will have to contain the same party id in both **From** and **To** sections. Below there is an example of a message sent using the Default WS Plugin:

```
<ns:UserMessage>
…
 <ns:PartyInfo>
  <ns:From>
   <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
blue</ns:PartyId>
   <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
  </ns:From>
  <ns:To>
   <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
blue</ns:PartyId>
   <ns:Role>http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/responder</ns:Role>
  </ns:To>
</ns:PartyInfo>
….
```

9.3. Message ID convention

Due to some limitations related to the uniqueness of the message identifier, a convention has been defined in this scenario. The message ID used for the received message is derived from the message ID used for the sent message with the following rule: the suffix ***"_1"*** is added to the sent message id.

Example:
sent message ID is **ae15851e-78fb-4b51-aac8-333c08c450d6@domibus**
received message ID is **ae15851e-78fb-4b51-aac8-333c08c450d6@domibus_1**

10. ADMINISTRATION TOOLS

10.1. Administration Console

Domibus administration console can be used by administrators and users to easily manage Domibus application.

The administration dashboard is reachable via the following URLs:

- http://your_server:your_port_number/domibus (Tomcat, WildFly and Weblogic)

The admin console is made of several sections:

Messages

On this page, the administrator can see the details of the messages and re-process them if required. The administrator can also navigate through the messages history and download specific messages if needed.

Message Filter

On this page, the administrator can set defined filters and access them individually for edition directly in the list.

Error Log

On this page, the administrator can view the list of application errors, make searches on error messages and filter them.

PMode

On this page, the administrator can upload, download and edit the PMode file. The administrator can also edit the list of parties configured in the PMode and access them individually for modification purposes. The user has also access to a list of archived PMode content that the user can restore.

JMS Monitoring

On this page, the administrator can monitor and manage the contents of the JMS queues.

Certificates

On this section, the user can manage the Domibus keystores, truststores and TLS truststores. The user can upload new keystore, truststore and TLS truststores to replace/remove the current one and define its password.

To force the reading of the keystore from the disk, the user can click on the Reload Keystore button on the Keystore page.

Users

On this page, the administrator can create and manage users including: grant access rights, change passwords, assign roles, etc.

Plugin Users

On this page, the administrator can manage the plugin users: create, delete, edit, grant access rights and roles, etc.

Audit

On this page, the administrator has an overview of changes performed in the PMode, Parties, Message Filter and Users pages.

Alerts

This page displays the alerts generated by Domibus in case of unusual behaviour of the application. The alerts are configured by the administrator.

Connection Monitoring

On this page the administrator can perform basic test of the communication configuration between two access points and see the status of these connections.

Logging

This page displays the logging levels of various libraries and packages and to change their levels.

Properties

This page displays the Domibus and external modules properties and their values and allows to change them.

Domains

This page displays the existing domains in Multi tenancy configuration and allows to activate or de-activate them at runtime.

Change Password

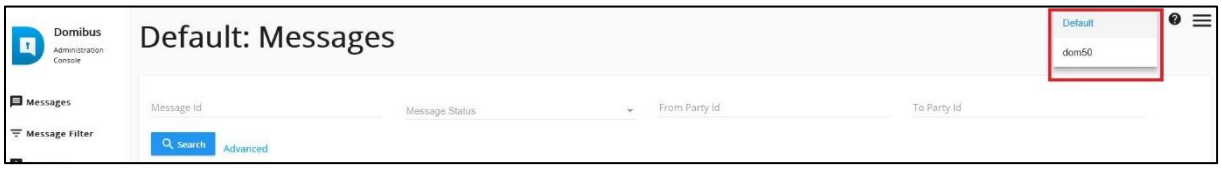
It is accessible from the hamburger menu found at the top-right corner of the screen. On this page the administrator can change his/her password if it is about to expire. This page is displayed also automatically, after the login, if the user has the default password.

10.2. Multitenancy

In Multitenancy mode, each tenant domain has its own set of configuration files: Keystore, Truststore, PMode, Domain properties, etc. Users are defined for each tenant domain.

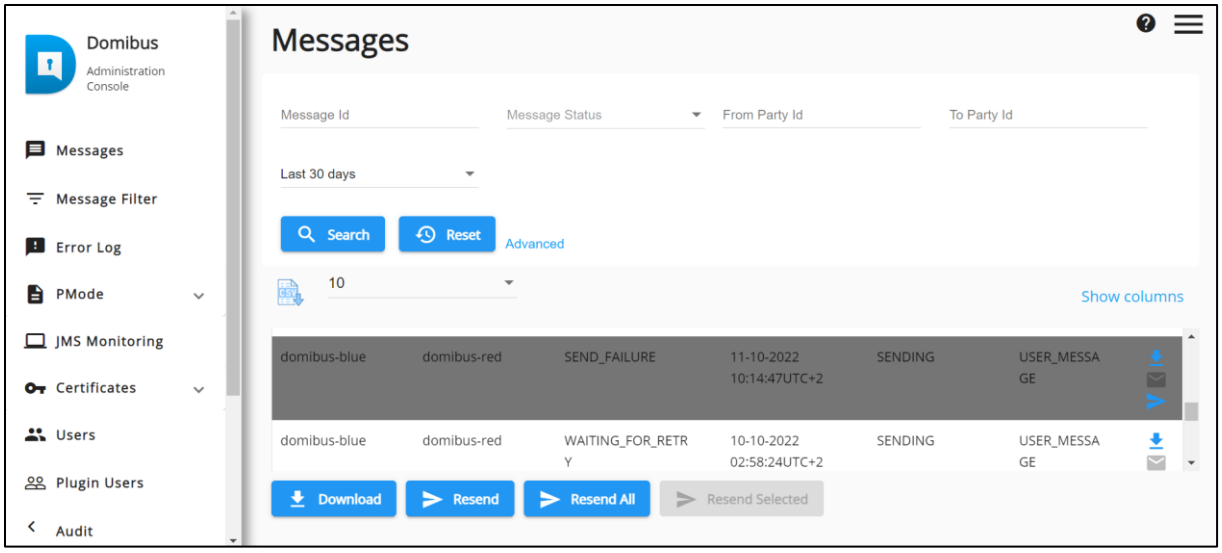
The user named **super** with role **ROLE_AP_ADMIN**, has the privileges to access all the available domains.

When logged as **super**, you are able to select a specific tenant domain in the upper right part of the admin console in a drop-down list (default or dom50 domain in the example below):



10.3. Message Log

Domibus administration dashboard includes a message logging page that gives the administrator information related to sent messages, received messages and their status (SENT, RECEIVED, FAILED, ACKNOWLEDGED, etc.):



From the admin console, the user can resend failed messages in different ways using 3 different buttons:

- **Resend**: resend one failed message either using the Resend Icon for each row or the Resend button at the bottom of the page.

- **Resend Selected**: resend several selected failed messages from the UI at the same time as a batch (Maximum limit =100).

- **Resend All** - resend all the failed messages filtered in the UI at the same time.
  - Domibus will first apply a search filter in the UI to find all the failed message id's which are eligible to be resent.
  - Domibus will store all the failed message id's to be resent in the 'tb_messages_to_resend' table in the DB.
  - The MessageResend quartz job will check the table for new entries every minute (the default value is stored in Domibus properties file as a cron expression and can be changed).
  - Domibus will get all the failed message Id's from the table and restore one message at a time.
  - If the number of messages to restore is >10, Domibus will directly restore the messages without storing in DB.

There is also support for downloading the non-repudiation XML receipts.

The following state machines illustrate the evolution of the processing of messages according to the encountered events:

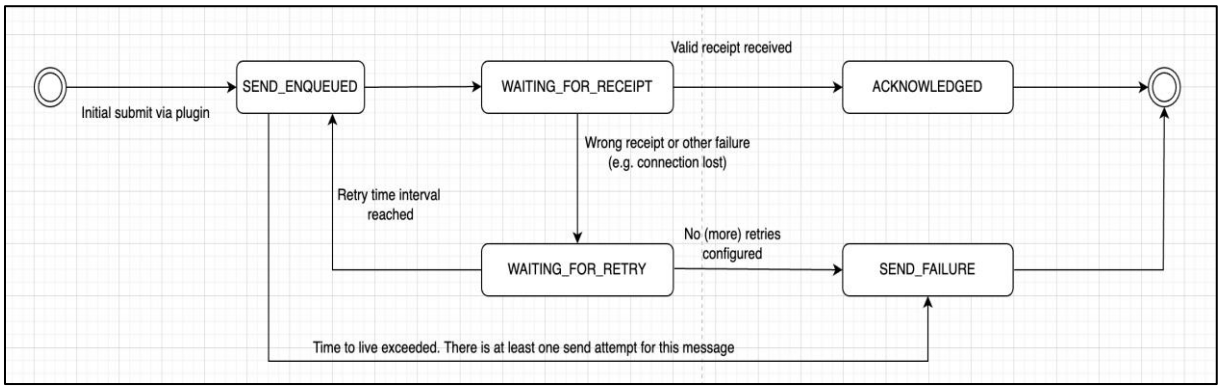


Figure 5 - State machine of Corner 2 (sending access point)

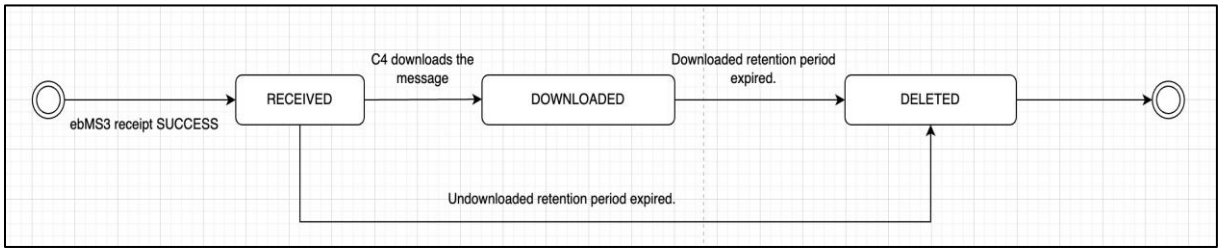


Figure 6 - State machine of Corner 3 (receiving access point)

10.4. Message Filtering

Domibus allows the routing of messages to different plugins, based on some messages attributes:

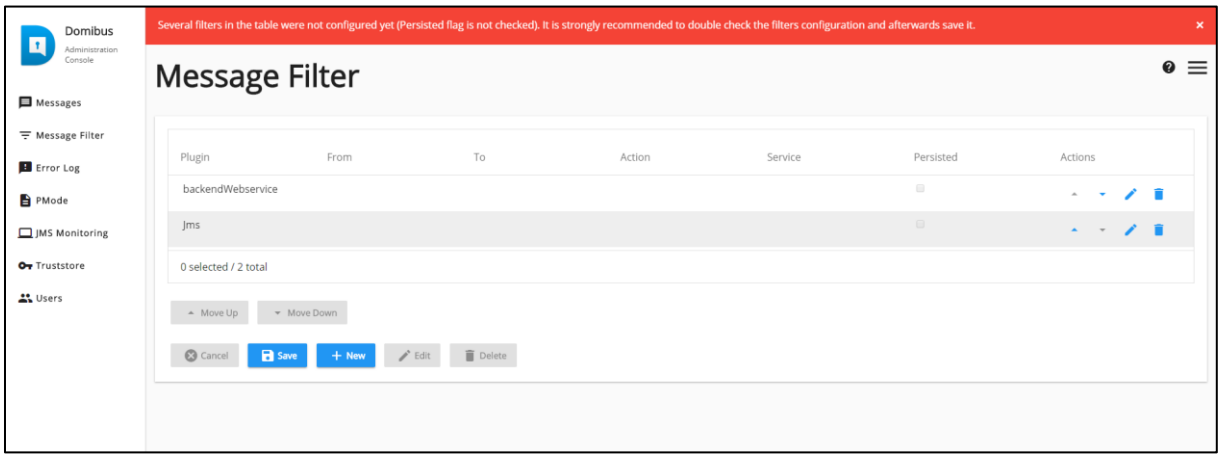
- **From**: initial sender (C1)
- **To**: final recipient (C4)
- **Action**: defined as 'Leg' in the PMode
- **Service**: as defined in the PMode

The following rules apply:

- Domibus takes into account the ordered list of 'filters' to route all messages. The first filter matching the filter criteria will define the target plugin. The order of the plugin is therefore important in the routing process.

  Note 1: if the filters are all mutually exclusive, the order would not matter.

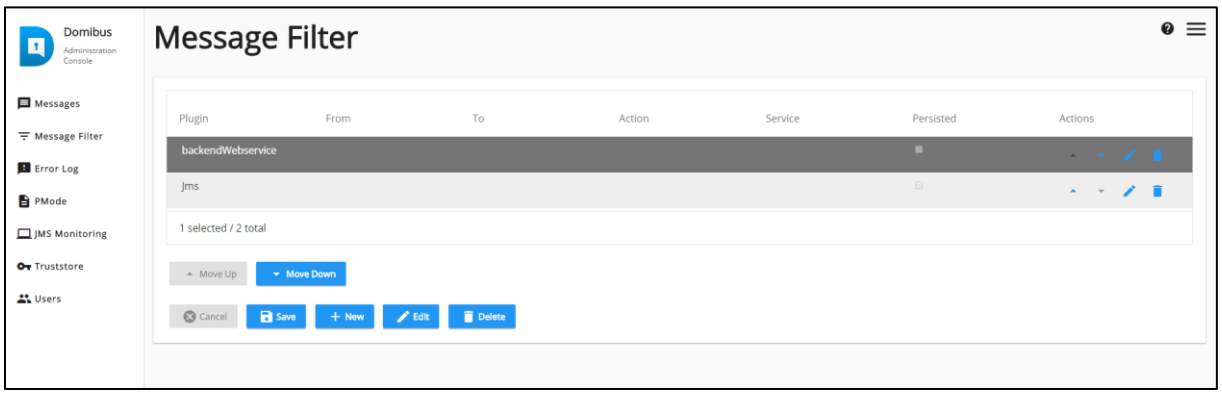
  Note 2: The 'Persisted' column indicates whether the plugin filter configuration has already been saved. If a plugin filter configuration has not already been saved, the 'Persisted' value is unchecked and an error message is shown on the top of the screen. In this case, it is strongly recommended to review the filters configuration and save it afterwards.



- One plugin may be applied to multiple filters. This is done by the use of the 'OR' criteria. (cf. backendWebservice in the example below).

- Multiple attributes could also be defined in one filter. This is done by the use of the 'AND' criteria.
  (cf. the first filter in the example below).

- One filter may have no criteria, meaning that all messages (not matching previous filters) will be routed to the corresponding plugin automatically. As a result, subsequent filters will therefore not be considered for any incoming message. In the example below, the last filter routes all remaining messages to plugin 'backendWebservice'.



Use the **New** and **Delete** buttons to create or delete a filter.

As the order matters, move up and down actions allow placing each filter in the right order:

Cf. **Move Up** and **Move Down** buttons.

After some changes have been applied to the filters, the **Cancel** and **Save** buttons become active:

- Press **Cancel** to cancel the changes
- Press **Save** to save the changes and activate them immediately.

The console will ask the user to confirm the operation, before proceeding.

Example of message attributes used for routing and matching the first filter used in the example above:

- **Action**: *TC1Leg1*
- **Service**: *bdx:noprocess:tc2*
- **From**: domibus-blue:urn:oasis:names:tc:ebcore:partyid-type:unregistered
- **To**: domibus-red:urn:oasis:names:tc:ebcore:partyid-type:unregistered

That information can be found in the incoming message received by Domibus (e.g. see below):

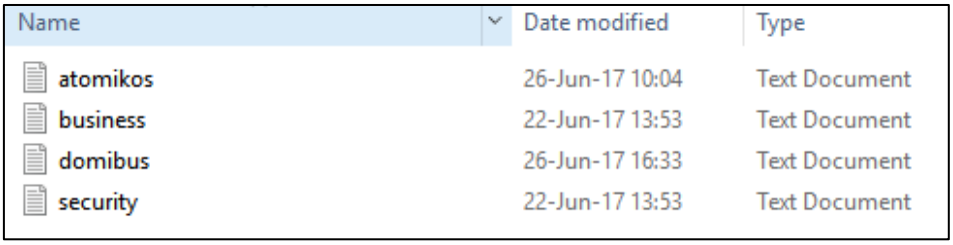
```
<ns:PartyInfo>
        <ns:From>
          <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
blue</ns:PartyId>
          <ns:Role>http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
        </ns:From>
        <ns:To>
          <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
red</ns:PartyId>
          <ns:Role>http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/responder</ns:Role>
        </ns:To>
</ns:PartyInfo>
      <ns:CollaborationInfo>
        <ns:Service type="tc1">bdx:noprocess</ns:Service>
        <ns:Action>TC1Leg1</ns:Action>
      </ns:CollaborationInfo>
```

10.5. Application Logging

10.5.1. Domibus log files

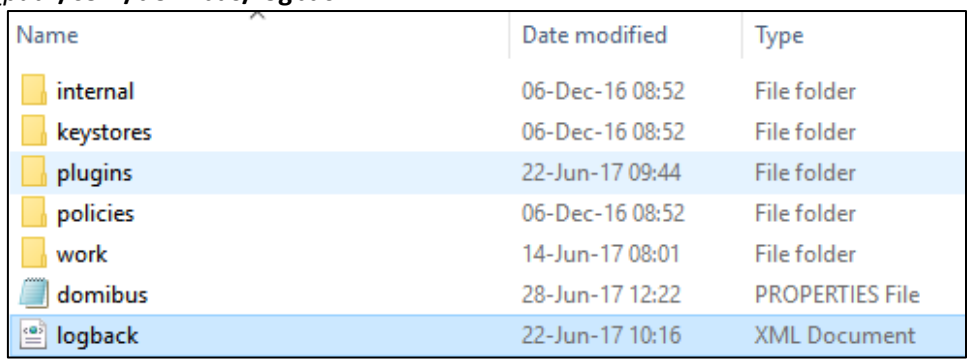
Domibus has three log files listed below:

1. domibus.log: this is the main log file log and contains both the security and business logs plus miscellaneous logs as debug information, logs from one of the framework used by the application, etc.
2. domibus-security.log: this log file contains all the security related information. For example, you can find information about the clients who connect to the application. By default, the security information is included in domibus.log and this log is disabled
3. domibus-business.log: this log file contains all the business related information. For example, when a message is sent or received, etc. By default, the business information is included in domibus.log and this log is disabled.
4. statistics.log: includes information on the occurrence of different events (receive message, submit message, etc).



10.5.2. Logging properties

It is possible to modify the configuration of the logs by editing the logging properties file: *edelivery_path***/conf/domibus/logback.xml**:



Async logging:

It is possible to improve logging speed and reduce logging latency by using async logging.
An example is present in the logging properties file: *edelivery_path***/conf/domibus/logback.xml**

a. Uncomment the part
```
<!-- Async logging: uncomment this-->
<!-- <appender name="DEFAULT-ASYNC-FILE" class="ch.qos.logback.classic.AsyncAppender">-->
<!-- <queueSize>3000</queueSize>-->
<!-- <discardingThreshold>0</discardingThreshold>-->
<!-- <appender-ref ref="file" />-->
<!-- </appender>-->
```

b. Comment the line
```
<appender-ref ref="file"/>
```

c. Uncomment the line
```
<!-- <appender-ref ref="DEFAULT-ASYNC-FILE" />-->
```

d. The root logging should look like this:
```
<root level="WARN">
<appender-ref ref="DEFAULT-ASYNC-FILE" />
<appender-ref ref="stdout"/>
</root>
```

e. Restart the application server

If you wish your log pattern to include the line number where the logging method was called, you should not use the standard logback %L or %line, but instead use %domibusLine (that was declared as a conversion rule in your xml file):

<conversionRule conversionWord="domibusLine"
converterClass="eu.domibus.logging.DomibusLineOfCallerConverter" />

<property name="encoderPattern" value="%d{ISO8601} [%X{d_user}] [%X{d_domain}]
[%X{d_messageId}] [%X{d_messageEntityId}] [%thread] %5p %c{1}:%domibusLine - %m%n"
scope="global"/>

10.5.3. Error Log page

To go to the error log page of the Domibus Admin Console, click on the **Error log** menu entry.

This option lists all the Message Transfers error logs and includes the **ErrorSignalMessageId**, **ErrorDetail** and **Timestamp**. You can sort messages by using the up or down arrow to search for a specific message.

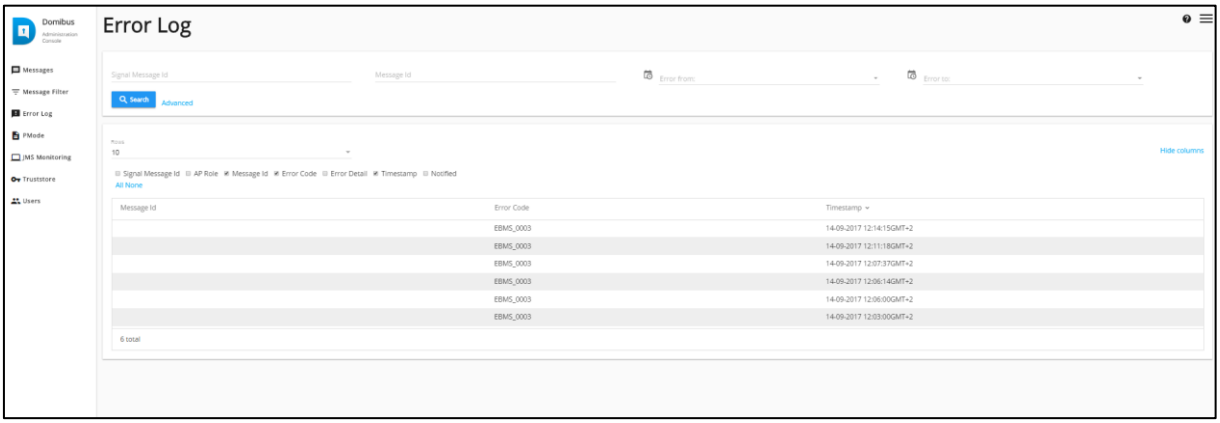


Figure 7 - Domibus – Error Log page

10.6. PMode

In the Administration console you can view the content of the current PMode:

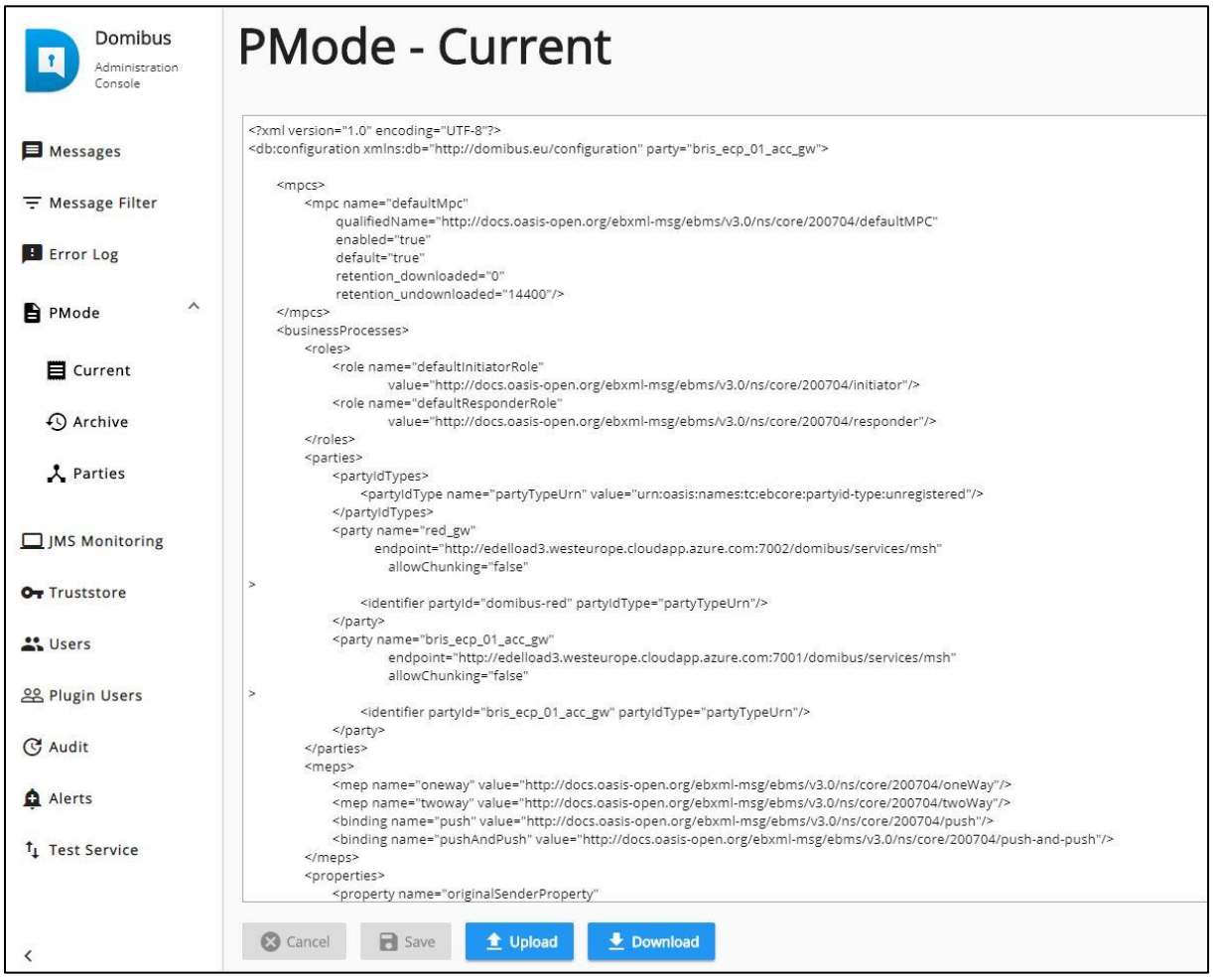
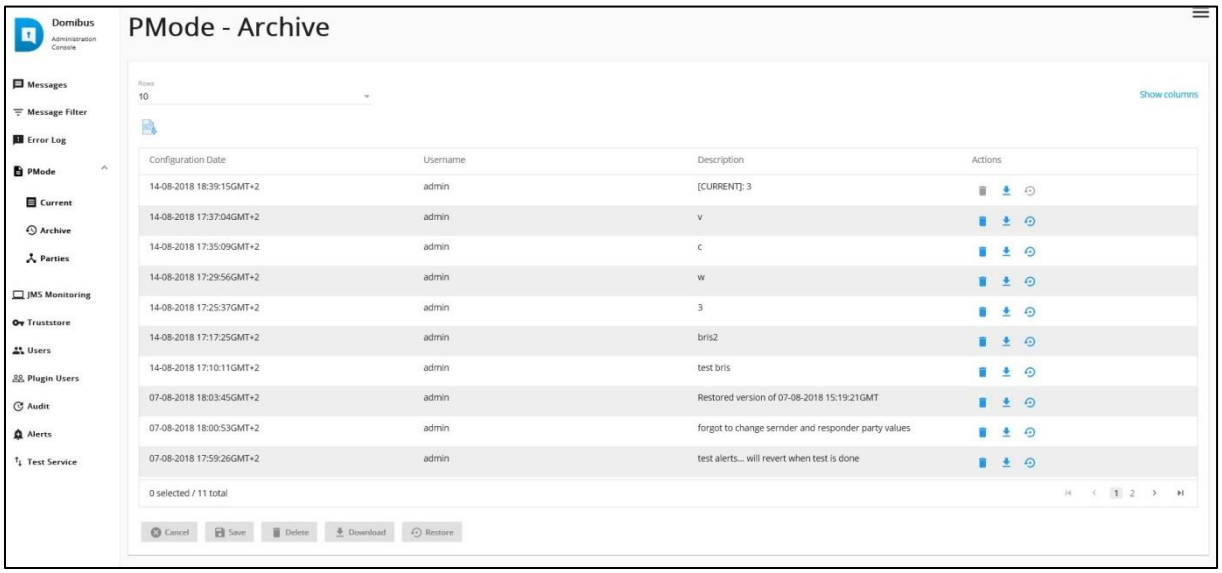


Figure 8 – PMode page

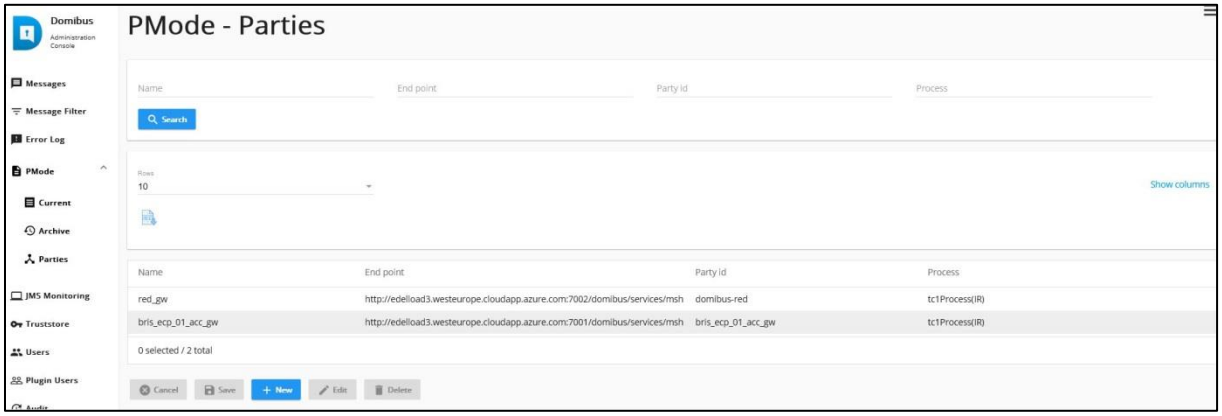
You can edit the content of your current PMode in the administration console and save the changes by clicking on **Save** or discard the changes by clicking on **Cancel**. You can **upload** a PMode file or **download** the current one.

Under **Archive** the history of the PMode changes is displayed:



Domibus keeps a snapshot of the PMode each time the PMode is modified. The user can restore a particular version and make it the current PMode by clicking on the restored button at the far right of the table.

Under Parties, the user can manage the parties in the PMode. Parties can be searched using filter criteria, they can be added, updated or deleted.



The PMode is updated and a new PMode snapshot is created when parties are added, updated or deleted.

10.7. Queue Monitoring

***Remark*:**
To prevent the user from moving messages from any queue to any other queue:

- The user should be able to move messages only to the original queue which can be retrieved from the JMS Messsage properties.
- In case the original queue cannot be determined, the user can move to any queue the message except the source.
- In case of more than one message to be moved, all messages must have the same original queue. Otherwise, an error message is displayed.

- In case the original queue is the same as the source queue, an error message is displayed.

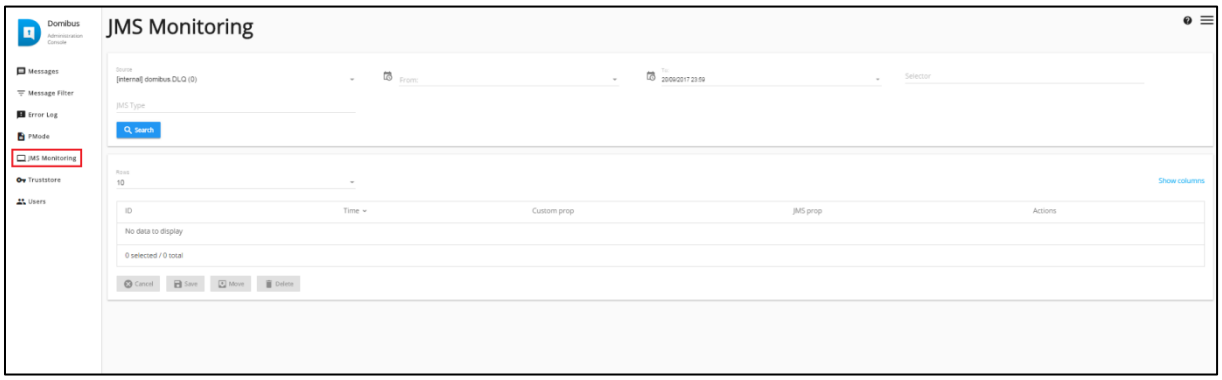
Domibus uses following JMS queues to handle the messages:

Destination type	JNDI name	Comment	Description
Queue	jms/domibus.internal.dispatch.queue	No redelivery because redelivery of MSH messages is handled via ebMS3/AS4.	This queue is used for scheduling messages for sending via the MSH.
Queue	jms/domibus.internal.notification.unknown		Notifications about received messages (by the MSH) that do not match any backend routing criteria will be sent to this queue. In production environment, this queue should be monitored to handle those messages manually.
Topic	jms/domibus.internal.command		This topic is used for sending commands to all nodes in a cluster. For example, it is used after a PMode was uploaded to notify all nodes to update their PMode cache (in case caching is enabled).
Queue	jms/domibus.backend.jms.replyQueue		This queue is used for sending replies back to the sender of a message. Replies contain: a correlationId, ebMS3 messageId (if possible), error messages (if available).
Queue	jms/domibus.backend.jms.outQueue		Messages received by the MSH (that match the routing criteria for the JMS plugin) will be sent to this queue.

Queue	jms/domibus.backend.jms.inQueue		This queue is the entry point for messages to be sent by the sending MSH.
Queue	jms/domibus.backend.jms.errorNotifyConsumer		This queue is used to inform the receiver of a message that an error occurred during the processing of a received message.
Queue	jms/domibus.backend.jms.errorNotifyProducer		This queue is used to inform the sender of a message that an error occurred during the processing of a message to be sent.
Queue	jms/domibus.notification.jms		Used for sending notifications to the configured JMS plugin.
Queue	jms/domibus.internal.notification.queue		This queue is used to notify the configured plugin about the status of the message to be sent.
Queue	jms/domibus.notification.webservice		Used for sending notifications to the configured WS plugin.
Queue	jms/domibus.DLQ		This is the Dead Letter Queue of the application. The messages from other queues that reached the retry limit are redirected to this queue.

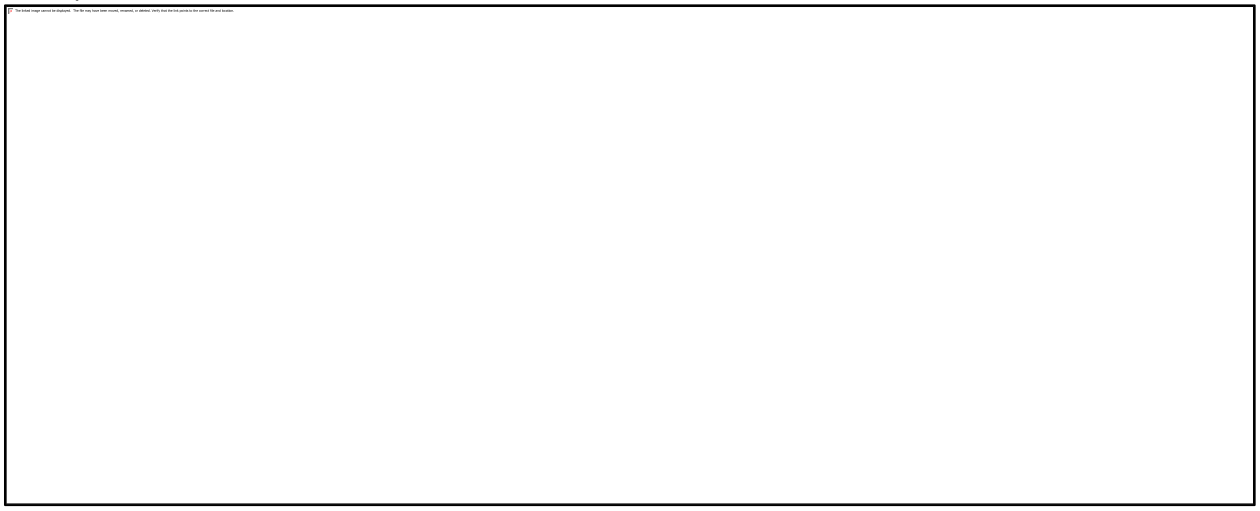
Table 4 - Queue Monitoring

All these queues can be monitored and managed using the **JMS Monitoring** page, which is accessible from the **JMS Monitoring** menu of the administration console:



Warning:
For Tomcat server, the maximum number of shown messages in the queue monitoring is defined by the 'domibus.listPendingMessages.maxCount' property.

In the **Source** field, we have all the queues listed, along with the number of messages pending in each queue:



If a queue is used internally by the application core, its name will start with **[internal].** A regular expression is used to identify all the internal queues. The value for this regular expression can be adapted in the **domibus.jms.internalQueue.expression** property from the *edelivery_path***/conf/domibus/domibus.properties** file.

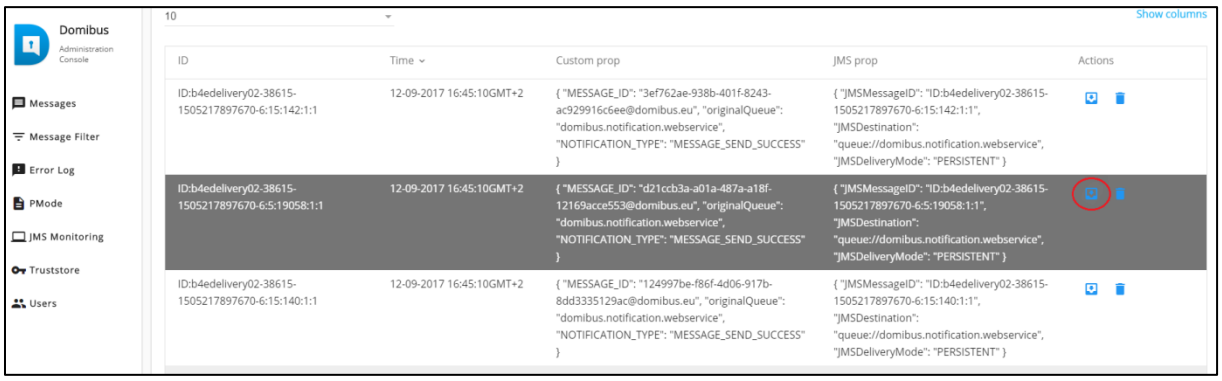
In the **JMS Monitoring** page the following operations can be performed:

1. Inspecting and filtering the messages from a queue based on the fields:
    - **JMS type**: the JMS header
    - **Selector**: in this field you can enter any JMS message properties with the correct expression to filter on it

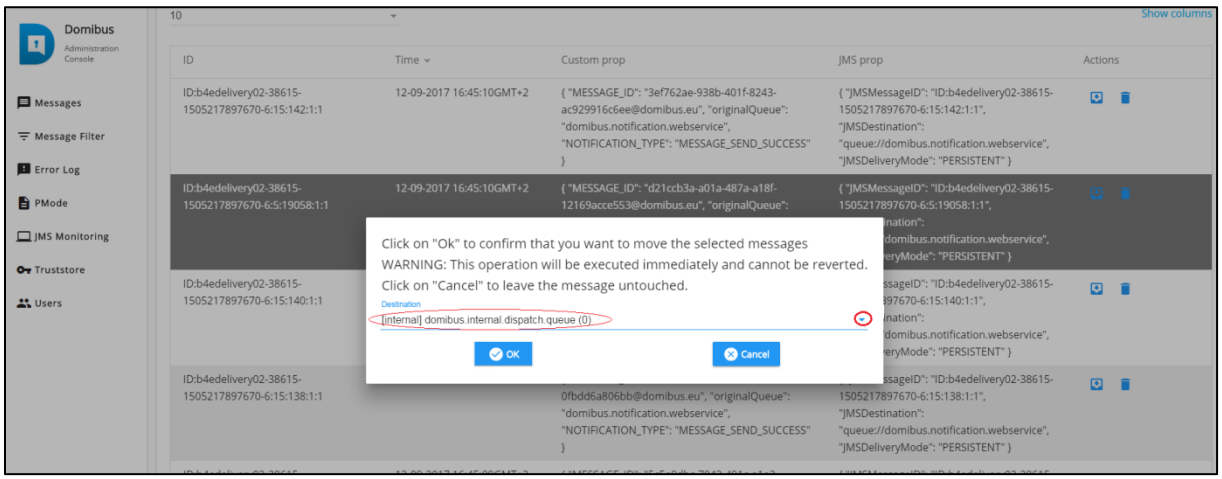
Remark:

For more information on the JMS message headers and the JMS message selector, please check the official documentation at https://docs.oracle.com/cd/E19798-01/821-1841/bnces/index.html.

2. Move a message:
    a. Move the message from the DLQ to the original queue:
        - Select the JMS message from the DLQ and press the **Move** icon (**in RED marker**):

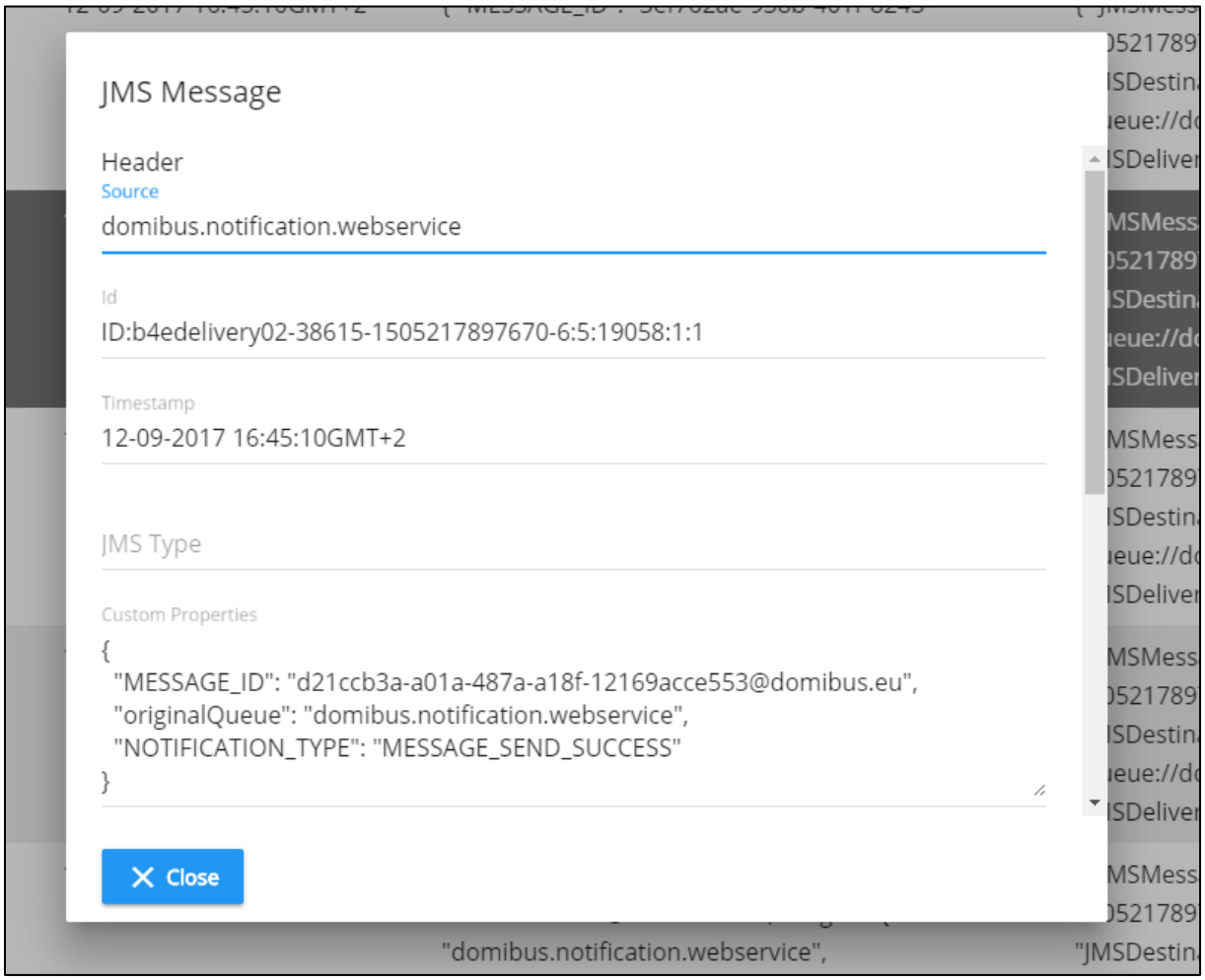


        - Select the original queue from the **Destination** dropdown list in the dialog box:



        - Press the **Ok** button in the dialog, and the message will be moved to the original queue.

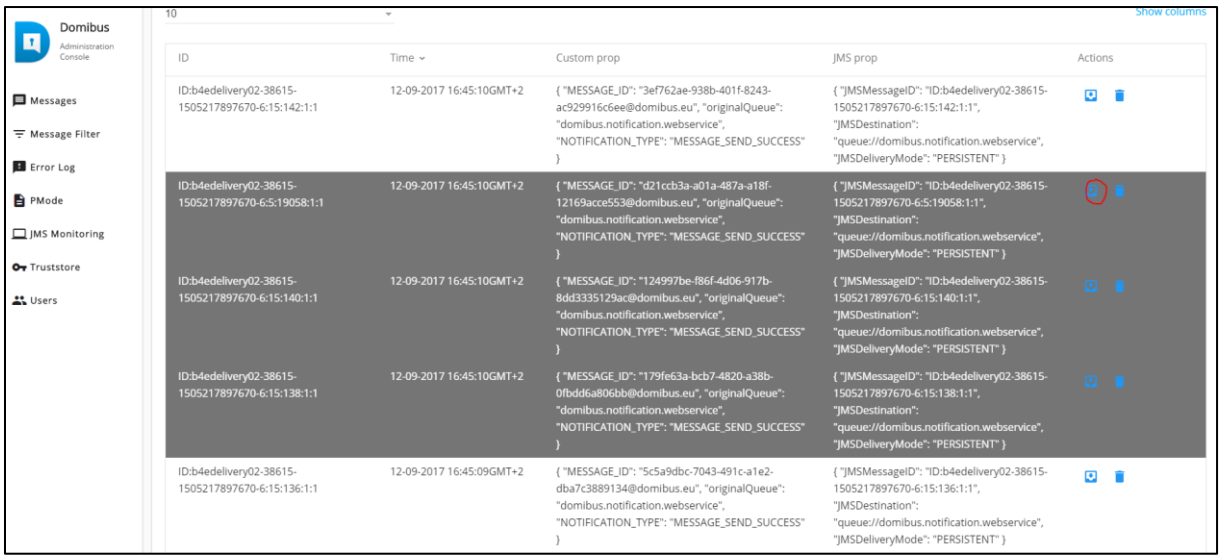
Note: the details of a message can be viewed by selecting it (double-clicking) from the message list:



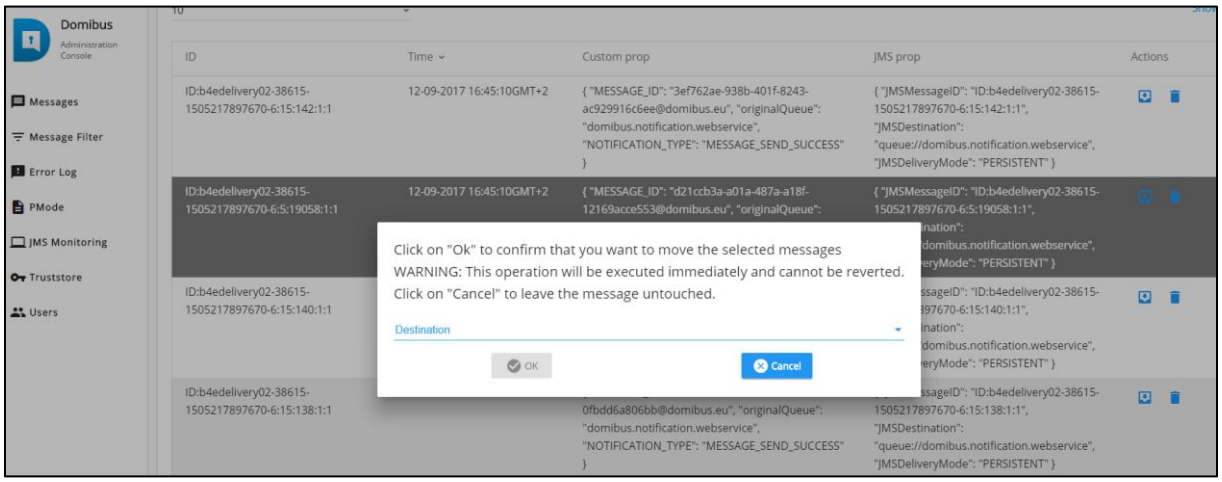
Click **Close** to exit the dialog box.

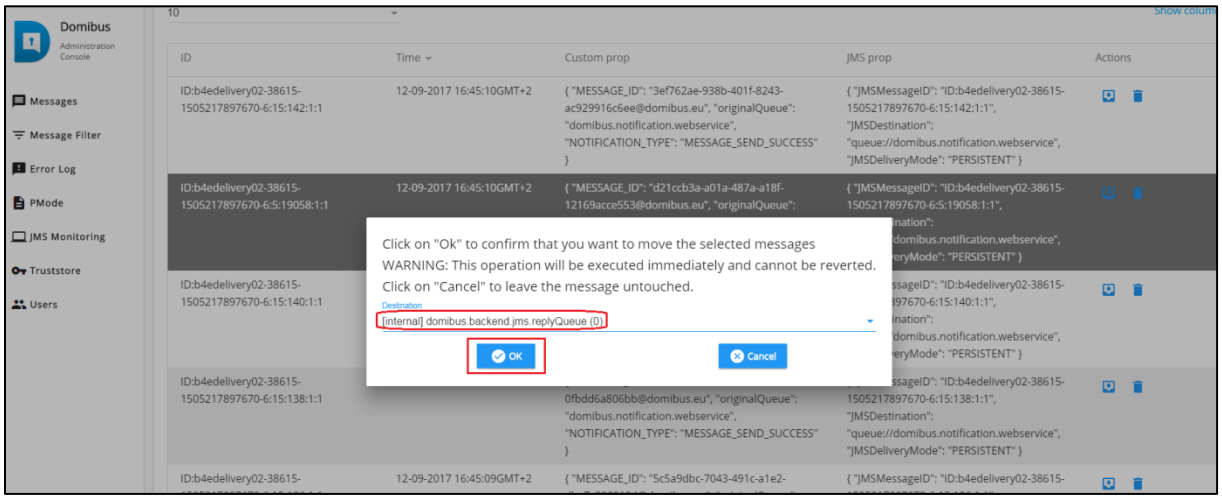
b. Move multiple messages from the DLQ to the original queue:

- Select multiple JMS messages from the DLQ and press the **Move** icon button:



- Select the original queue from the Destination dropdown list, and click **Ok**.

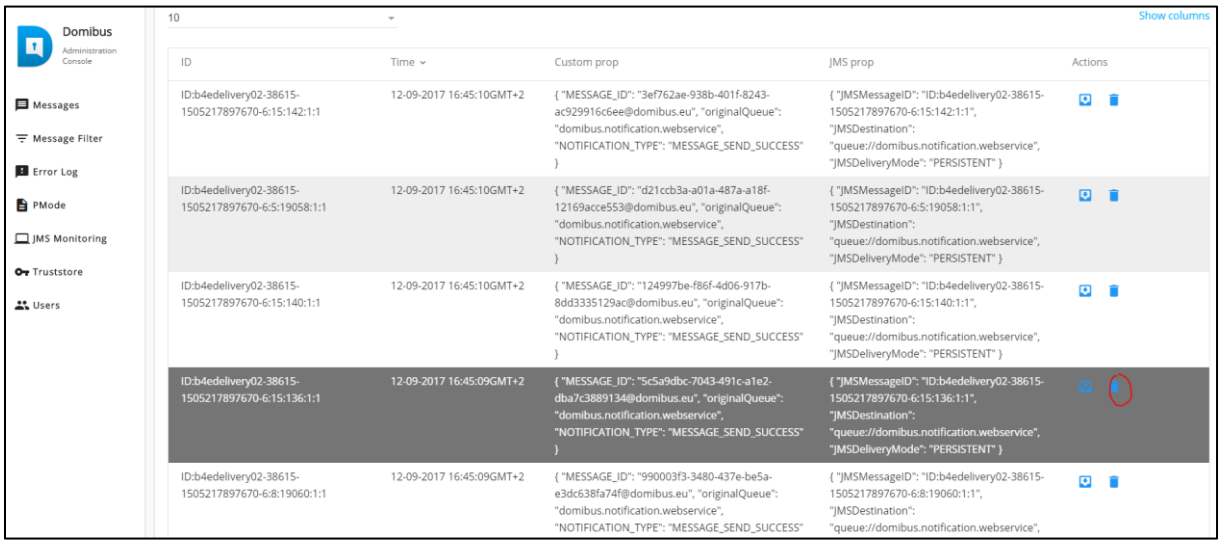




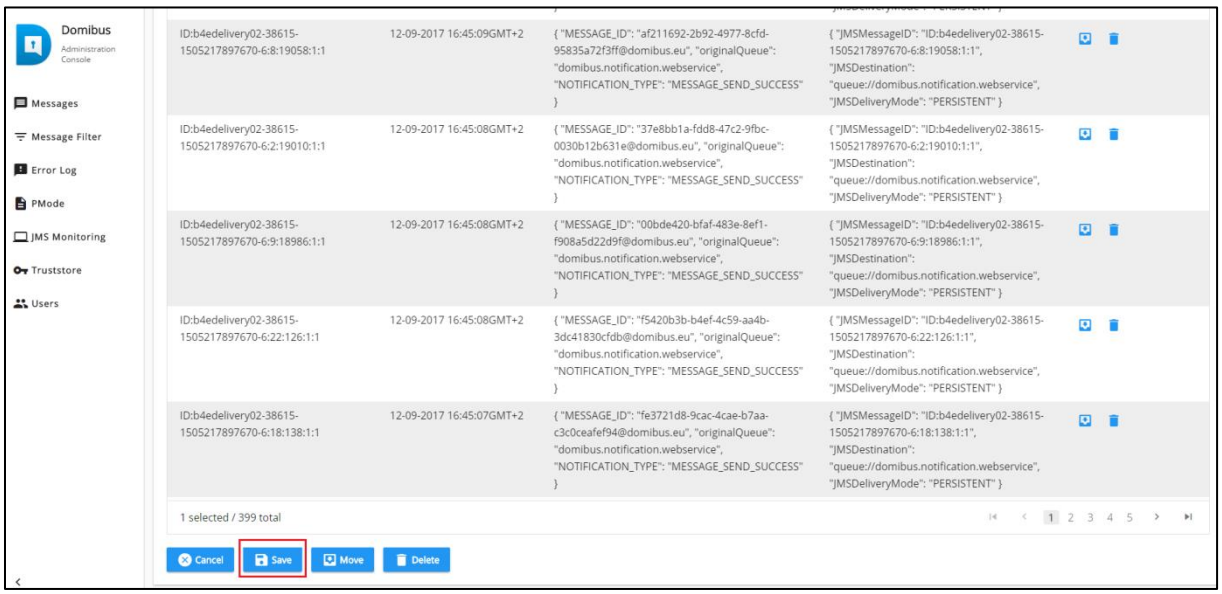
***Remark**:*

Please make sure that all the selected messages came from the same source queue. Use the filtering capabilities to ensure this.

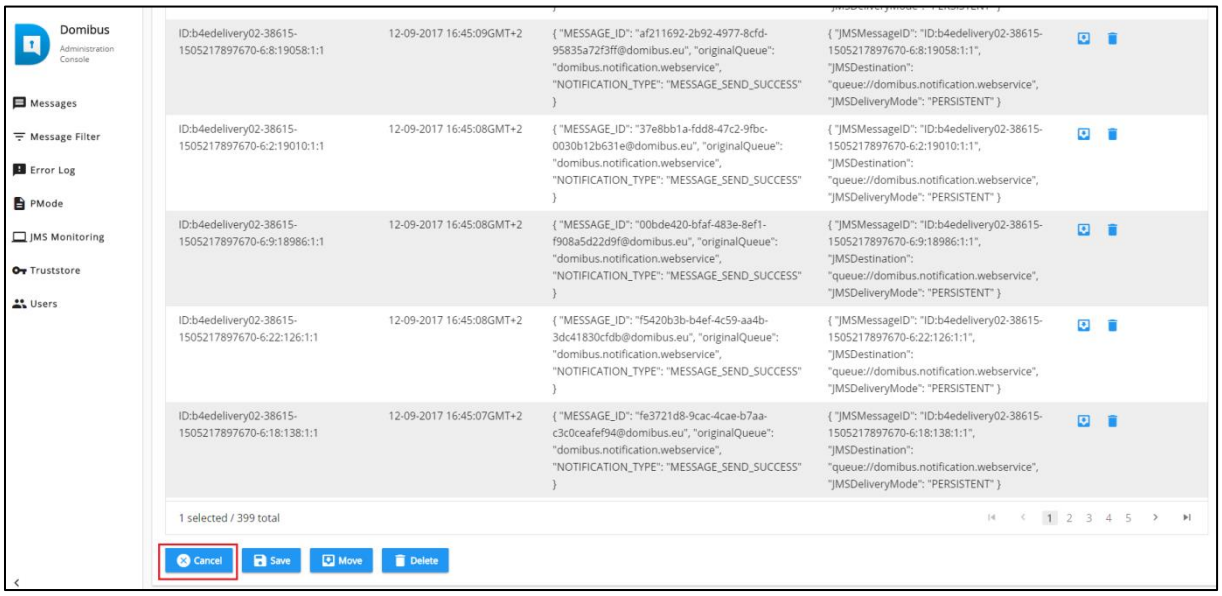
3. Delete message(s): delete one or more messages from one queue:
   - Select one or several JMS messages from the source queue and press the **Delete** button:



- By clicking the **Delete** button, the selected messages are removed from the screen, but you still have to confirm your changes by clicking on the **Save** button. As long as you have not clicked on the **Save** button, your changes are not taken into account in the system.



- To cancel the changes you made, click on the **Cancel** button instead:



10.8. Configuration of the queues

Queues should be configured appropriately and according to the backend system needs and re-delivery policy.

10.8.1. Tomcat

Domibus uses ActiveMQ as JMS broker. The various queues are configured in the *edelivery_path***/conf/domibus/internal/activemq.xml** file.

Please see ActiveMQ redelivery policy and configure the parameters below if needed:

```
<redeliveryPlugin fallbackToDeadLetter="true"
            sendToDlqIfMaxRetriesExceeded="true">
  <redeliveryPolicyMap>
    <redeliveryPolicyMap>
      <defaultEntry>
                    <!-- default policy-->
                    <redeliveryPolicy maximumRedeliveries="10"      redeliveryDelay="300000"/>
            </defaultEntry>
       <redeliveryPolicyEntries>
            <redeliveryPolicy queue="domibus.internal.dispatch.queue"
maximumRedeliveries="0"/>
            <redeliveryPolicy queue="domibus.internal.pull.queue" maximumRedeliveries="0"/>
</redeliveryPolicyEntries>
    </redeliveryPolicyMap>
  </redeliveryPolicyMap>
</redeliveryPlugin>
```

Access to the JMS messaging subsystem is protected by a username and a password in clear text defined in the domibus.properties file *edelivery_path***/conf/domibus/domibus.properties**.
It is recommended to change the password for the default user:

```
activeMQ.username=domibus
activeMQ.password=changeit
```

Remark:

*The user (**activeMQ.username**) and the password (**activeMQ.password**) defined in the **domibus.properties** file are referenced in the authentication section of the **activemq.xml** file provided.*

10.8.2. WebLogic

Please use the admin console of WebLogic to configure the re-delivery limit and delay if necessary.

10.8.3. WildFly

Please use the admin console of WildFly to configure the re-delivery limit and delay if necessary.

10.9. Certificates

In the **Certificates** section, you can manage the Domibus keystores, truststores and TLS truststores.

You can upload a new keystore, to replace the current one or download it.

To force the reading of the keystore from the disk, you can use the **Reload keystore** button on the **Keystore** page.

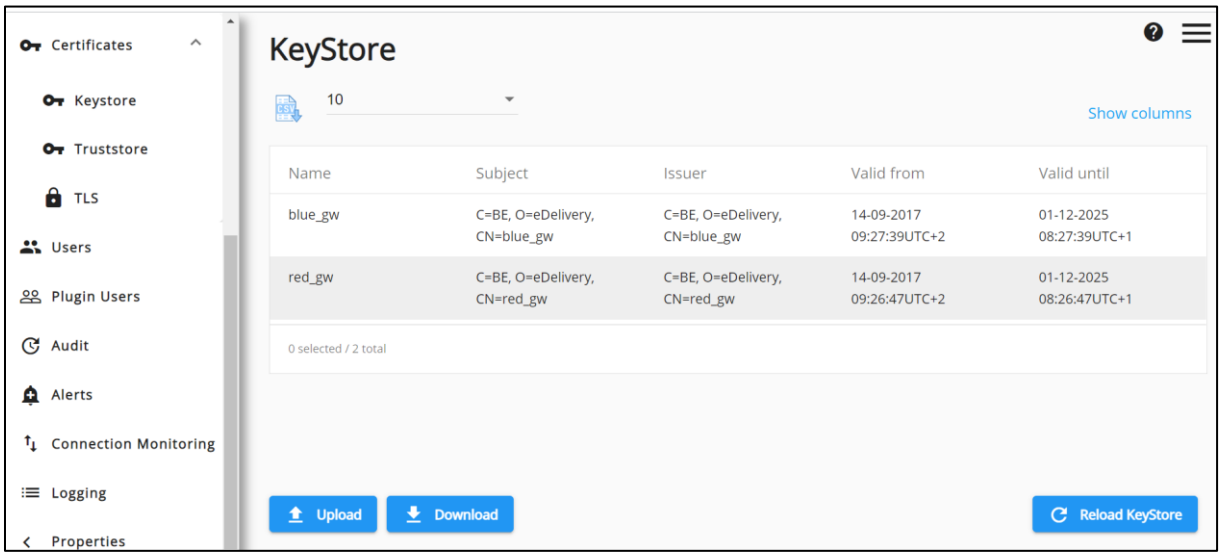
For Domibus keystore and truststore, to force the reloading of the keystore from the disk (possibly because it was modified manually there), one can use the **Reload** button located on the bottom right side of the page.

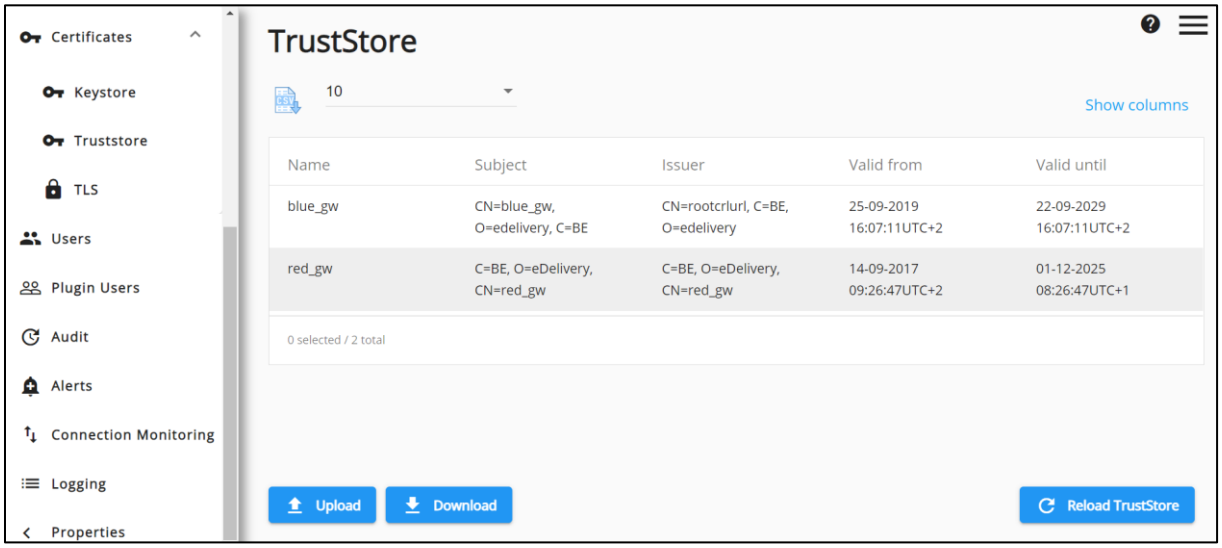
For Domibus Truststore and the TLS Truststore, besides uploading and downloading operations, one can manage the trusted certificates individually, by adding or removing them.

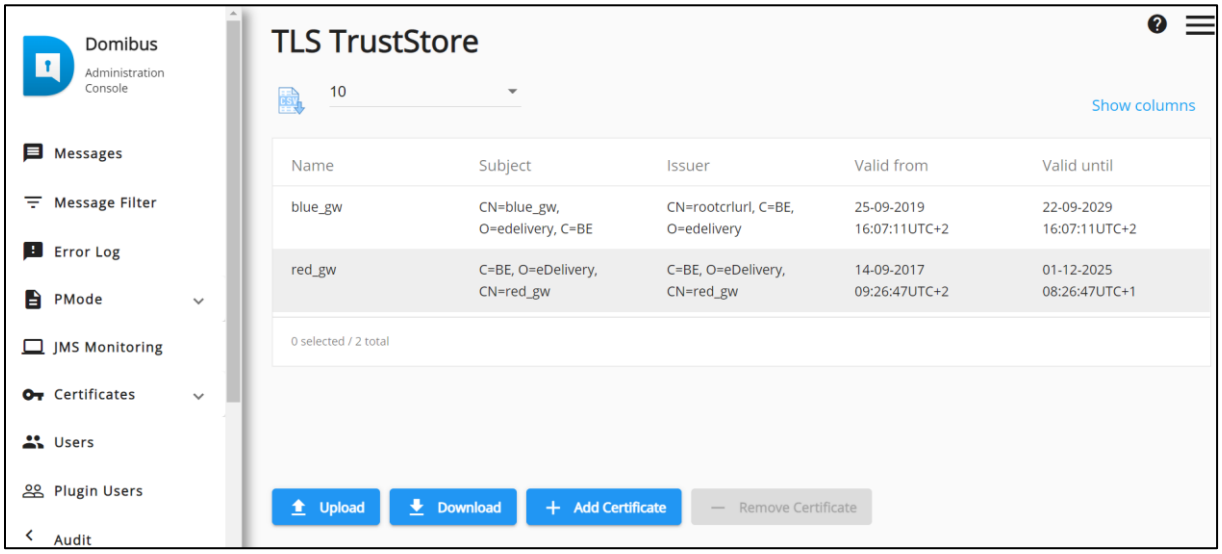
In case the newly uploaded truststore or keystore has a different type than the current used one, the Domibus properties that specify the location and type are also changed to to reflect this.

In case the uploaded store is identical to the currently used one, it will not be replaced, so no backup will be made. If one tries to add a certificate to the truststore and it is already there, it will not be added, so no backup will be made.

When upgrading from a 5.0.x (up to 5.0.4) version, when starting Domibus for the first time, Domibus looks for keystores in the database (if the keystores were kept in the database). Domibus finds newer keystores, they are copied on the disk and deleted from de database. A keystore that is kept in the database is considered as newer than the corresponding one on the disk if the last operation on it (upload, add, remove certificates) was done later than the modification time of the disk file.







Domibus also exposes the REST API external services for both truststore and TLS truststore for the following operations:

- To upload a truststore file
- To download a truststore file
- To get the entires from the truststore
- To add a certificate to the truststore
- To remove a certificate from the truststore

More API details of these external services are in Domibus REST Service Open API documentation which is part of the Domibus distribution artefacts (see [REF1]).

10.9.1. saveCertificateAndLogRevocationJob

saveCertificateAndLogRevocationJob checks and removes the unused certificates from Domibus. This job is triggered based on the cron property 'domibus.certificate.check.cron'. This cron expression specifies the frequency of the certificate revocation check. By default, this job run every day.

When a user manually deletes and replaces a certificate in the config location, he will get an alert asking "The store file on the disk has different content than the one loaded and used in Domibus. Would you like to refresh?": if the user clicks 'Yes', the file is replaced and the old certificate details in Domibus becomes unused.

SaveCertificateAndLogRevocationJob checks unused certificates and removes their details from Domibus certificate tables. It also creates a deleted entry in the audit table. The user can see the 'deleted' certificates (by domibus-quartz user) details in the Audit page of Domibus admin console.

When a user replaces the certificate via Domibus admin console, Domibus removes the old certificate details (unused) directly and creates a deleted entry in the audit table as a result of this replacement action. The user can see the 'deleted' certificates (by admin user) details in the Audit page of Domibus admin console. In this case, Domibus does a backup of the old certificates and it will store them in the backup folder in the config location.

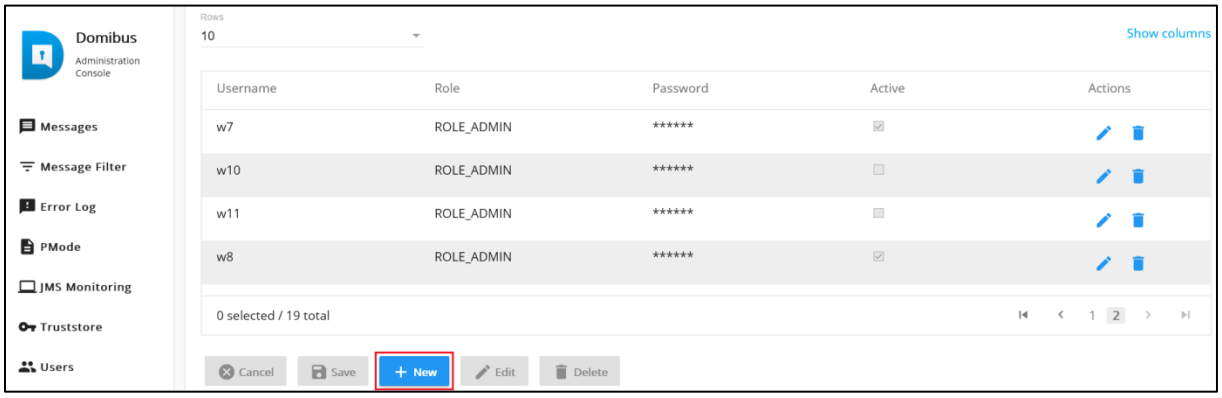
10.10. Users

Admin Console users are used to access Domibus Admin Console:

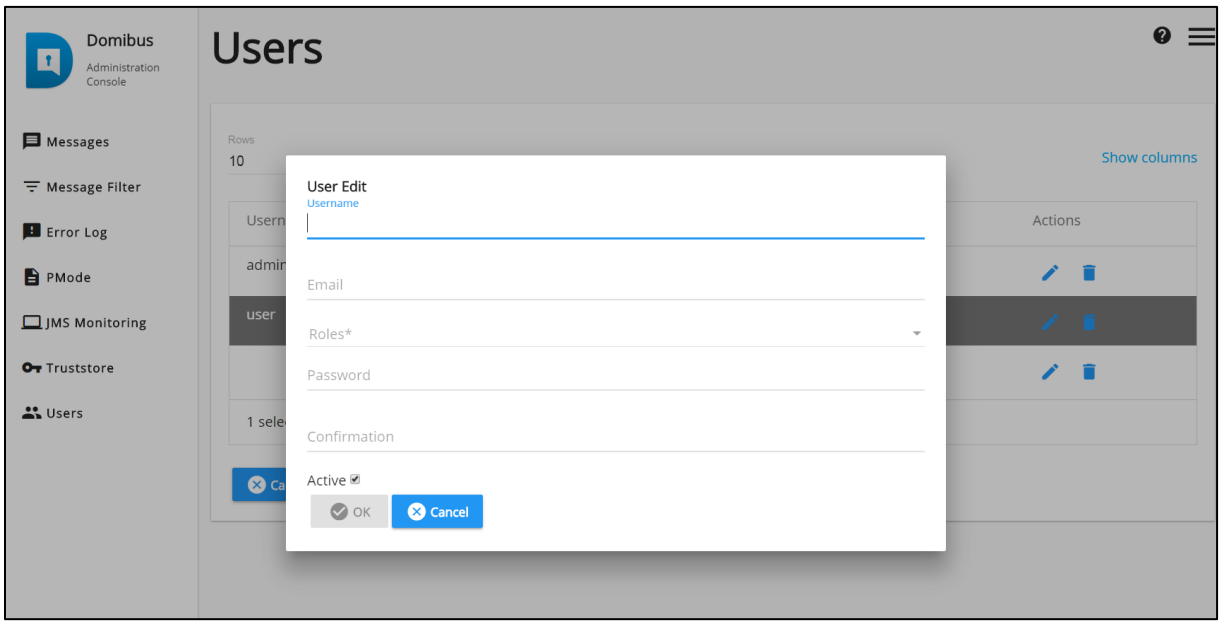
- ROLE_USER - can only access two pages Messages and Error Log, read only. Resending a failed message is not possible for this role.
- ROLE_ADMIN - has access to all pages, can change all that is allowed to be changed on these pages, but only for his own domain.
- ROLE_AP_ADMIN - similar to ROLE_ADMIN, only that it can see any domain. AP ADMIN can create and remove domains at runtime (with some limitations).

10.10.1. Adding new users

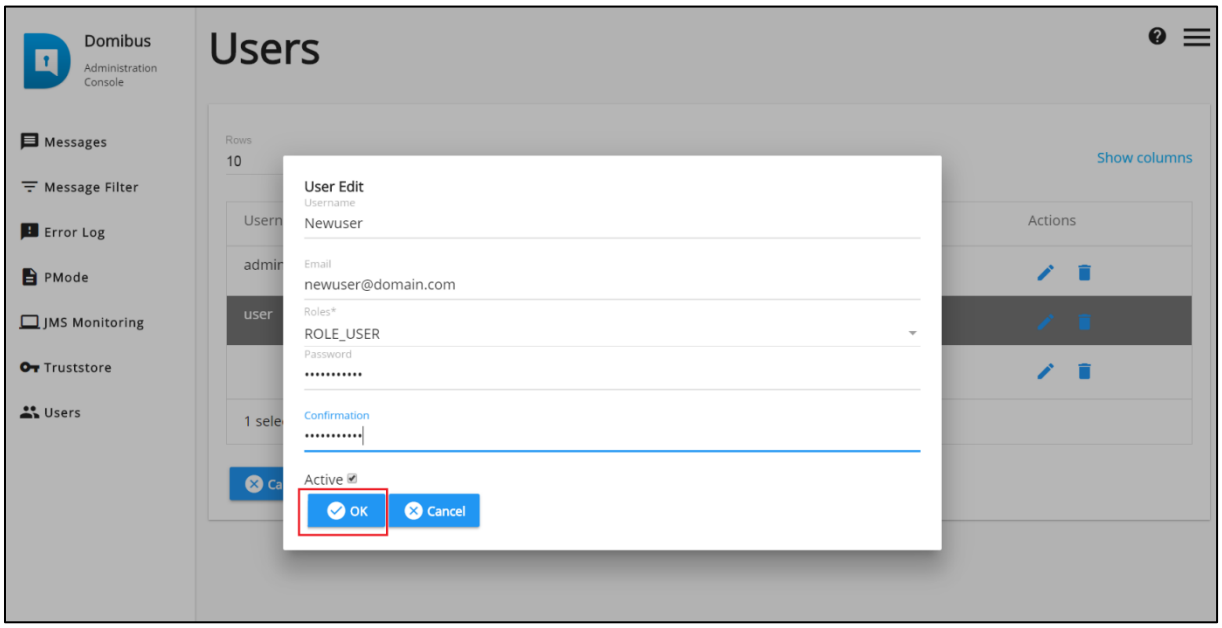
1. New users can be added to the existing default users (**admin** and **user**) by clicking on **New**:



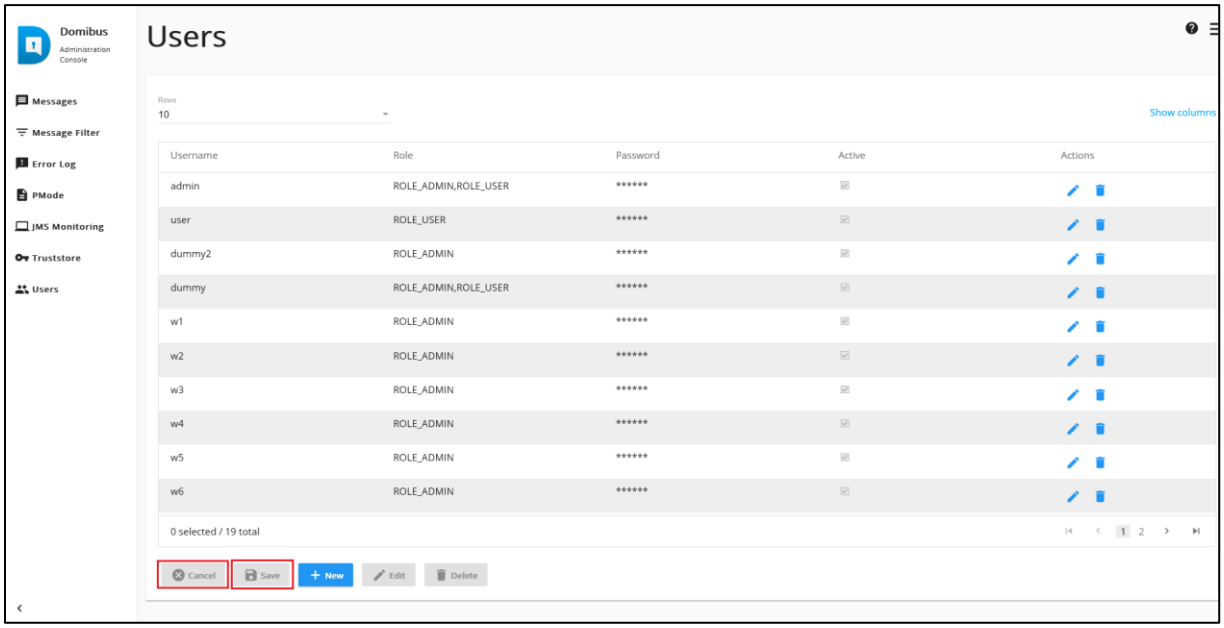
2. For each new user, you must enter a username, an email, a role and a password:



3. Click on **OK**:



4. Again, once the user has been created, do not forget to click on the **Save** button on the **Users** page to register your changes in the system:

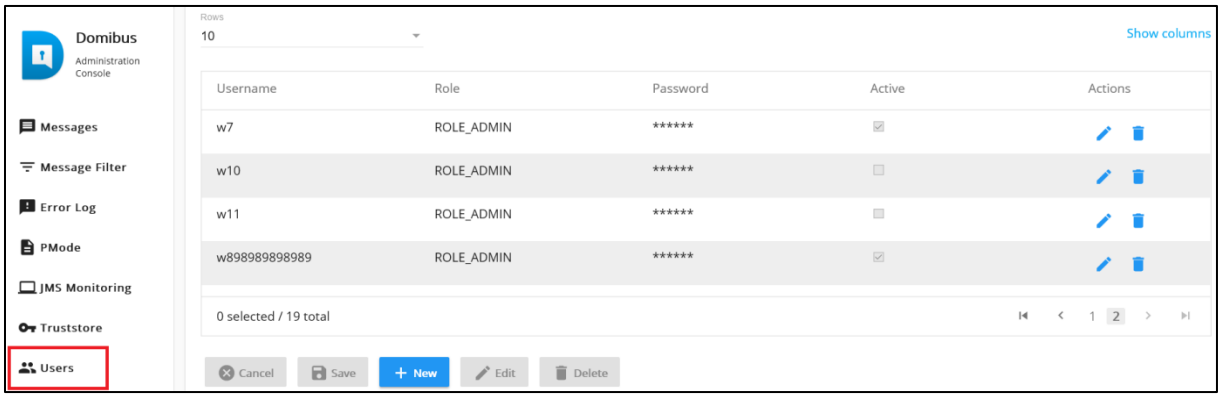


10.10.2. Changing passwords

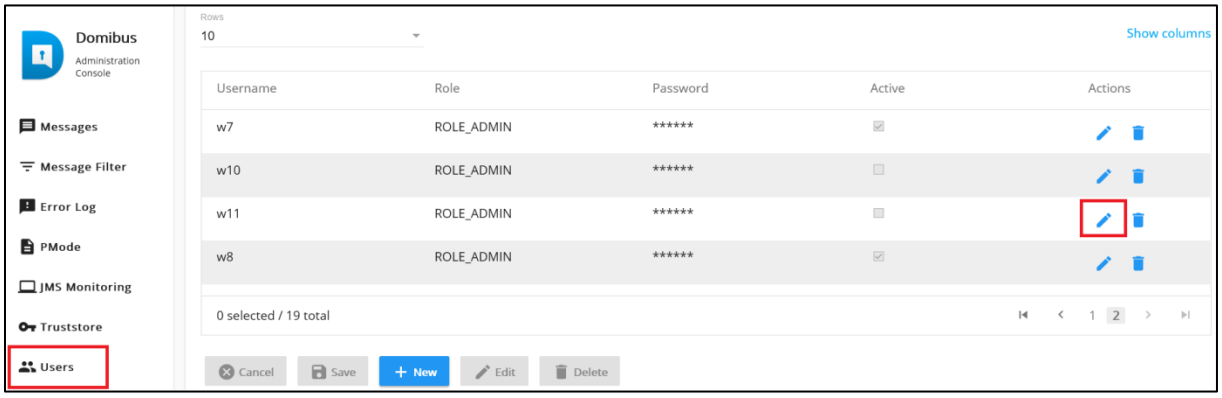
All user passwords have an expiration period, configured in the domibus properties. Some days before expiring (also configured in properties), the user receives a warning after the login and also an alert. The new password cannot be one of the last 5 used passwords (the number can be configured). Also, the password must meet complexity rules configured in the properties. If it does not meet them, then an error message is displayed (can also be configured).

The passwords of the default users (admin, user and super users) automatically expire after 3 days. This period can be configured. Once logged-in with the default password, the system redirects the user to the Change Password page so that he/she can immediately change it. The default password check can be disabled from the properties.

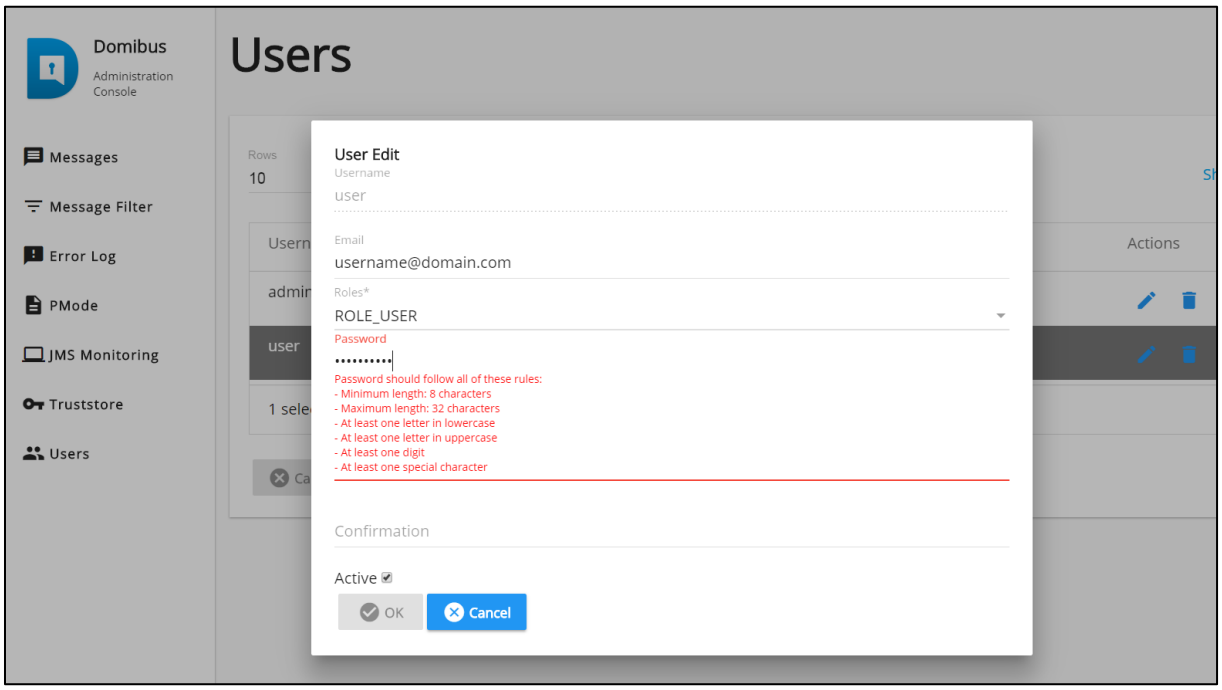
1. In order to change the password for a user, navigate to the **Users** menu entry to obtain the list of configured users:



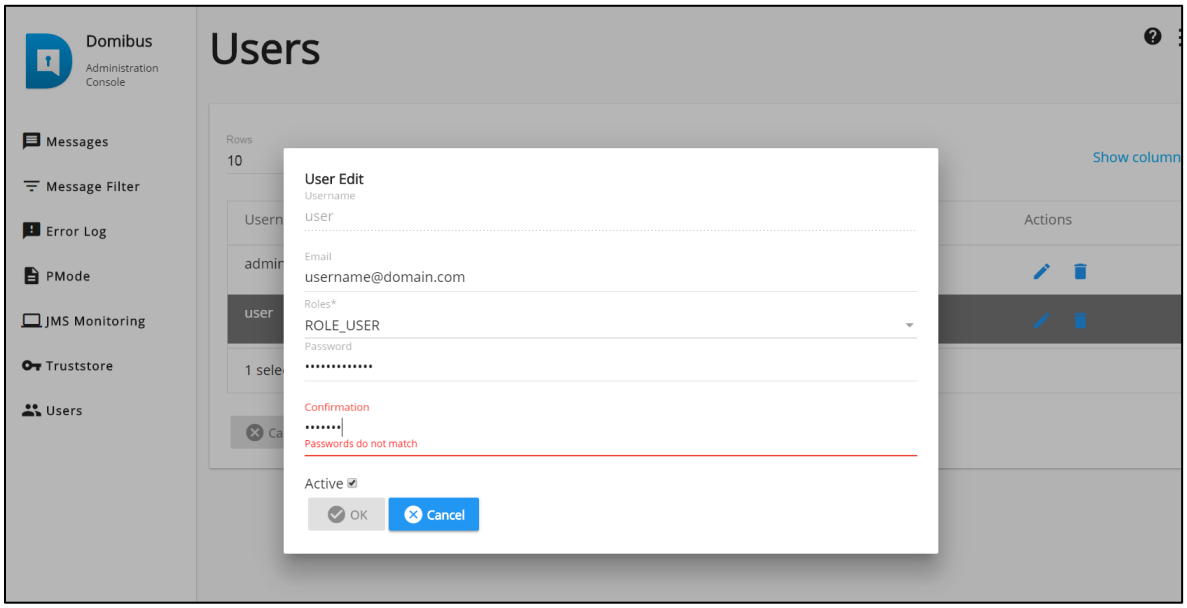
2. To edit the user details, click on the **EDIT** icon (in **RED**). DO NOT click on the BIN icon as this would DELETE the record.



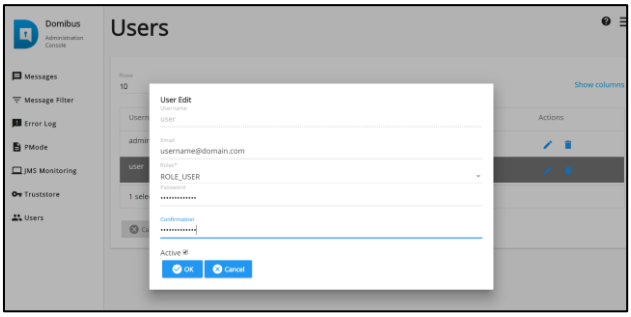
3. In the popup window, choose a new password using the rules shown:



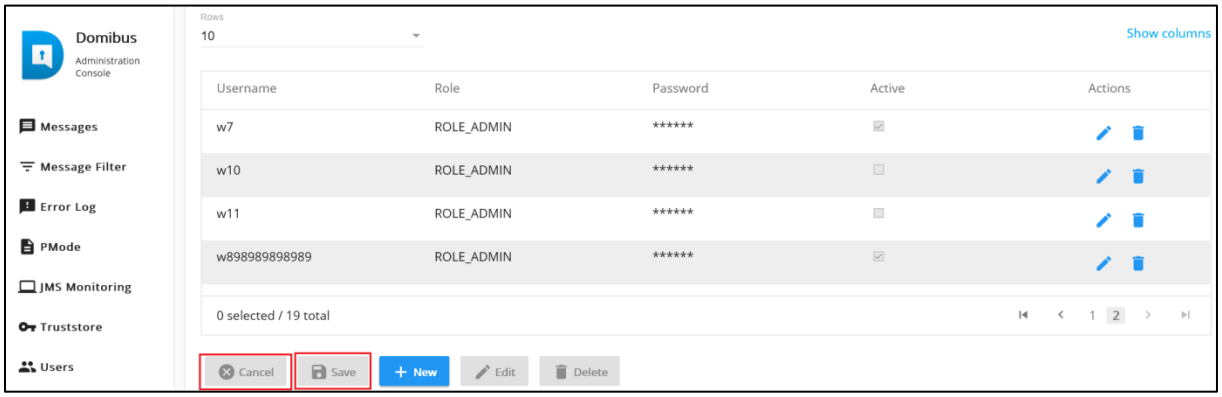
4. Confirm the password:



5. Click on **OK**:

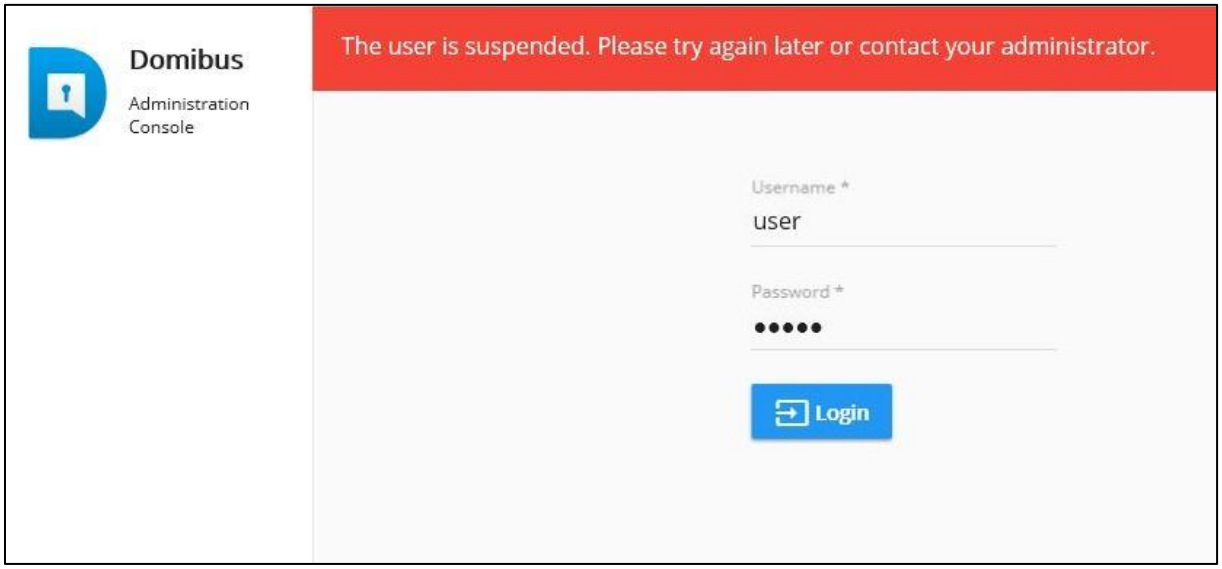


6. When done, either click on **Save,** to save the new password or **Cancel** to leave the password unchanged.



10.10.3. User Account Lockout Policy

A user account lockout policy has been implemented on Domibus Admin Console. By default, if a user tries to log to the Admin Console with a wrong password 5 times in a row, his account will be suspended (locked):

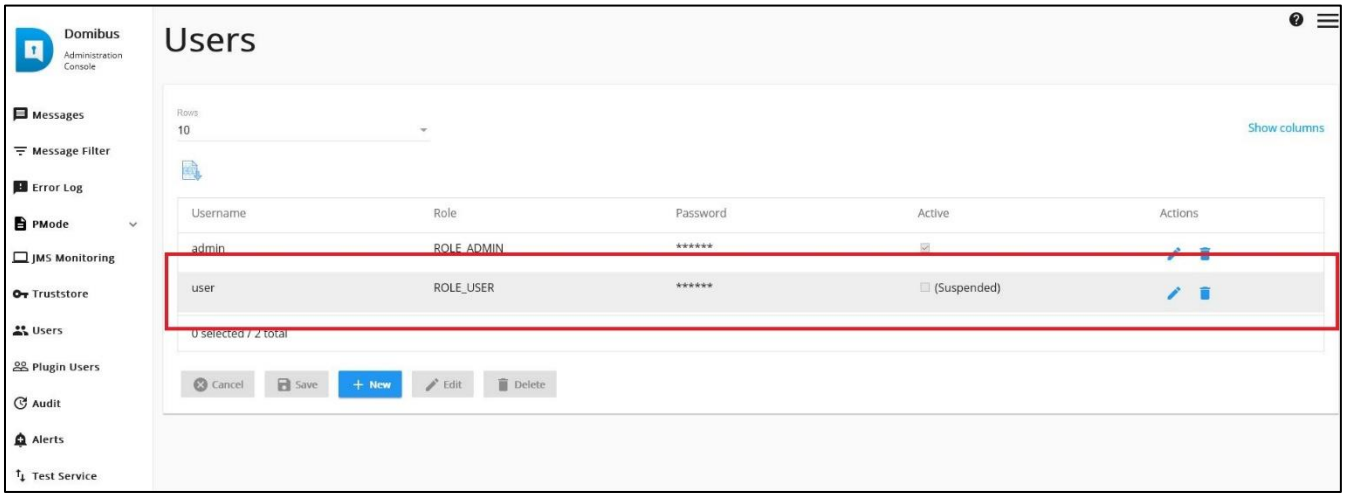


You can define in domibus.properties the number of failed attempts after which a user's account will be locked (see also §5.2- "Domibus Properties").

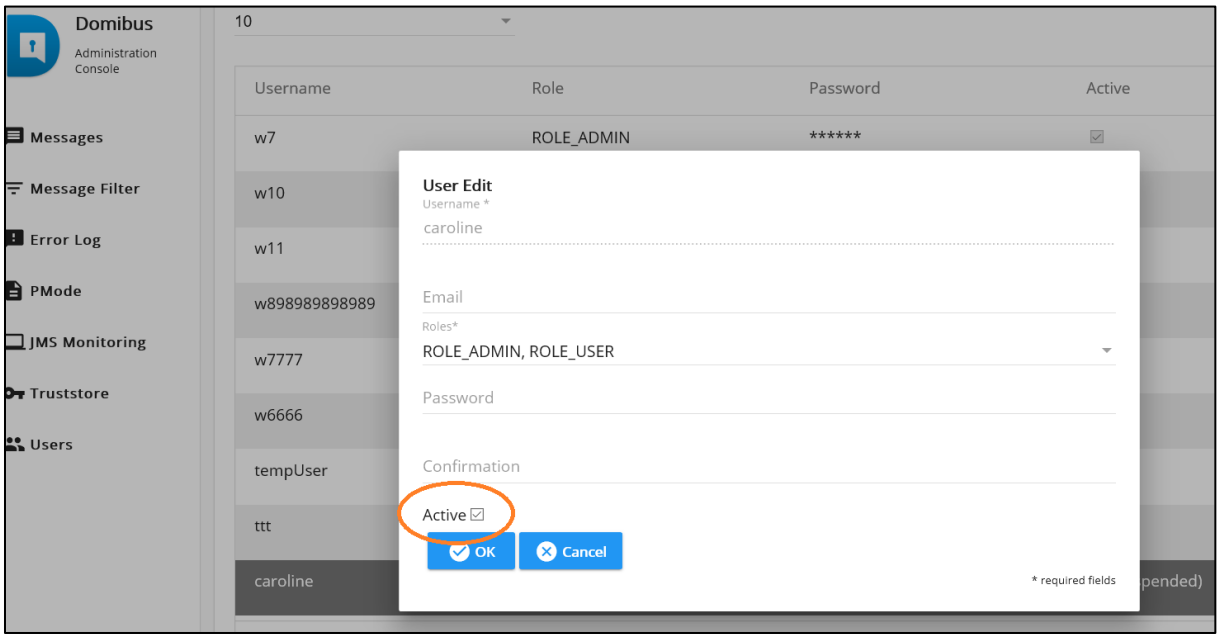
By default, a user remains suspended during one hour before his account is automatically unlocked and the user can try to log again.

If the user wants his account to be unlocked without waiting the default one hour, he can ask his administrator to unlock the account. To unlock the account, the administrator must change the user's status on the Admin Console from "Suspended" to "Active".

Select the suspended user and click on "Edit":



Re-activate the user (unlock it) by checking the "Active" status and confirming with OK:



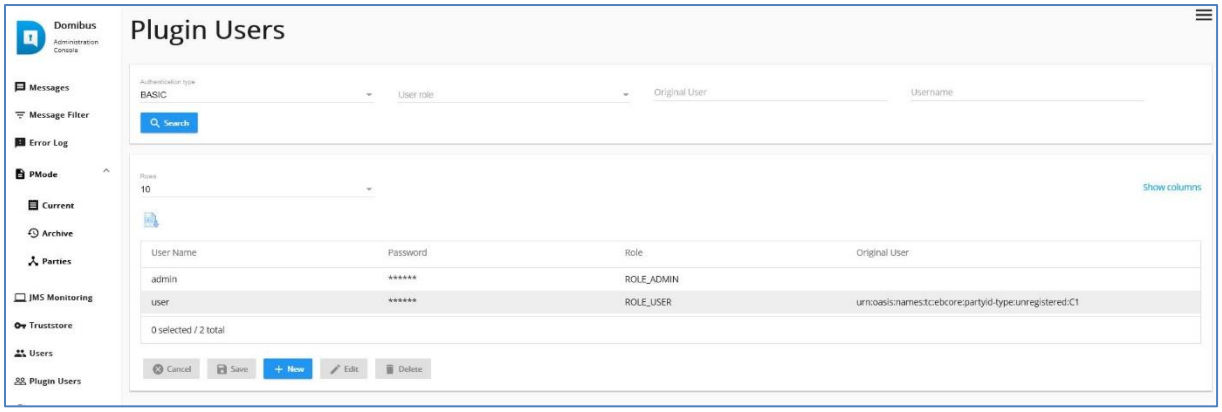
Do not forget to click on **Save** on the next window and then on **Yes** to confirm the change.

10.11. Plugin Users

In Multitenancy mode, the plugins security is activated by default, no matter if value configured in domibus.properties for the **domibus.auth.unsecureLoginAllowed** property.

This is needed to identify the request performed by the user and associate it to a specific domain. As a result, every request sent to Domibus needs to be authenticated.

A plugin must use a configured plugin user associated to a specific domain to authenticate every request sent to Domibus. The management of the plugin users is implemented in the **Plugin Users** page:



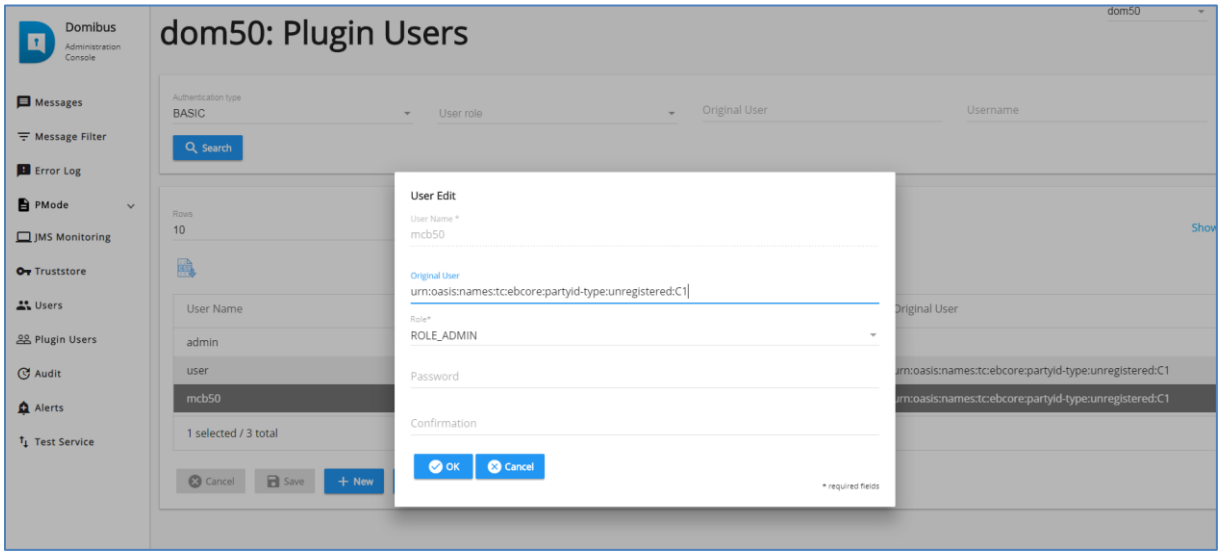
Plugin users are used to call Domibus ext API:

- ROLE_USER - can only call the APIs for messages that belong to this user (originalSender or finalRecipient match);
- ROLE_ADMIN - can call the APIs for any message.

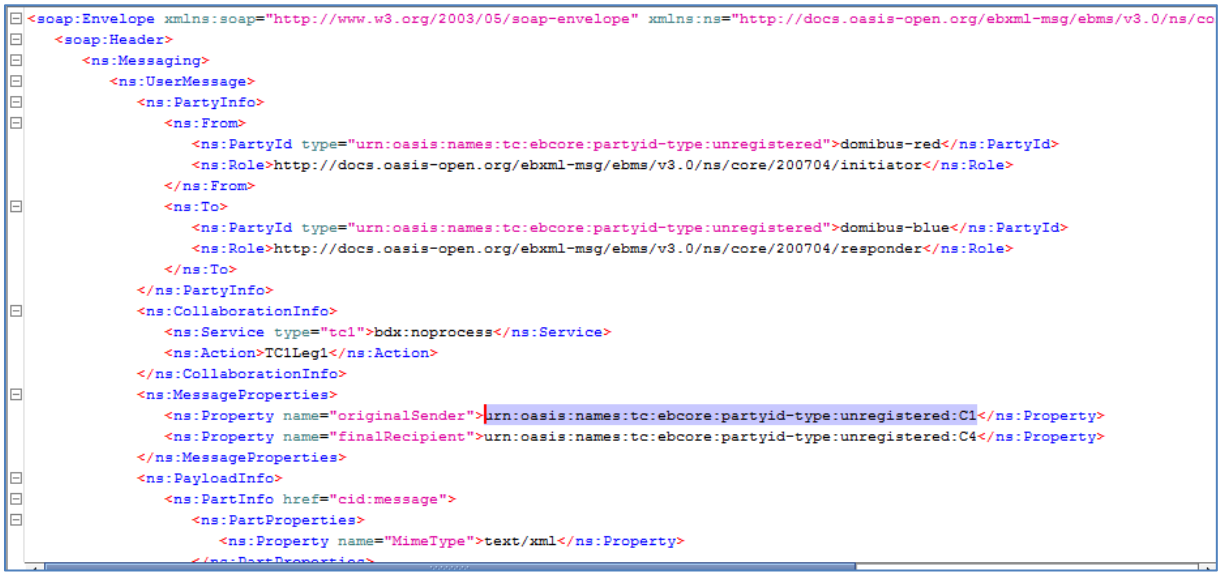
All plugin user passwords have an expiration period, configured in the domibus properties. The new password cannot be one of the last 5 used passwords (the number can be configured). Also, the password must meet complexity rules configured in the properties. If it does not meet them, then an error message is displayed (can also be configured).

The passwords of the default users expire in 1 day. This period can be configured.

The example below shows a **plugin user** that has been added:



Note that the Original user ID can be obtained from the **orginalSender** Property in **the SoapUI** project as shown here:



Do not forget to click on **Save** on the next window and then on **Yes** to confirm the change.

10.12. Audit

Audit support: Domibus keeps track of changes performed in the PMode, Parties, Message Filter and Users pages.

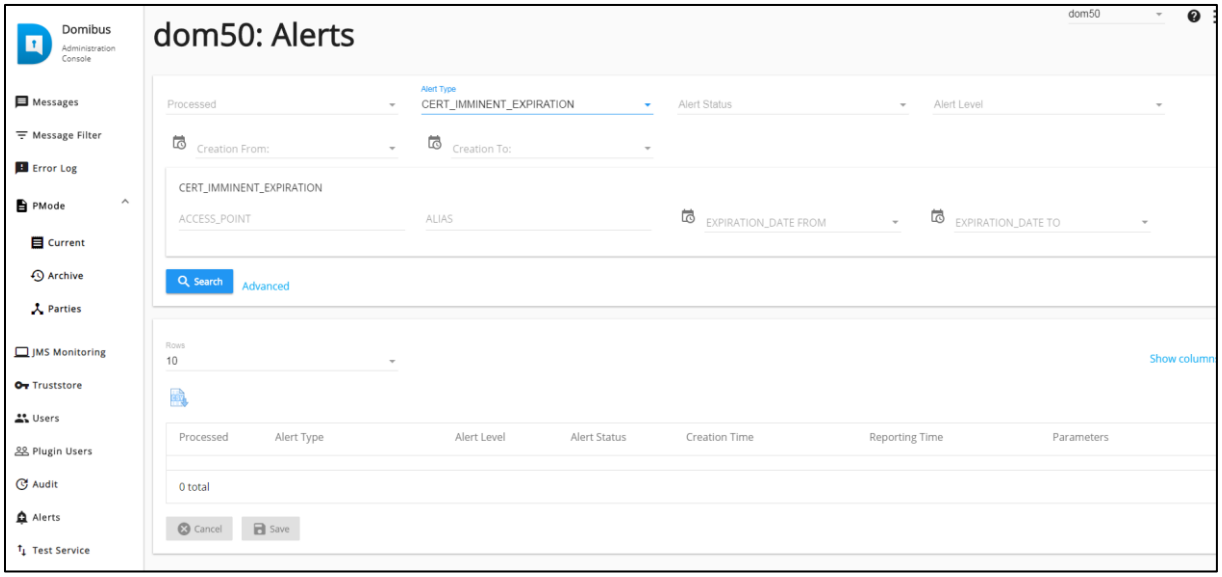
10.13. Alerts

Users can configure the alert feature as described in §20 – *"Alerts"*.
The purpose of the alert feature is to use different available media to notify the Domibus administrator in case of unusual behaviour. Currently alerts can be sent via mail.

The notification emails are sent to the destination recipient or recipients, configured in domibus properties. Also, for the alerts pertaining to the admin console users, the alerts are sent to the saved email address of the user to whom the notification is addressed.

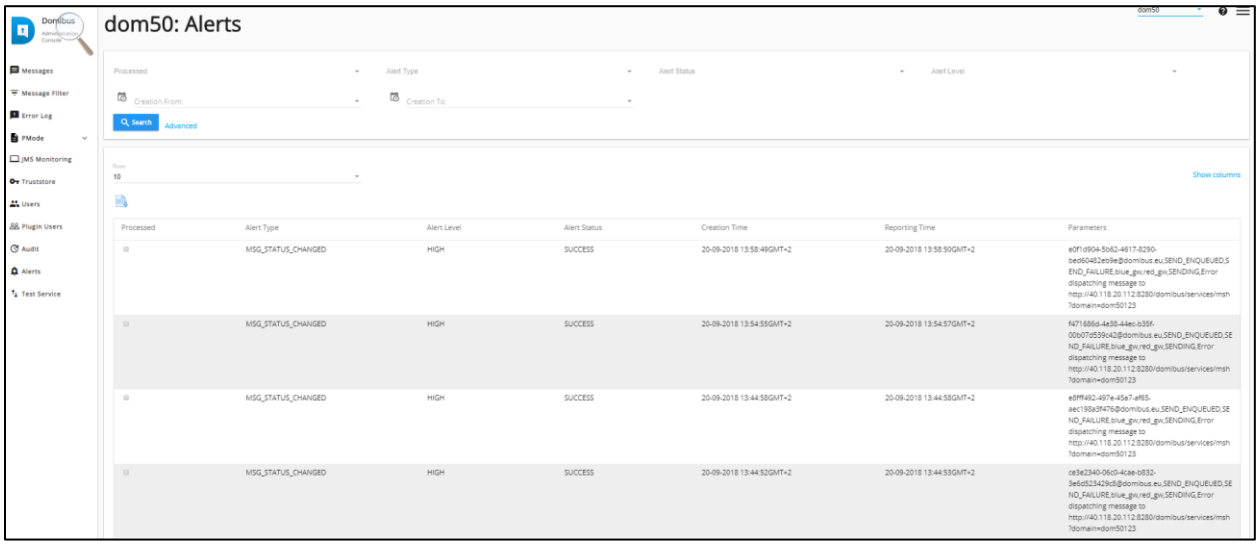
There are three types of alerts that can be configured: Message status change, Authentication issues and Certificate expiration.

Example: If the **CERT_IMMINENT_EXPIRATION** alert is selected, the following screen is presented:



The generated alerts can be checked in the **Alerts** page of the Administration console.

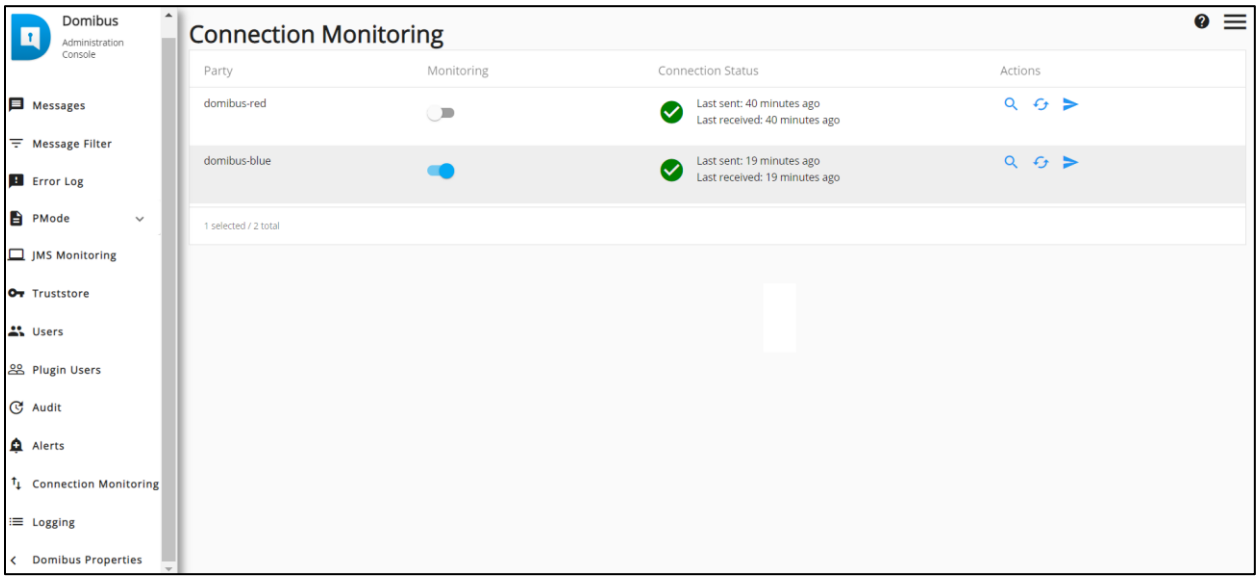
10.13.1. Example: Alerts on SEND_FAILURE



10.14. Connection Monitoring

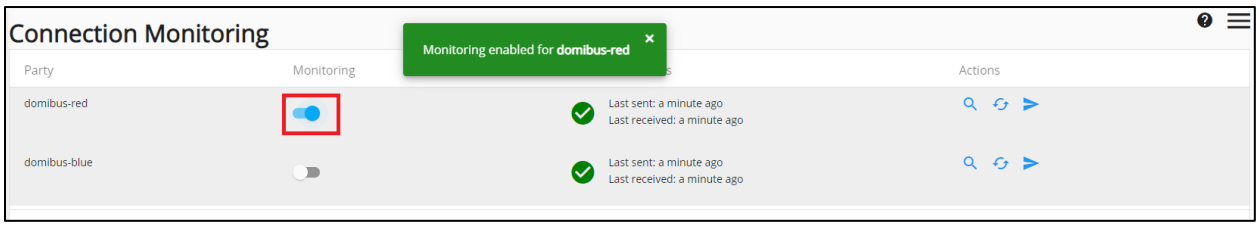
The **Connection Monitoring** section allows communication partners to perform a basic test of the communication configuration (including security at network, transport and message layer, and reliability) in any environment, including the production environment.

All parties that are defined in the Domibus properties are listed on the **Connection Monitoring** page of the Administration console, as shown below.



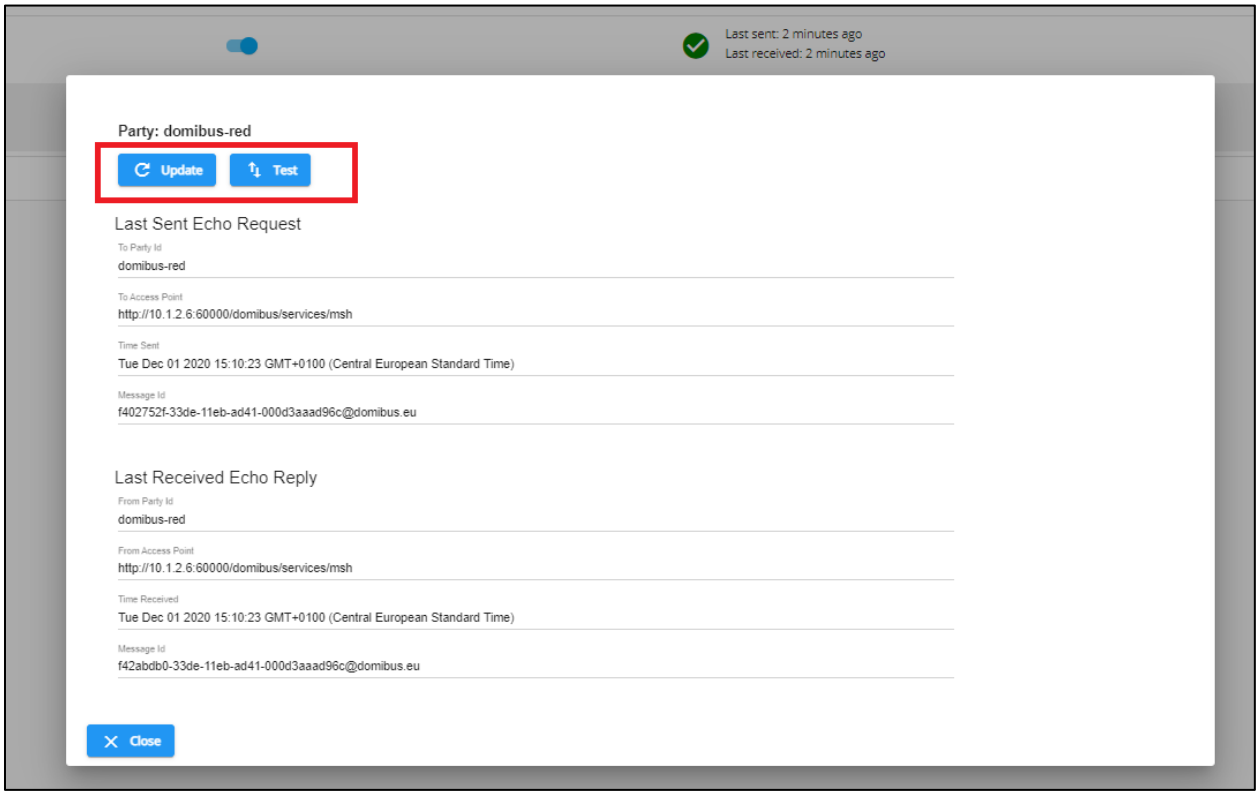
The user can activate or deactivate the monitoring feature by clicking on the Monitoring button of the desired party. Once activated, the monitoring service will send a test message on a frequency defined in the 'domibus.monitoring.connection.cron' property of the domibus.properties file (see §5.2 – "Domibus Properties").

The user can also activate or deactivate the monitoring of parties in the 'domibus.monitoring.connection.party.enabled' property of the domibus.properties file (see § 5.2 – "Domibus Properties").



The user can manually trigger a test by clicking on the Arrow under **Actions.**

To see the details of the connection that was tested, the user can click on the magnifying glass under **Actions**:

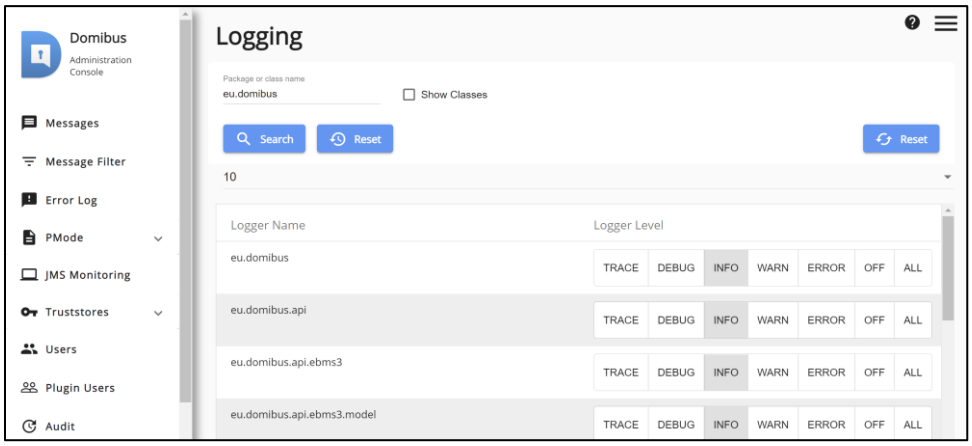


Clicking on **Test** will launch a connection test manually and clicking on **Update** will refresh the connection test information.

The connection monitoring feature also updates the status of each party in the database if it changes. This status can then be used by the **smart retry** feature. If smart retry is activated for a certain party, then if the retry scheduler needs to execute an attempt it will do so only if that party is reachable.

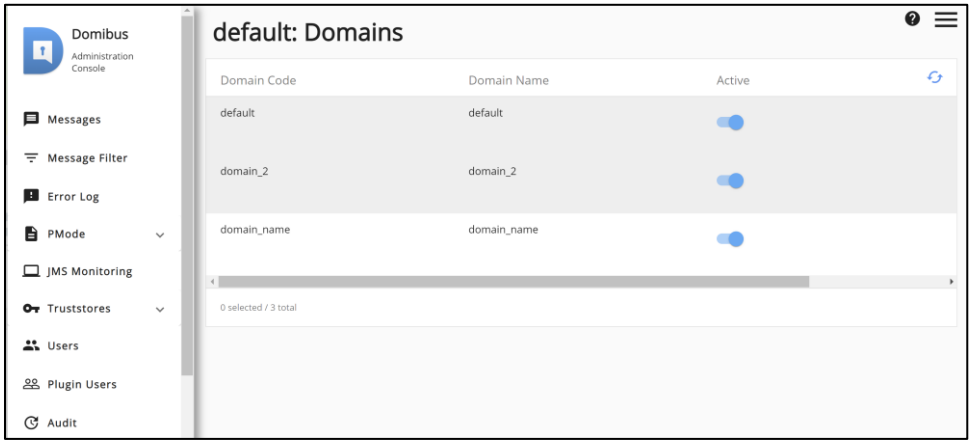
10.15. Logging

In the Logging section of the Administration Console, the list of all packages logging levels are displayed and can also be modified or reset.



10.16. Domains

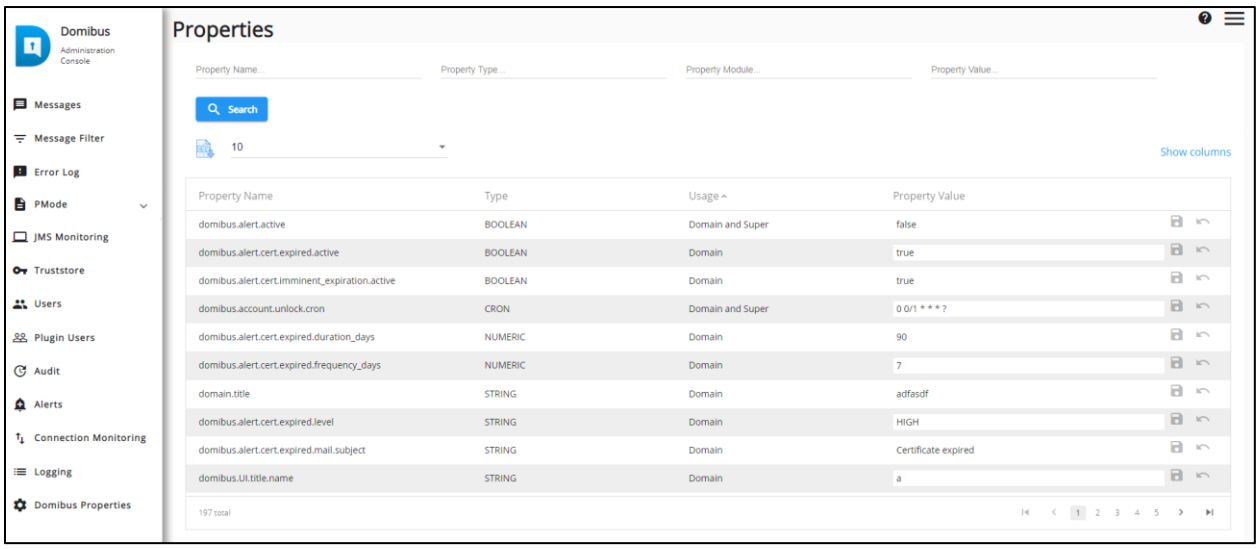
In the Domains section of the Administration Console, the list of all available domains is displayed and you can activate or deactivate a domain at runtime.



10.17. Properties

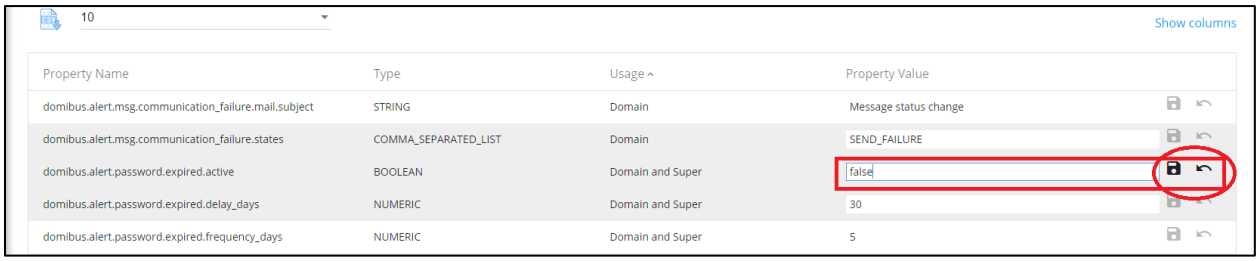
In the Properties section of the Administration Console, the list of all Domibus Properties are displayed (details on Domibus properties can be found in §5.2 - Domibus Properties). Some of the displayed properties can be edited, others are read-only.

Remark: When the Domibus server(s) is(are) restarted, the Domibus properties are reverted back and changes made via the Administration Console are lost. This feature is useful when a user wants to test a change in a Domibus property at runtime.



To change a Domibus property, the user clicks in the **Property Value** field and edits it (if the property is read-only, the user will not be able to edit that field). Once done, the user clicks on the **Save** icon to save the changes.

To revert the changes, the user can click on the **Back** arrow next to the Save icon: The back-arrow is only active while editing a specific field, and only restores the property to the value it had at the moment of starting editing, but not to the initial value in the domibus.properties file.



11. LARGE FILES SUPPORT

Domibus supports transfers between Access Points of files up to 2 GB using Java 8. In order to compute the message signature, Domibus loads the whole message into memory using a byte array. In Java, byte arrays can hold a maximum of 2 GB hence the Domibus limitation of 2 GB.

In order to optimize the sending of such large files, HTTP chunking is activated by default in the connection with the receiver Access Points. As chunked encoding is useful when sending larger amounts of data but decreases the performance on smaller amounts, Domibus uses a threshold to activate the chunking when appropriate only.

The following properties are used to configure chunking: domibus.dispatcher.allowChunking and domibus.dispatcher.chunkingThreshold. For more information about these properties, please refer to §5.2- “Domibus Properties”.

11.1. Split and Join

Support for large files bigger than 2 GB is supported using the Split and Join feature. It provides a mechanism for allowing a Sending MSH to split a large MIME-enveloped SOAP message, referred to as the source message, into a set of smaller MIME-enveloped SOAP messages, referred to as fragment messages, which MUST be joined at the Receiving MSH side. The resulting target message is an identical copy of the source message. The feature also supports compression.

The Split and Join feature is implemented according to the ebMS3 Part 2 “Large Message Splitting and Joining” [EBMS3P2], profiled and adjusted for use with eDelivery AS4.

Split and Join is currently supported in Domibus only in Tomcat in combination with the File System Plugin.

However custom plugins can use the Plugin API to send and receive messages using Split and Join. There are specific constraints, such as including long running operations in a JTA transaction which need to be taken into account.

The Split and Join feature is only supported for push mode, not for pull mode.

In order to activate the usage of Split and Join the leg configuration used by Domibus must have a splitting attribute configured as shown below:

```
.....
<splittingConfigurations>
  <splitting name="default"
        fragmentSize="500"
        compression="true"
        joinInterval="1440"/>
</splittingConfigurations>

<legConfigurations>
  <!--
  Please add the attribute "splitting"(pointing to a splitting configuration)
  to a specific leg in case you want to activate splitAndJoin feature
  -->
  <legConfiguration name="pushTestcase1tc1Action"
      service="testService1"
      action="tc1Action"
      splitting="default"
.....
```

Split and Join is used to send large files and therefore to handle this type of files, Domibus uses the file system to store the result of the intermediary operations needed to split and join the files. Therefore Domibus needs up to 4 times the size of payload in file disk space.

If a payloadProfile attribute is set for the legConfiguration used for Split & Join, the maxSize attribute of this profile should have the value increased from maxSize=”2147483647” to maxSize="9223372036854775807” otherwise Domibus is not able to send payloads over 2Gb.

12. Data Archiving

12.1. What's archiving?

Data archiving consists of moving messages that have been processed successfully or unsuccessfully by the access point to an external storage location for long-term retention.

Archived data consists of older data that have been processed at the communication level by the access points that are still significant to the business and may be needed for future reference. They may also be retained for legal constraints.

Data archives are indexed and searchable to allow easy retrieval.

It is not recommended to use Domibus as an archiving solution. Nevertheless, if the data really needs to be stored for long periods, then it is possible to set the Data Retention Policy to allow it to be extracted from the database through the webservices or through an external archiving tool.

12.2. Data Retention Policy

A data retention policy is a procedure established by the business for continuous information storage for operational, legal or compliance reasons.

The data retention policy needs to be defined based on the business needs and constraints.

In Domibus, the data retention policy can be found in the PMode file:

```
<mpcs>
    <mpc name="defaultMpc"
         qualifiedName="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultMPC"
         enabled="true"
         default="true"
         retention_downloaded="0"
         retention_undownloaded="14400"
         retention_sent="-1"
         retention_metadata_offset="2880"
         delete_message_metadata="true"
         max_batch_delete="1000"/>
</mpcs>
```

In the above extract of the sample PMode configuration of Domibus, the retention_undownloaded attribute set to **14400 minutes** (10 days). The delete_message_metadata is enabled (set to 'true') and the retention_metadata_offset attribute is set to 2880 minutes (2 days). This means that if the message is not downloaded, its payload will be deleted after 10 days, and only the metadata containing the information of the receiver and the acknowledgement will be retained for another 2 days. After those 2 days, the metadata will also be deleted.

The data retention policy is set by default to **0 minutes** for messages in status DOWNLOADED (the **retention_downloaded** attribute). This means that the message will be instantaneously deleted as soon as it is downloaded.

The **retention_undownloaded** attribute defaults to **-1**, this meaning messages with status RECEIVED will never be deleted.

These two parameters, retention_downloaded abd retention_undownloaded, can therefore be modified to meet the needs of the business.

The **retention_sent** parameter is optional and refers to messages in status ACKNOWLEDED and SEND_FAILURE. This parameter has the default value **-1** meaning the messages in these statuses should never be deleted.

The **delete_message_metadata** parameter is also optional. By default, it is set to false. When true, the entire message (including its metadata) is deleted.

When **delete_message_metadata** parameter is set to 'true', messages are deleted in bulk. The batch size of a bulk delete is defined by the optional field max_batch_delete. Defaults to 1000 (the max limit in Oracle).

12.3. Data Extraction

In order to keep the metadata and the payload of the message for a longer period than the one set, in the PMode, it is recommended to extract it to an external storage. As long as the retention worker does not delete it, data can be extracted through the webservices or through an external archiving tool.

For more information, please refer to the Data Model provided in the "Domibus Software Architecture Document" that can be found on the Digital single web portal [REF6].

13. DATABASE PARTITIONING

Partitioning allows tables, indexes, and index-organized tables to be **subdivided into smaller pieces**, enabling these database objects to be managed and accessed at a finer level of granularity.

Domibus may be configured to use partitions on Oracle database. It uses partitions by range for the main table and partitions by reference for the other tables.

Domibus partitions are created based on the format of the primary key and the granularity is of one hour.

Date and hour prefixed key format: **YYMMDDHH<10digits_increment>**

Example:

220329130000000001 where 22032913is the datehour prefix and 0000000001 is the 10 digits sequence increment.

13.1. Configure partitions – Oracle

Open a command line session, log in and execute the following commands:

```
sqlplus sys as sysdba   (password should be the one assigned during the Oracle installation )
==========================================================================
Once logged in Oracle:
GRANT CREATE JOB TO <edelivery_user>;


CONNECT <edelivery_user>
SHOW USER; (should return: edelivery_user)
@oracle-x.y.z- partitioning.ddl

EXIT
==========================================================================
```

Remarks:

1. Replace <edelivery_user> with corresponding value.

2. DDL/SQL scripts must be run with the @ sign from the location of the scripts

When the partitioning sql script is ran, it creates **one-hour** partitions for 7 days in advance. It also creates an oracle job “GENERATE_PARTITIONS_JOB” that runs once every day. This job is responsible to create new partitions for the 8th day, to assure continuity.

This job must be closely **monitored** to make sure partitions are created successfully.

Remark: Partitioning is not yet implemented for MySql.

13.2. Partitioning anexisting 5.0 database that is not partitioned

Download domibus-msh-distribution-X.Y.Z-sql-scripts.zip (see §3.1–*"Binaries repository"* for the download location) and unzip it to a convenient location.

Make sure your [edelivery_user] has the following grants:

GRANT REDEFINE ANY TABLE TO [edelivery_user];

GRANT CREATE MATERIALIZED VIEW TO [edelivery_user];

GRANT EXECUTE ON DBMS_REDEFINITION TO [edelivery_user];

GRANT SELECT ON USER_CONSTRAINTS TO [edelivery_user];

Connect to your database using the [edelivery_user] user execute oracle-5.0-partitioning-populated-table.ddl to create the stored procedures required for partitioning:

@<path_to_the_unzipped_scripts>/oracle-5.0-partitioning-populated-table.ddl

Enable output and run PARTITION_USER_MESSAGE passing as parameter the name of the schema where TB_USER_MESSAGE is, then turn off server output.

For example:

SET SERVEROUTPUT ON;

EXECUTE PARTITION_USER_MESSAGE('DOMIBUS');

SET SERVEROUTPUT OFF;

Then execute oracle-5.0-partition-detail-tables.sql and oracle-5.0-create-partitions-job.sql to partition by reference the detail tables and to create the partitioning job:

@<path_to_the_unzipped_scripts>/oracle-5.0-partition-detail-tables.sql

@<path_to_the_unzipped_scripts>/oracle-5.0-create-partitions-job.sql

13.3. Data retention with partitions

With partitions, a new **retention mechanism** is in place for Domibus. It is possible to configure Domibus to delete messages by dropping an entire partition, once all messages on a specific partition have expired.

On conf/domibus/domibus.properties, following property is set:

```
domibus.retentionWorker.deletion.strategy= PARTITIONS
```

The retention mechanism is defined by the retention values configured in the PMode. It computes the maximum retention period higher than -1 for all message statuses (received, downloaded, sent or failed) and evaluates all partitions older than this value. This increases the chances that each partition is only verified once before being dropped

For each partition, once all messages on a partition have expired, the partition is dropped (all messages are deleted at once). If the retention value for a status is -1, then we drop that partition only if there are no messages in that status on that partition. If the retention value is sest to -1, or no custom value is set, will not take full advantage of the database partitioning. It is advisable to set a value, so taht messages are not kept forever, or to use a different retention strategy. See the chapter on Data Retention Policy for details about the default values of the retention policy attributes. For example, the mpc configuration below will keep downloaded messages for one hour, undownloaded message for 1 day, and sent messages for 10 days:

```
<mpcs>
    <mpc name="defaultMpc"
         qualifiedName="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultMPC"
         enabled="true"
         default="true"
         retention_downloaded="60"
         retention_undownloaded="1440"
         retention_sent="14400" />
</mpcs>
```

There is a direct dependency between the archiving mechanism and the retention mechanism. When archiving is enabled, retention will not delete messages unless they were previously archived. For one partition, the retention mechanism checks that all messages are both expired and archived before droping the partition.

13.4. Partitions alerts

Following the logic described in §13.3 Data retention with partitions, all messages on a partition that is verified for expiration should already be in the **final** state. If some messages are not in the final state, **an alert is triggered**. The frequency of the alert may be configured in conf/domibus/domibus.properties and by default is 1 (one alert per day).

```
# ------------------------------- Alert management: Partitions ----------------
#Frequency in days between alerts.
domibus.alert.partition.expiration.frequency_days=1
```

14. eARCHIVING

The amount of data that will be exported by Domibus can be quite large. Therefore, Domibus will export the data to be archived in a shared file system.

Domibus will use an exporting mechanism, which can be configured in the Domibus property file, to continuously export data in batches in a preconfigured directory. The frequency of the export and the size of each batch can be configured according to the business needs (see §14.2-Sanitizer export, under 'Exported data').

On demand, there will be a possibility to re-export a previously exported batch using a REST API.

After each successful batch export, Domibus will notify the eArchiving client that new data can be archived. The eArchiving client will read the exported batch data from the file system. Afterward it will notify Domibus about the outcome of the archived batch, whether it has been successful or not.

14.1. Continuous export

The continuous export is a Domibus mechanism that can be configured to periodically export messages into a shared folder using a specific archival export format (see §14.2-Sanitizer export, "Exported Format").

Only the messages that reached the final state and were not previously exported by the continuous export procedure will be exported.

The continuous export will be configured to export messages starting from a specific date in the past. This will also allow the possibility to restart the periodic continuous export from a specific past date. This scenario could occur when there was a general issue in the export mechanism and the process must be reinitialized. By default, the continuous export start date is 01/01/1970, meaning that it will consider all messages. If this value is changed to a more recent date, all the messages older than this date will not be exported and therefore not deleted by the retention policy mechanism (see §14.4 - Retention policy).

The continuous export start date advances even if all the messages from a specific period are not in a final state and are not exported. In such scenario, a fallback mechanism called Sanitizer export will pick up the remaining messages. Domibus will check periodically if the continuous export start date does not advance within a specified amount of time configured in Domibus properties file. A Domibus alert will be send in such cases (see §20-Alerts) and a manual action must be taken to investigate why the old messages did not reach a final state.

As Domibus is handling messages reliably, it is possible for messages to be in a non-final state while recovering a failure to be sent. In this situation, the normal process of archiving will not select such messages. It will be possible to configure Domibus to filter the messages taking into account for archiving by setting a property in Domibus properties (either describing the default time retry timeout of Domibus, or defining the MPCs of the PMode). Those values are rounded at the hour mark. For example, for a runtime at 15h12:

- if domibus.earchive.batch.retry.timeout=5, the archiving job will not consider messages sent after 15h00.

- if domibus.earchive.batch.retry.timeout=30, the archiving job will not consider messages sent after 14h00.

The continuous export only looks forward, any issue with failed or expired export batches will have to be dealt with using the REST-API manual exports.

Domibus notifies the eArchiving client using a callback method via a REST endpoint when a batch export is completed or failed (see §14.5.6 - Notification from the archiving client that it has successfully archived or failed to archive a specific batch). The eArchiving client can start processing the batch after it has received the signal. In case Domibus fails to deliver a notification to the eArchiving client, it will re-attempt to deliver the notification later until the maximum number of attempts is reached. The maximum number of attempts and the delay time between the notifications will be configured in the Domibus properties file. If Domibus fails to notify the eArchiving client even after the maximum number of attempts is reached, it will send an alert (see §20 - Alerts) and a manual action must be taken.

The continuous export will export messages in batches having a maximum number of messages. The batch maximum number of messages is configured statically in the Domibus properties file. This means that changing the batch maximum number of messages requires a Domibus restart. Domibus might export empty batches which will assert that there are no messages eligible for export during a specific timespan.

14.2. Sanitizer export

The Sanitizer Export is a mechanism which exports messages which were not exported by the Continuous Export. It has been created to optimize the performance of the Sanitizer Export job.

The Continuous Export start date advances, for performance reasons, in case it encounters messages which are not in final state. The Sanitizer Export catches and exports the messages that are skipped by the Continuous Export mechanism.

If this sanitizer job finds a non-final message, an alert is sent.

If the start date of the continuous job is stopped, an alert is sent.

Some actions are required by the Domibus Admin and e-archiving service Admin, to make sure all messages are archived:

- Trigger changes for messages in non-final status:

  PUT /ext/monitoring/messages /enqueued/{messageId}/send

  This will re-send a message which has a SEND_ENQUEUED status.

- Move sanitizer start date in the past, so that messages are picked up and archived:

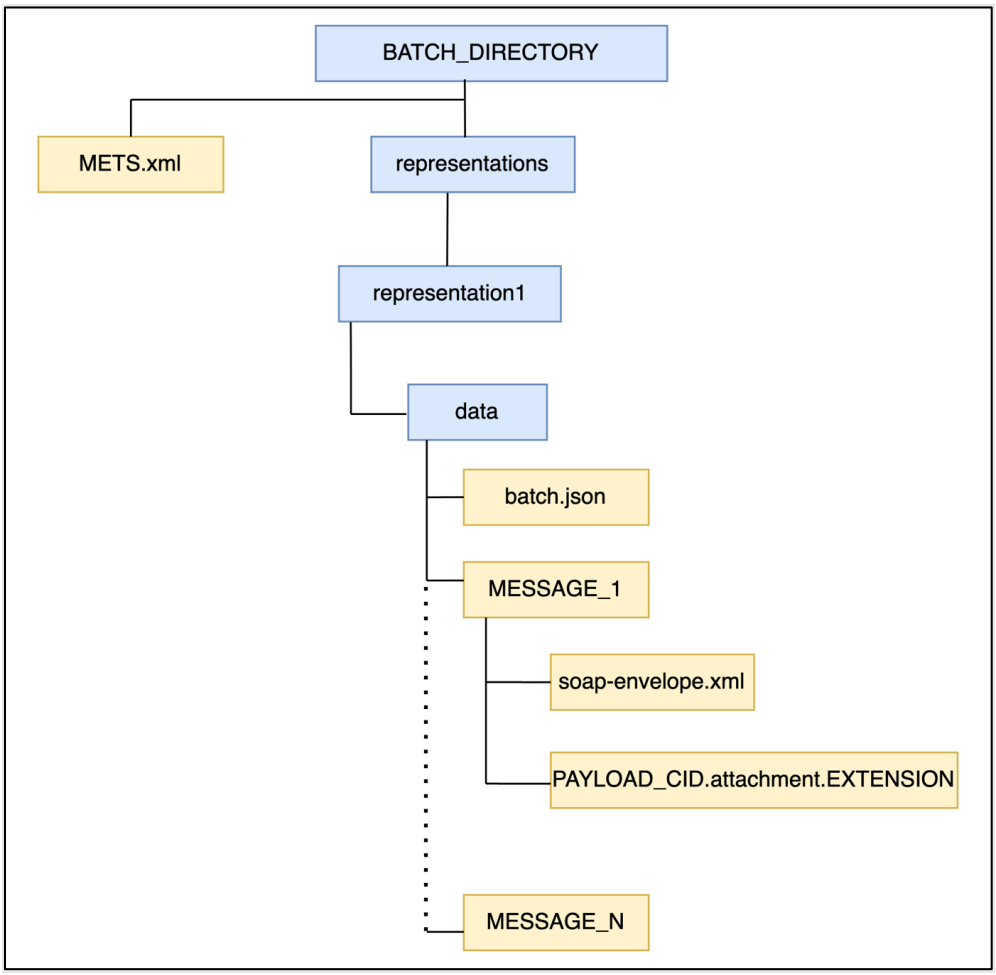
  PUT /ext/archive /sanity-mechanism/start-date

This will force the sanitizer to start at a given date provided by the user. All messages newer than this date will be considered for archiving if they are not already archived, not deleted and in a final successful state.

Exported Data

Domibus exports only messages that are not yet archived and that are in a final state: RECEIVED, DELETED, DOWNLOADED or ACKNOWLEDGED. Payloads are decompressed before being exported.

Messages will be exported in batches. For each exported batch, Domibus creates in a preconfigured shared file system a directory named based on an UUID. The structure of the batch is using EARK SIP format as illustrated below:



Where:

- **BATCH_DIRECTORY**
    - The directory in which Domibus exports messages contained in the batch. This directory is named based on a UUID.
- **METS.xml**
    - The batch manifest file. It contains:

- a list of all the exported files and their checksum for all the exported messages
- the batch id

Below is an example of a METS.xml file for a batch with id a46ab3d0-c710-4d73-b58d-e93e30b53a80. Please note that the example given below is not valid against the schema and it presents the most relevant elements of the METS.xml document.

The usage of the RODA library will be strongly considered to produce a correct METS.xml file. The EARK version 1 will be used.

NOTE: The METS.xml file does not contain the batch.json checksum. This is to avoid circular dependency with the the batch.json file which already contains the checksum of the METS.xml file.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<mets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.loc.gov/METS/"
      xmlns:xlink="http://www.w3.org/1999/xlink" OBJID="a46ab3d0-c710-4d73-b58d-e93e30b53a80" TYPE="ERMS"
      xlink:CONTENTTYPESPECIFICATION="SMURFERMS" PROFILE="http://www.dasboard.eu/specifications/sip/v03/METS.xml"
      xsi:schemaLocation="http://www.loc.gov/METS/ schemas/mets.xsd" LABEL="root level METS file for an IP">
    <metsHdr CREATEDATE="2017-01-31T13:07:22.6970809+02:00" RECORDSTATUS="NEW"
             LASTMODDATE="2017-01-31T13:07:22.6970809+02:00">
        <agent ROLE="CREATOR" TYPE="OTHER" OTHERTYPE="SOFTWARE">
            <name>Domibus</name>
        </agent>
        <!-- Batch Id -->
        <metsDocumentID>a46ab3d0-c710-4d73-b58d-e93e30b53a80</metsDocumentID>
    </metsHdr>
    <fileSec ID="e2a80c69-1eb4-4c2a-a029-5d77bf53d325">
        <fileGrp ID="e4d9422e-4c26-493b-929b-580198c34055" USE="Files root">
            <fileGrp ID="b6f4b954-9bd4-4179-93d3-37732b90923d" USE="data">
                <file ID="fec0430c-9152-4662-86fe-f9e78dad9baf" MIMETYPE="application/ octet-stream"
                      CREATED="2017-01-31T13:07:22.7470810+02:00"

>
                    <FLocat LOCTYPE="URL" xlink:type="simple" xlink:href="file:data/batch.json"/>
                </file>
            </fileGrp>
            <!-- message 1 files -->
            <!-- The message 1 files are grouped in a fileGrp with an ID that uses the AS4 UserMessage ID -->
            <fileGrp ID="9a0c6088-70ac-43b1-ab57-2f9d1f0204b7" USE="data">
```

```
                <file ID="aec0430c-9152-4662-86fe-f9e78dad9baf" MIMETYPE="text/xml" SIZE="11717"
                      CREATED="2017-01-31T13:07:22.7470810+02:00"
                      CHECKSUM="0d71382407d6a13af515761a6e1abd0e8b0784dab73e1a52427aaa9dbc4f73a9"
                      CHECKSUMTYPE="SHA-256">
                    <FLocat LOCTYPE="URL" xlink:type="simple"
                            xlink:href="file:data/9a0c6088-70ac-43b1-ab57-2f9d1f0204b7/soap-envelope.xml"/>
                </file>
                <file ID="bec0430c-9152-4662-86fe-f9e78dad9baf" MIMETYPE="text/xml" SIZE="12717"
                      CREATED="2017-01-31T13:07:22.7470810+02:00"
                      CHECKSUM="1d71382407d6a13af515761a6e1abd0e8b0784dab73e1a52427aaa9dbc4f73a9"
                      CHECKSUMTYPE="SHA-256">
                    <FLocat LOCTYPE="URL" xlink:type="simple"
                            xlink:href="file:data/9a0c6088-70ac-43b1-ab57-2f9d1f0204b7/message.attachment.xml"/>
                </file>
                <file ID="cec0430c-9152-4662-86fe-f9e78dad9baf" MIMETYPE="application/pdf" SIZE="42717"
                      CREATED="2017-01-31T13:07:22.7470810+02:00"
                      CHECKSUM="2d71382407d6a13af515761a6e1abd0e8b0784dab73e1a52427aaa9dbc4f73a9"
                      CHECKSUMTYPE="SHA-256">
                    <FLocat LOCTYPE="URL" xlink:type="simple"
                            xlink:href="file:data/9a0c6088-70ac-43b1-ab57-2f9d1f0204b7/invoice.attachment.pdf"/>
                </file>
            </fileGrp>
            <!-- .........-->

            <!-- message n files -->
            <fileGrp ID="0a0c6088-70ac-43b1-ab57-2f9d1f0204b7" USE="data">
                <!-- message n files -->
            </fileGrp>
        </fileGrp>
    </fileSec>
</mets>
```

- **batch.json**

    - A JSON file containing metadata related to the batch such as:

        - The version of data format exported
        - Batch id
        - The request id that triggered the creation of the batch. For a manual request, multiple batches can be created following a request export. For a continuous export the request id is always empty.
        - Request type: continuous or manualThe status of the batch export: success
        - Error code of the error in case of failure. The error codes will be defined at a later stage.
        - Error description of the error in case of failure
        - A timestamp of the batch export
        - The time period of the messages included in the batch: message start date and message end date. Adding the period in the batch.json might introduce a performance penalty. This will be further analysed during the implementation and in case it is degrading the performance, the message start date and message end date could be removed.
        - A checksum of the batch manifest METS.xml file
        - The list of exported message ids

Example of a batch.json file:

```
{
  "version" : "1",  ````
  "batchIid": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "requestTtype": "continuous",
  "status": "failed",
  "errorCode": "DOM_001",
  "errorDescription": "Failed to export batch",
  "timestamp": "2021-06-25T12:00:00Z",
  "messageStartId": "2021-01-25T12:00:00Z",
  "messageEndId": "2021-01-26T12:00:00Z",
  "manifestChecksum":
"sha256:01ba4719c80b6fe911b091a7c05124b64eeece964e09c058ef8f9805daca546b",
  "messages": [
    "123c6088-70ac-43b1-ab57-2f9d1f0204b7",
    "567c6088-70ac-43b1-ab57-2f9d1f0204b7"
  ]
}
```

- **MESSAGE_1 … MESSAGE_N**

    - The directory in which Domibus exports the files of a UserMessage. The directory is named after the exported message id and it contains the following files:

        - soap-envelope.xml

- Contains the Soap Envelope as it was exchanged between C2 and C3

- All the message payloads

  - Each payload is named based the payload CID from the AS4 message. The extension is derived based on the payload mime type.

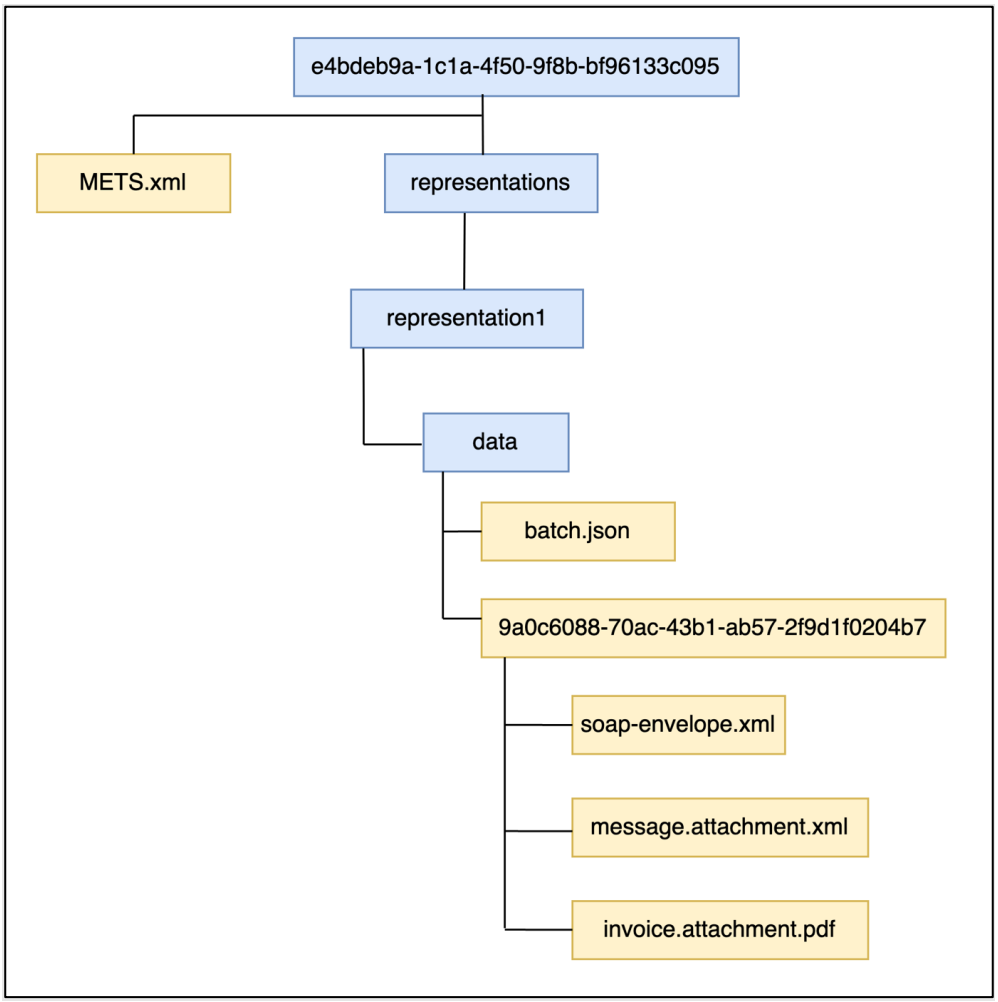
  NOTE: The payloads exported in the batches will be decompressed by Domibus in case they are stored internally as compressed.

  Example:

  For a payload with CID value "**message**" and the mime type "**text/xml**", the payload file is named **message**.attachment.**xml**

  - The payloads will be exported as they were exchanged between C2 and C3

Example of the structure of folder for a batch with id **e4bdeb9a-1c1a-4f50-9f8b-bf96133c095** containing one exported message with id **9a0c6088-70ac-43b1-ab57-2f9d1f0204b7** which has two payloads with CID **message** and **invoice**:



14.3. Audit

Relevant actions related to the archiving mechanism will be audited in the Domibus log files. Examples of relevant actions:

- Archiving client requests a manual export
- A batch is exported: the content of the complete batch.json file will be logged.
- Notifications sent from Domibus to the eArchiving client and vice-versa

Domibus audits the following events in the logs:
(Code: Text template)

- BUS-083: Enqueue continuous batch [{}].
- BUS-084: Archiving client requests a manual (re-)export for batch [{}].
- BUS-085: A batch [{}] is exported to file path: {}!
- BUS-086: Export Notification for batch [{}] is sent from Domibus to the eArchiving client!
- BUS-087: Received Archive Notification for batch: [{}] with message: [{}] from the eArchiving client to Domibus!
- BUS-088: Received Archive Failed notification for batch: [{}] with message: [{}] from the eArchiving client to Domibus!
- BUS-089: Export failed batch: [{}]. Error message: [{}]!
- BUS-090: Batch: [{}] with first [{}] and last message: [{}] is Archived.

14.4. Retention policy

When an archiving client is integrated and configured, the Domibus retention policy will consider a message for deletion only if the archival client has successfully archived it. This is the case even if the message is expired according to the retention policy configuration from the PMode.

Domibus will define a retention policy for the exported batches. The retention policy value will be configured in the Domibus property file, and it will have a default value of 1 month. If a batch is not archived during this time, it is considered as expired and it will be eligible for deletion.

A batch can also be eligible for deletion if the archiving client notifies Domibus that it has successfully archived it or it has failed to archive it. A failed batch can always be re-exported on demand using the REST API (see § 14.5- eArchiving interface).

14.5. eArchiving interface

The integration between Domibus and the archiving client will be done using REST APIs and a shared file system for exporting batch data.

In this section we will describe in detail the REST API that must be implemented by each system.

The Open API document for eArchiveClient is part of the Domibus distribution artefacts (see [REF1]).

14.5.1. Security

From a **security perspective**, it is RECOMMENDED that the communication between Domibus and the archiving client is performed over HTTPS.

The Domibus REST API is protected with basic authentication. The Domibus Admin Guide will contain an example of an HTTP request using basic authentication once it will be updated to cover the eArchiving feature.

A Plugin User must be created upfront in the Domibus Administration Console and used by the archiving client which MUST supply the basic authentication headers on each Domibus REST API call.

Domibus will be able to call the callback archiving REST API interface with or without basic authentication headers. This will be configured statically in the Domibus property file.

Domibus will expose the following REST API to be used by an archiving client. Please check the section §14.5- eArchiving interface for the meaning of the fields from the REST responses.

14.5.2. Get batch by batch ID

This REST endpoint will fetch any batch from any status given its batch ID.

HTTP method: GET

Parameters:

- batchId: bacth id of the batch

```
curl -X 'GET' 'http://172.70.1.5:8080/domibus/ext/archive/batches?batchId=
3950092f-5805-11ec-8197-9c5c8ec0f1ad' \
  -H 'accept: application/json'
```

Response: HTTP 200 status with body:

```
{
  "batchId": "3950092f-5805-11ec-8197-9c5c8ec0f1ad",
  "requestType": "CONTINUOUS",
  "status": "EXPORTED",
  "errorCode": null,
  "errorDescription": null,
  "enqueuedTimestamp": "2021-12-08T09:00:00.000+0000",
  "messageStartDate": 21120609,
  "messageEndDate": 21120609,
  "manifestChecksum":
"sha256:939c282837187d32196a80070b33901ee4f77db41de46fbb2c12449f74b29de6",
  "messages": []
}
```

List batch export requests that are queued (continuous and manual)

This REST endpoint will export the list of batches that are queued to be processed asynchronously by Domibus. It can be used for monitoring purposes.

HTTP method: GET

Parameters:

- lastCountRequests: return last N enqueued batch export requests - if this parameter is given all others filters are ignored.
- requestTypes: return batches for given batch types (Values: CONTINUOUS, MANUAL)
- startDate: start day-time of batches enqueued
- endDate: end day-time of batches enqueued
- pageStart: the offset from which the message IDs export will start
- pageSize: maximum number of records in the page

```
curl -X 'GET' 'http://172.70.1.5:8080/domibus/ext/archive/batches/queued?requestType=CONTINUOUS&startDate=2021-12-06T00%3A00%3A00Z&endDate=2021-12-07T00%3A00%3A00Z&pageStart=0&pageSize=100' \
  -H 'accept: application/json'
```

Response example:

```
{
  "filter": {
    "lastCountRequests": 0,
    "requestTypes": [
      "CONTINUOUS"
    ],
    "startDate": "2021-12-06T00:00:00Z",
    "endDate": "2021-12-07T00:00:00Z"
  },
  "pagination": {
    "pageStart": 0,
    "pageSize": 100,
    "total": 1
  },
  "queuedBatches": [
    {
      "batchId": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
      "requestType": "CONTINUOUS",
      "enqueuedTimestamp": "2021-12-07T11:36:21.726Z",
      "messageStartDate": 21120100,
      "messageEndDate": 21120700,
      "messages": ["123c6088-70ac-43b1-ab57-2f9d1f0204b7", "567c6088-70ac-43b1-ab57-2f9d1f0204b7"]
    }
  ]
}
```

where:

- **enqueuedTimestamp** is the timestamp when Domibus adds the batch to the queue

14.5.3. Get the messageId exported in a batch

This REST endpoint provides the message IDs exported in a batch. All message IDs are exported if the limit and start parameters are not provided.

HTTP method: GET

Parameters:

batchId: batch id of the message ids,

pageStart: the offset from which the message IDs export will start
pageSize: maximum number of records in the pageRequest example

```
curl -X  'GET' \
 http://172.70.1.5:8080/domibus/ext/archive/batches/exported/123c6088-70ac-43b1-ab57-2f9d1f0204b7/messages?pageStart=0&pageSize=100' \
  -H 'accept: application/json'
```

Response example:

```
{
  "batchId": "123c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "pagination": {
    "pageStart": 0,
    "pageSize": 5,
    "total": 1236
  },
  "messages": [
    "123c6088-70ac-43b1-ab57-2f9d1f0204b7",        "567c6088-70ac-43b1-ab57-2f9d1f0204b7","143c6088-70ac-43b1-ab57-2f9d1f0204b7",   "153c6088-70ac-43b1-ab57-2f9d1f0204b7", "163c6088-70ac-43b1-ab57-2f9d1f0204b7"
  ]
}
```

where:

- **total** is the total number of message IDs contained in the batch

14.5.4. History of exported batches

This REST endpoint provides a history of exported batches with status success, failed or expired. It allows the archiving client to validate if it has archived all exported batches.

HTTP method: GET

Parameters:

- messageStartDate: start date and hour of the exported messages in the batch yyMMddHH
- messageEndDate: end date of the exported messages included in the batch
- statuses: filter by list of batch statues
- includeReExportedBatches: batch re-export status (true/false; includes batches for which a re-export has been requested using the REST endpoint)
- pageStart: the offset/page from which the message IDs export will start. List is sorted by batch request date
- pageSize: maximum number of records in the pageRequest example:

```
curl -X 'GET' \
'http://172.70.1.5:8080/domibus/ext/archive/batches/exported?messageStartDate=21100100&messageEndDate=21123100&statuses=EXPORTED&reExport=false&pageStart=0&pageSize=100' \
  -H 'accept: application/json' \
```

Response example:

```
{
```

```
  "pagination": {
    "pageStart": 0,
    "pageSize": 100,
    "total": 10
  },
  "filter": {
    "messageStartDate": 21100100,
    "messageEndDate": 21123100,
    "statuses": [
      "EXPORTED"
    ],
    "includeReExportedBatches": false
  },
  "exportedBatches": [
    {
      "batchId": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
      "requestType": "CONTINUOUS",
      "status": "EXPORTED",
      "enqueuedTimestamp": "2021-12-07T12:20:20Z",
      "messageStartDate": 21100100,
      "messageEndDate": 21100102,
      "manifestChecksum":
"sha256:01ba4719c80b6fe911b091a7c05124b64eeece964e09c058ef8f9805daca546b",
      "messages": [
        "123c6088-70ac-43b1-ab57-2f9d1f0204b7",
       "567c6088-70ac-43b1-ab57-2f9d1f0204b7"
      ]
    }
  ]
}
```

14.5.5. Request to export a batch based on batch id

This REST endpoint will export a new batch with a new batch id containing the same messages that were already exported in a batch identified by the batch id provided as a parameter. The batch id identifying the previously exported batch will not be automatically deleted or modified in the database or on the disk storage. The retention mechanism can potentially delete it later (see §14.5.13-Receive notification when an expired batch has been deleted).

This endpoint can be used in cases where the export or archival of a batch has failed or it expired as well as for other unexpected situations.

The request contains a batch id that has been extracted, for instance, from the history of batch requests (see §14.5.4-History of exported batches).

HTTP method: PUT

Parameters: batch id

Request example

```
curl -X 'PUT' \
  'http://172.70.1.5:8080/domibus/ext/archive/batches/9a0c6088-70ac-43b1-ab57-
2f9d1f0204b7/export' \
  -H 'accept: application/json'
```

Response example:

```
{
  "batchId": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "status":                                  "                              EXPORTED",
  "timestamp":                                               "2021-06-25T12:00:00Z"
```

```
}
```

Response example with error :

```
{
  "batchId": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "status": "ERROR",
  "message": "Failed to request a manual export",
  "timestamp": "2021-06-25T12:00:00Z"
}
```

14.5.6. Notification from the archiving client that it has successfully archived or failed to archive a specific batch

This REST endpoint will be used by the archiving client to confirm that a batch was archived successfully or that it failed to archive it. The request contains the batch identifier which allows Domibus to identify all messages in the batch to mark them as archived and eligible for purging.

Note that, for performance reasons, Domibus will asynchronously mark the batch messages as archived.

Therefore, this REST endpoint only confirms to the client that it has acknowledged the notification and it does not mean that the batch messages are already marked as archived.

HTTP method: PUT

Parameters:

- batchId: batch id
- status: sets final batch status: ARCHIVED – batch was successfully archived, ARCHIVE_FAILED: client failed to archive exported batch
- message: set message – reason for failed batch

Request example

```
curl -X 'PUT' \
  'http://172.70.1.5:8080/domibus/ext/archive/batches/exported/9a0c6088-70ac-43b1-ab57-2f9d1f0204b7/close?status=ARCHIVED' \
  -H 'accept: application/json'
```

Response example

```
{
  "batchId": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "status": "ARCHIVED",
}
```

14.5.7. Get messages which were not archived within a specific period

This REST endpoint can be used to check if all AS4 messages received or sent within a specific period were archived.

The response will contain the list of the message IDs which were not archived during the specified period.

HTTP method: GET

Parameters:

- messageStartDate: Message start date of the period to be checked
- messageEndDate: Message end date of the period to be checked.
- pageStart: The offset/page of the result list.
- pageSize: Maximum number of returned records/page size.
- Request example:

```
curl -X 'GET' \
  'http://172.70.1.5:8080/domibus/ext/archive/messages/not-archived?messageStartDate=2021-10-01T00%3A00%3A00Z&messageEndDate=2021-12-31T00%3A00%3A00Z&pageStart=0&pageSize=100' \
  -H 'accept: application/json' \
```

Response example:

```
{
  "pagination": {
    "pageStart": 0,
    "pageSize": 100,
    "total": 5
  },
  "messages": [
    ["123c6088-70ac-43b1-ab57-2f9d1f0204b7",        "567c6088-70ac-43b1-ab57-2f9d1f0204b7","143c6088-70ac-43b1-ab57-2f9d1f0204b7",   "153c6088-70ac-43b1-ab57-2f9d1f0204b7", "163c6088-70ac-43b1-ab57-2f9d1f0204b7"]
  ]
}
```

where:

- **total** is the total number of message IDs contained in the batch

14.5.8. Get the current start date of the continuous export

This REST endpoint will expose the continuous export mechanism current start date (see section **Error! Reference source not found.**).

HTTP method: GET

Parameters: none

Request example

```
curl -X 'GET' \
  'http://172.70.1.5:8080/domibus/ext/archive/continuous-mechanism/start-date' \
  -H 'accept: application/json' \
```

Response example: 21120100

14.5.9. Set the current start date of the continuous export

This REST endpoint forces the continuous archiving process to start at a given date provided by the user. All messages older than this date will be consider for archiving if they are not already archived, not deleted and in a final state.

HTTP method: PUT

Parameters:

- MessageStartDate: Start date and hour. The value is 8 digit number with format yyMMddHH!

Request example

```
curl -X 'PUT' \
  'http://172.70.1.5:8080/domibus/ext/archive/continuous-mechanism/start-date?messageStartDate=21100100'
```

14.5.10. Get the current start date of the sanity export

This REST endpoint will expose the sanity export mechanism current start date.

HTTP method: GET

Parameters: none

Request example

```
curl -X 'GET' \
  'http://172.70.1.5:8080/domibus/ext/archive/sanity-mechanism/start-datehttp://172.70.1.5:8080/domibus/ext/archive/sanity-mechanism/start-date' \
  -H 'accept: application/json' \
```

Response example: 21120100

14.5.11. Set the current start date of the sanity export

This REST endpoint forces the sanity archiving process to start at a given date provided by the user. All messages older than this date will be consider for archiving if they are not already archived, not deleted and in a final state.

HTTP method: PUT

Parameters:

- MessageStartDate: Start date and hour. The value is 8 digit number with format yyMMddHH!

Request example

```
curl -X 'PUT' \
  'http://172.70.1.5:8080/domibus/ext/archive/sanity-mechanism/start-date?messageStartDate=21100100' \
```

14.5.12. Receive notification when a batch has been exported in the shared folder

Domibus notifies the archiving client when a batch has been exported in the shared folder. The notification is performed for a successful and for a failed export.

Request:

- REST endpoint example: /domibus/archive/batches/{batch_id:.+}/export-notification
- HTTP method: PUT
- Request examples:

```
{
  "batchId": "e7c99242-5362-11ec-b6f6-0242ac460105",
  "requestType": "CONTINUOUS",
  "status": "EXPORTED",
  "timestamp": "2021-12-02T11:28:00Z",
  "messageStartDate": 21100100,
  "messageEndDate": 21100102,
  "messages": [
    "ea69b73d-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eabbf5f0-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eafaaca3-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eb38ee26-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eb744979-4f74-11ec-9039-0242ac460105@domibus.eu"
  ]
}
```

```
  "batchId": "e7c99242-5362-11ec-b6f6-0242ac460105",
  "requestType": "CONTINUOUS",
  "status": "FAILED",
  "timestamp": "2021-12-02T11:28:00Z",
  "messageStartDate": 21100100,
  "messageEndDate": 21100102,
  "messages": [
    "ea69b73d-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eabbf5f0-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eafaaca3-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eb38ee26-4f74-11ec-9039-0242ac460105@domibus.eu",
    "eb744979-4f74-11ec-9039-0242ac460105@domibus.eu"
  ],
  "errorCode": "BUS-089",
  "errorDescription": "Export failed batch: [e7c99242-5362-11ec-b6f6-0242ac460105].
Error message: Can not read payload!"
}
```

```
{
  "batch_id": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "request_type": "continuous",
  "status": "failed",
  "error_code": "DOM_001",
  "error_description": "Failed to export batch",
  "timestamp": "2021-06-25T12:00:00Z",
  "message_start_date": "2021-01-25T12:00:00Z",
  "message_end_date": "2021-01-26T12:00:00Z",
  "messages": [
```

```
    "123c6088-70ac-43b1-ab57-2f9d1f0204b7",
    "567c6088-70ac-43b1-ab57-2f9d1f0204b7"
  ]
}
```

Response: empty response with HTTP 200 status

14.5.13. Receive notification when an expired batch has been deleted

Domibus notifies the archiving client when it deletes an expired batch (see section **Error! Reference source not found.**).

Request:

- REST endpoint example: /domibus/archive/batches/{batch_id:.+}/stale-notification
- HTTP method: PUT
- Request example:

```
{
  "batch_id": "9a0c6088-70ac-43b1-ab57-2f9d1f0204b7",
  "request_type": "continuous",
  "status": "success",
  "timestamp": "2021-06-25T12:00:00Z",
  "message_start_date": "2021-01-25T12:00:00Z",
  "message_end_date": "2021-01-26T12:00:00Z",
  "messages": [
    "123c6088-70ac-43b1-ab57-2f9d1f0204b7",
    "567c6088-70ac-43b1-ab57-2f9d1f0204b7"
  ]
}
```

Response: empty response with HTTP 200 status

15. NON REPUDIATION

In order to guarantee non-repudiation, the sending Access Point (C2) stores the full **SignalMessage**, including the **MessageInfo,** the Receipt (that contains the **NonRepudiationInformation** for each part) and the signature of the receipt by the receiver Access Point (C3).

This will guarantee that the receiver Access Point (C3) cannot deny having received a message from the sender Access Point (C2) during the sending process. However, if the initial sender (C1) wants to be sure that the final recipient (C4) cannot deny having received a specific content inside this message, then the sender must be able to show the specific content that was used to produce the receiver Access Point (C3) signature.

Domibus, as a sending Access Point (C2), keeps track of the metadata of the sent messages but does not store the actual message payloads. Therefore it is recommended that the initial sender (C1) stores the message payloads safely for the time needed to guarantee non-repudiation of the sent messages.

In order to guarantee non-repudiation, the receiving Access Point (C3) stores the full UserMessage and the associated signature of the sender (C2).

This will guarantee that the sender Access Point (C2) cannot deny having sent a message to the receiver during the sending process. However, if the final recipient (C4) wants to be sure that the sender cannot deny having sent a specific content inside this message, then the final recipient (C4) must be able to show the specific content that was used to produce the sender Access Point signature (C2).

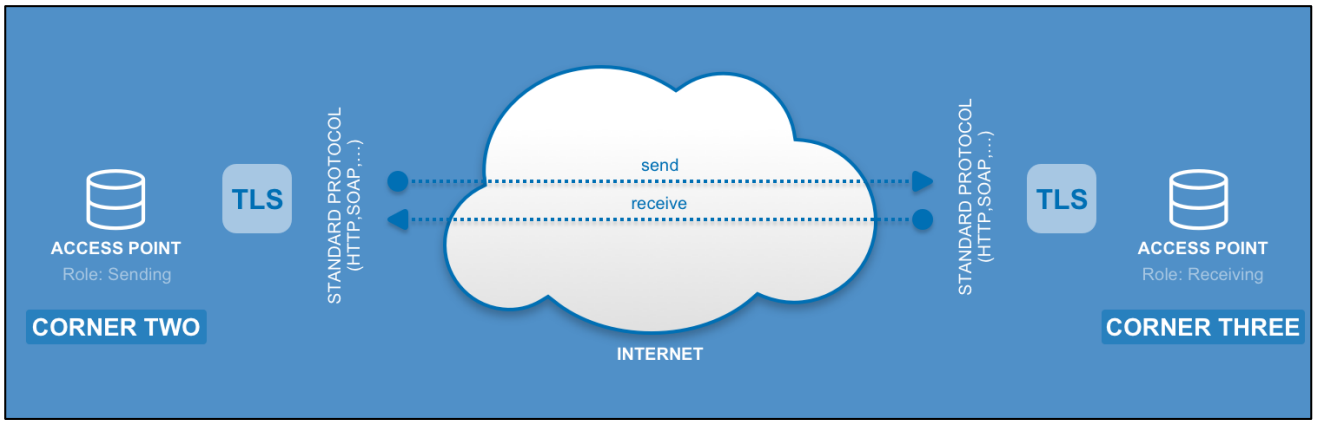
Domibus, as a receiving Access Point (C3), keeps track of the metadata of the received messages and will store the message payloads, only for the (limited) duration configured in the retention period (specified in the PMode). Therefore it is recommended that the final recipient (C4) either stores the message payloads safely or aligns the retention period on the receiving Access Point (C3) with the time needed to guarantee non-repudiation of the received messages.

16. TLS CONFIGURATION

16.1. TLS Configuration

16.1.1. *Transport Layer Security in Domibus*

In addition to the message level security, Domibus may be configured to exchange messages using TLS (HTTPS). The use of TLS is mandatory according to the eDelivery AS4 profile. However, you can choose to configure it in the Access Point itself or delegate it to another appropriate network component.



16.1.2. *Client Side Configuration*

The implementation of the Domibus MSH is based on the CXF framework. According to CXF documentation, when using an "https" URL, CXF will, by default, use the certs and keystores that are part of the JDK. For many HTTPs applications, that is enough and no configuration is necessary. However, when using custom client certificates or self-signed server certificates or similar, you may need to specifically configure in the keystores and trust managers and such to establish the SSL connection.

Apache provides full description of all possible configuration of the **tlsClientParameters** [see http://cxf.apache.org/docs/client-http-transport-including-ssl-support.html#ClientHTTPTransport(includingSSLsupport)-ConfiguringSSLSupport].

In Domibus, the TLS configuration is read from the file edelivery_*path*/**conf/domibus/clientauthentication.xml** and it is used as fallback when Domibus is configured in multi tenancy mode.

In multi tenancy mode, the file name is prefixed by the domain name and it is located in the edelivery_*path*/**conf/domibus/domains/domain_name** (f.i.: domain_name_clientauthentication.xml).

Below example presents two possible configurations, One-Way SSL and Two-Way SSL:

clientauthentication.xml – One-Way SSL:

```
<http-conf:tlsClientParameters disableCNCheck="true" secureSocketProtocol="TLS"
      xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
      xmlns:security="http://cxf.apache.org/configuration/security">

<security:trustManagers>
```

```
    <security:keyStore type="JKS" password="your_trustore_password"
                file="${domibus.config.location}/keystores/your_trustore_ssl.jks"/>
</security:trustManagers>
</http-conf:tlsClientParameters>
```

In One-Way SSL, the sender validates the signature of the receiver using the public certificate of the receiver, provided in *your_trustore_ssl*.jks.

clientauthentication.xml – Two-Way SSL:

```
<http-conf:tlsClientParameters disableCNCheck="true" secureSocketProtocol="TLS"
    xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
    xmlns:security="http://cxf.apache.org/configuration/security">
  <security:trustManagers>
    <security:keyStore type="JKS" password="your_trustore_password"
        file="${domibus.config.location}/keystores/your_trustore_ssl.jks"/>
  </security:trustManagers>
   <security:keyManagers keyPassword="your_keystore_password">
    <security:keyStore type="JKS" password="your_keystore_password"
        file="${domibus.config.location}/keystores/your_keystore_ssl.jks"/>
  </security:keyManagers>
</http-conf:tlsClientParameters>
```

In Two-Way SSL, both the sender and the receiver sign the request and validate the trust of the other party. In addition to the public certificate of the receiver (*your_trustore_ssl*.jks), the private certificate of the sender is also configured (*your_keystore_ssl*.jks).

Remark:

TLSv1.3 is preferred for eDelivery AS4 Profile, but TLSv1.2 is allowed.

When self-signed certificates are used, the CN check must be disabled: **disableCNCheck="true"**.

The attribute **disableCNCheck** specifies whether JSSE should omit checking if the host name specified in the URL matches the host name specified in the Common Name (CN) of the server certificate. The attribute is "false" by default and must not be set to "true" during production use (cf.**[REF7]**).

16.1.3. Server side configuration

16.1.3.1. Tomcat 9.x

In Server.xml, add a new connector with the **SSLEnabled** attribute set to "true":

```
<Connector SSLEnabled="true"
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
        scheme="https" secure="true"
        keystoreFile="${domibus.config.location}/keystores/your_keystore_ssl.jks"
keystorePass="your_keystore_password"
        clientAuth="false" sslProtocol="TLS" />
```

The keystore jks location and password must be specified, otherwise the default ones will be taken into account.

TLS version can also be specified.

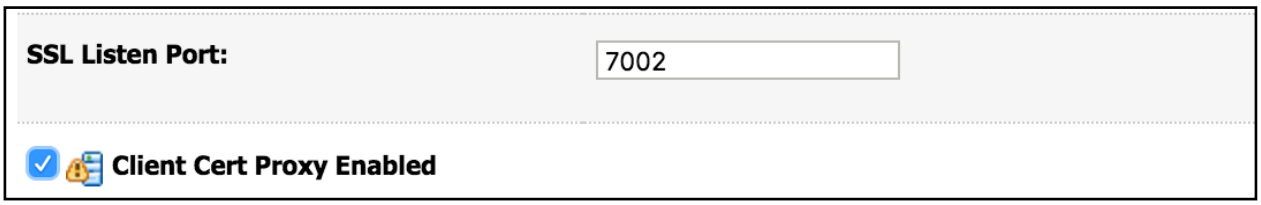
The above connector has **clientAuth="false"**, which means that only the server has to authenticate itself (One Way SSL). To configure "Two Way SSL", which is optional in the eDelivery *AS4* Profile, set **clientAuth="true"** in Server.xml and provide the location of the *your_truststore_ssl*.**jks** file so that the server can verify the client:

```
<Connector SSLEnabled="true"
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
        scheme="https" secure="true"
        keystoreFile="${domibus.config.location}/keystores/your_keystore_ssl.jks"
keystorePass="your_keystore_password"
        truststoreFile="${domibus.config.location}/keystores/your_truststore_ssl.jks"
truststorePass="your_trustore_password"
        clientAuth="true" sslProtocol="TLS" />
```

16.1.3.2. WebLogic

1. Specify the use of SSL on default port 7002:

Go to Servers ➔ select Server Name ➔ Configuration ➔ General then **click** on **Client Cert Proxy Enabled**:



2. Add keystore and truststore:

Go to Servers ➔ select Server Name ➔ Configuration ➔ Keystores and SSL tabs and use **Custom Identity and Custom Trust** then set keystore and truststore jks.

Disable basic authentication at WebLogic level:

By default WebLogic performs its own basic authentication checks before passing the request to Domibus. As we want basic authentication to be performed by Domibus, we need to disable it at the application server level.

To do so, in **DOMAIN_HOME/config/config.xml,** add the following highlighted section:

```
....

<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>

</security-configuration>
```

16.1.3.3. WildFly

The keystore JKS (e.g: bluek.jks) location and password must be specified in the **standalone-full.xml** file as follows.

In this setup only the server has to authenticate itself (One Way SSL).

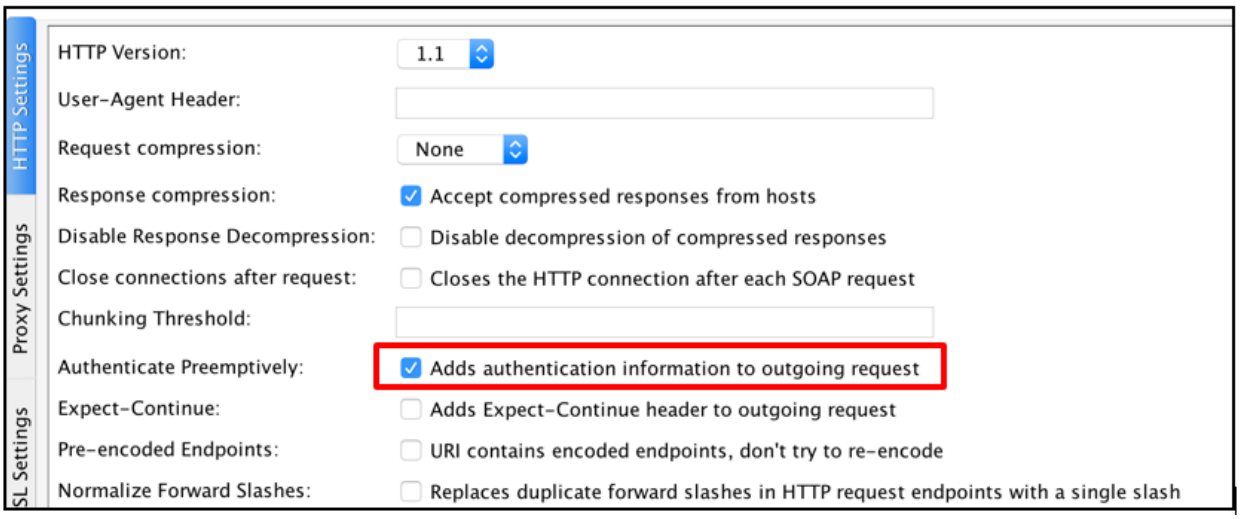
```
..
        </sasl>
        <tls>
          <key-stores>
            <key-store name="applicationKS">
              <credential-reference clear-text="JKS Password"/>
              <implementation type="JKS"/>
              <file path="keystores/gateway_keystore.jks" relative-to="jboss.server.config.dir"/>
            </key-store>
          </key-stores>
          <key-managers>
            <key-manager name="applicationKM" key-store="applicationKS" generate-self-signed-
certificate-host="localhost">
              <credential-reference clear-text="Private Key Password"/>
            </key-manager>
..
```

The "Two Way SSL", which is optional in the eDelivery *AS4* Profile is configured via the <server-ssl-contexts> option. For more information, please check the Wildlfy documentation.

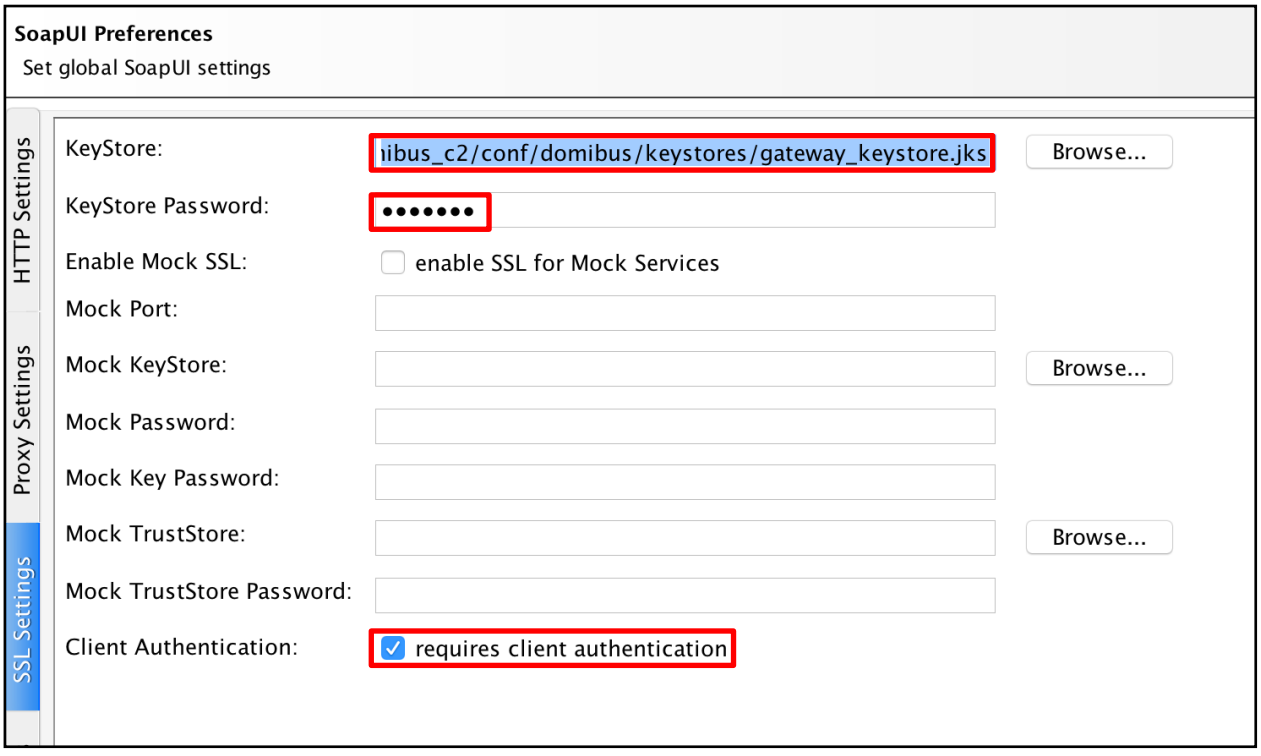
```
        ..
          </sasl>
        <tls>
          <key-stores>
            <key-store name="applicationKS">
              <credential-reference clear-text="JKS Password"/>
              <implementation type="JKS"/>
              <file path="keystores/gateway_keystore.jks" relative-to="jboss.server.config.dir"/>
            </key-store>
          </key-stores>
          <key-managers>
            <key-manager name="applicationKM" key-store="applicationKS" generate-self-signed-
certificate-host="localhost">
              <credential-reference clear-text="Private Key Password"/>
            </key-manager>
<server-ssl-contexts>
            <server-ssl-context name="applicationSSC" key-manager="applicationKM"/>
          </server-ssl-contexts>..
```

16.1.3.4. Configure Basic and Certificates authentication in SoapUI

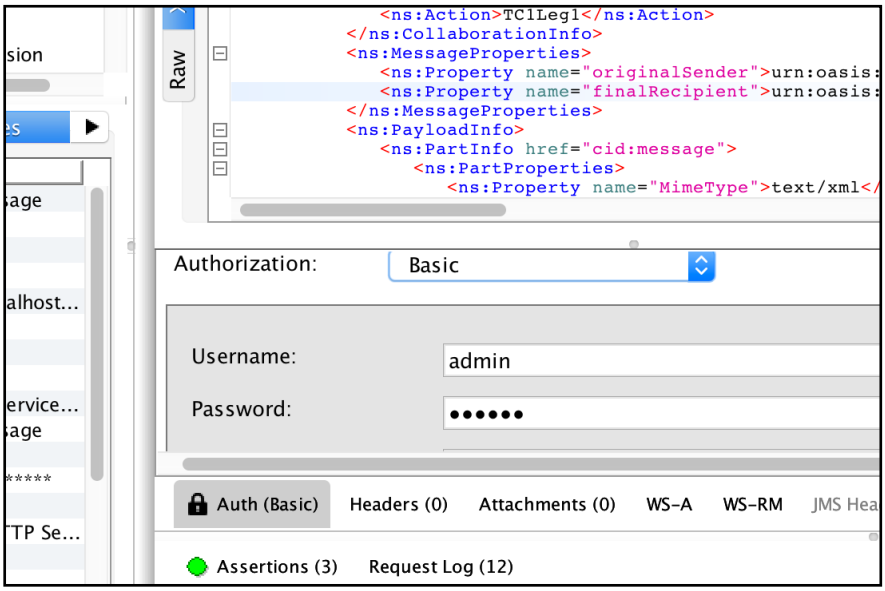
Go to File ➔ Preferences ➔ HTTP Settings and check the option **Adds authentication information to outgoing requests**:



Go to File ➔ Preferences ➔ SSL Settings, add the **KeyStore** and **KeyStore Password** and check the option **requires client authentication**:



To allow Basic Authentication, select the Auth tab, click Add New Authorization and select Basic. Enter user and password (e.g. Username = ***admin***; for the password, look in the logs for the phrase: “Default password for user admin is”):



16.1.3.5. PMode update

If you enable HTTPS, then your PMode Configuration Manager needs to make sure that all other endpoint PModes are modified accordingly.

With the SSL connector configured as above, the MSH endpoint is now: **https://***your_domibus_host***:*8443*/domibus/services/msh**.

After the updates, upload the PModes via the Admin Console:

Example:

```
<party name="party_id_name1"
endpoint=
"https:// party_id_name1_hostname:8443/domibus/services/msh">
```

17. DYNAMIC DISCOVERY OF UNKNOWN PARTICIPANTS

17.1. Overview

In a dynamic discovery setup, the sender and/or the receiver parties and their capabilities are not configured in advance.

The sending Access Point will dynamically retrieve the necessary information for setting up an interoperability process from the Service Metadata Publisher (SMP). The SMP stores the interoperability metadata which is a set of information about the recipient or end entity (its identifier, supported business documents and processes) and AP (metadata which includes technical configuration information about the receiving endpoint, such as the transport protocol and its address) cf.[REF8].

The receiving AP registers its metadata in the SMP and configures the PMode to be able to accept messages from trusted senders that are not previously configured in the PMode. The receiving AP will have to configure one process in its PMode for each SMP entry.

The mapping between the PMode process and the SMP entry is defined for PEPPOL in "§17.3 – *PMode configuration for PEPPOL"* and for OASIS in "§17.8 - *PMode configuration for OASIS*".

Please note that the sender does not have to be registered in the SMP and the receiver merely extracts its identifier from the received message.

The following sections describe how to configure Domibus AP to use Dynamic Discovery (§17.3 – *"PMode configuration for PEPPOL"*, §17.3.3 – "*Sender and Receiver PMode"*, §17.8 – "*PMode configuration for OASIS",* §17.9 – "*Policy and certificates for OASIS"*).

17.2. Domibus configuration for PEPPOL

To enable the integration with the SMP/SML components, Domibus requires some changes in the domibus.properties configuration file which include:

1. Adding the following properties to enable the usage of the PEPPOL dynamic discovery client:

   domibus.dynamicdiscovery.client.specification">PEPPOL

2. Setting the dynamic discovery client to use certificates to access the SMP. These certificates are different in TEST and PRODUCTION environments, therefore we need to specify the Mode used by the dynamic discovery client by setting the following property:

   domibus.dynamicdiscovery.peppolclient.mode">TEST

3. Setting the "**domibus.smlzone**" property.

17.3. PMode configuration for PEPPOL

17.3.1. *Sender PMode*

In a dynamic discovery process, the receiver of the messages is not known beforehand and therefore the **PMode.Responder** parameter SHOULD NOT be set.

The dynamic discovery process must include a leg which maps the configured entry (action, service and service type – see section §17.5 – "*Message format for PEPPOL*") of the Receiver in the SMP.

The security policy to be used in the leg is the policy that embeds the Binary Security Token into the security header (see section §5.1.1 – *"Security Policies"* for more information):

```
security="eDeliveryAS4Policy_BST"
```

Sample Sender PMODE configuration extract:

```
...
<services>
    <service name="testService1"
        value="urn:www.cenbii.eu:profile:bii05:ver2.0"
        type="cenbii-procid-ubl"/>
</services>
<actions>
    <action name="tc1Action"
        value=" busdox-docid-qns:: urn:oasis:names:specification:ubl:schema:xsd:CreditNote-
2::CreditNote##urn:www.cenbii...."/>
</actions>
<securities>
    <security name="eDeliveryAS4Policy_BST"
        policy="eDeliveryAS4Policy_BST.xml"
        signatureMethod="RSA_SHA256"/>
</securities>
<legConfigurations>
    <legConfiguration name="pushTestcase1tc1Action"
    service="testService1"
    action="tc1Action"
    defaultMpc="defaultMpc"
    reliability="AS4Reliability"
    security="eDeliveryAS4Policy_BST"
    receptionAwareness="receptionAwareness"
    propertySet="eDeliveryPropertySet"
    payloadProfile="MessageProfile"
    errorHandling="demoErrorHandling"
    compressPayloads="true"/>
</legConfigurations>
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
```

```
    <initiatorParties>
        <initiatorParty name="senderalias"/>
    </initiatorParties>
    <!-- no responderParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
...
```

17.3.2. Receiver PMode

Dynamic discovery configuration of the receiver is similar to the configuration of the sender, except that the roles are swapped: the sender of the messages is not known beforehand. As a consequence the ***PMode.Initiator*** parameter SHOULD NOT be set.

```
...
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <responderParties>
        <responderParty name="receiveralias"/>
    </responderParties>
    <!-- no initiatorParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
...
```

17.3.3. Sender and Receiver PMode

Dynamic discovery configuration when the Access Point acts as both sender and receiver would look like these following lines:

```
...
<services>
    <service name="testService1"
        value="urn:www.cenbii.eu:profile:bii05:ver2.0"
        type="cenbii-procid-ubl"/>
</services>
<actions>
    <action name="tc1Action"
        value=" busdox-docid-qns:: urn:oasis:names:specification:ubl:schema:xsd:CreditNote-
2::CreditNote##urn:www.cenbii...."/>
</actions>
<securities>
    <security name="eDeliveryAS4Policy_BST"
        policy="eDeliveryAS4Policy_BST.xml"
```

```
        signatureMethod="RSA_SHA256"/>
</securities>
<legConfigurations>
    <legConfiguration name="pushTestcase1tc1Action"
    service="testService1"
    action="tc1Action"
    defaultMpc="defaultMpc"
    reliability="AS4Reliability"
    security="eDeliveryAS4Policy_BST"
    receptionAwareness="receptionAwareness"
    propertySet="eDeliveryPropertySet"
    payloadProfile="MessageProfile"
    errorHandling="demoErrorHandling"
    compressPayloads="true"/>
</legConfigurations>
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <initiatorParties>
        <initiatorParty name="senderalias"/>
    </initiatorParties>
    <!-- no responderParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
<process name="tc2Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <responderParties>
        <responderParty name="receiveralias"/>
    </responderParties>
    <!-- no initiatorParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
```

17.4. Policy and certificates for PEPPOL

The receiver must include the certificate of the trusted authority(ies) in its truststore. It will only accept messages that were signed with certificates issued by the trusted authority(ies) (cf. §29 – "Annex 1 - Usage of certificates in PEPPOL and OASIS" for more information).

17.5. Message format for PEPPOL

When dynamic discovery is used, the "to" field should not be statically configured in the PMode (the "to" field may even be omitted in the message). The lookup is performed by C2 based on the **finalRecipient** message property.

Note: In Peppol, the service@type has a fixed value while the service@value is made of ProcessIdentifier@Scheme::ProcessIdentifier

Example of a message using the **finalRecipient** for dynamic discovery:

```
<ns:UserMessage>
 <ns:PartyInfo>
  <ns:From>
   <ns:PartyId type="urn:fdc:peppol.eu:2017:identifiers:ap">senderalias</ns:PartyId>
   <ns:Role> http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
  </ns:From>
  <ns:To>
  </ns:To>
 </ns:PartyInfo>
 <ns:CollaborationInfo>
     <ns:Service type="cenbii-procid-ubl">urn:www.cenbii.eu:profile:bii05:ver2.0</ns:Service>
<ns:Action>busdox-docid-qns:: urn:oasis:names:specification:ubl:schema:xsd:CreditNote-
2::CreditNote##urn:www.cenbii.eu:transaction:biitrns014:ver2.0:extended:urn:www.peppol.eu:bis:pep
pol5a:ver2.0::2.1</ns:Action>
 </ns:CollaborationInfo>
  <ns:MessageProperties>
   <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</ns:Property>
    <ns:Property name="finalRecipient" type="iso6523-actorid-
upis">0007:9340033829test1</ns:Property>
  </ns:MessageProperties>
</ns:UserMessage>
```

17.6. SMP entry

The following table describes the mapping between the PMode static configuration and the dynamic SMP records structure:

SMP Endpoint registration record	PMode attributes
ServiceMetadata/ServiceInformation/ProcessIdentifier	PMode[1].BusinessInfo.Service
ServiceMetadata/ServiceInformation/DocumentIdentifier	Pmode[1].BusinessInfo.Action
ServiceInformation/Processlist/Process/ServiceEndpointList/Endpoint/EndpointReference/Address	Pmode[].Protocol.Address

Table 5 - SMP Entry Mapping

The Service Metadata Record also provides the receiving end certificate. This certificate can be used to encrypt the message to be sent to the receiver. The certificate can also provide the name of the Access Point for this PMode by using the Certificate CNAME as the PMode identifier (cf.[REF9]).

17.7. Domibus configuration for OASIS

To enable the integration with the SMP/SML components, Domibus requires some changes in the domibus.properties configuration file:

1. Add the following properties to enable the usage of the OASIS dynamic discovery client:

   domibus.dynamicdiscovery.client.specification"> OASIS

   Note: this property is not mandatory as it defaults to the above value.
2. Set the property "**domibus.smlzone**", e.g. "ehealth.acc.edelivery.tech.ec.europa.eu"

17.8. PMode configuration for OASIS

17.8.1. Sender PMode

In a dynamic discovery process, the receiver of the messages is not known beforehand and therefore the **PMode.Responder** parameter SHOULD NOT be set.

The dynamic discovery process must include a leg which maps the configured entry (action, service and service type – cf. 17.10 – "Message format for PEPPOL") of the Receiver in the SMP.

The security policy to be used in the leg is the policy that embeds the Binary Security Token into the security header (see section §5.1.1 – *"Security Policies"* for more information):

```
security="eDeliveryAS4Policy_BST"
```

Sample Sender PMODE configuration extract:

```
...
<services>
    <service name="testService1"
        value="urn:www.cenbii.eu:profile:bii05:ver2.0"
        type="cenbii-procid-ubl"/>
</services>
<actions>
    <action name="tc1Action"
        value="'your-schema-name'::urn:oasis:names:specification:ubl:schema:xsd:CreditNote-
2::CreditNote##urn:www.cenbii...."/>
</actions>
<securities>
    <security name="eDeliveryAS4Policy_BST"
        policy="eDeliveryAS4Policy_BST.xml"
        signatureMethod="RSA_SHA256"/>
</securities>
<legConfigurations>
    <legConfiguration name="pushTestcase1tc1Action"
    service="testService1"
    action="tc1Action"
    defaultMpc="defaultMpc"
    reliability="AS4Reliability"
    security="eDeliveryAS4Policy_BST"
    receptionAwareness="receptionAwareness"
    propertySet="eDeliveryPropertySet"
    payloadProfile="MessageProfile"
    errorHandling="demoErrorHandling"
    compressPayloads="true"/>
</legConfigurations>
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <initiatorParties>
        <initiatorParty name="senderalias"/>
    </initiatorParties>
    <!-- no responderParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
...
```

Remark:

Schema name should be added to action value. E.g: ***ehealth-actorid-qns::urn:oasis:names:specification:ubl:schema:xsd:CreditNote-2::CreditNote##urn:www.cenbii...***

17.8.2. Receiver PMode

The dynamic discovery configuration of the receiver is similar to the configuration of the sender, except that the roles are swapped: the sender of the messages is not known beforehand. As a consequence, the **PMode.Initiator** parameter SHOULD NOT be set.

```
...
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    inding="push"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <responderParties>
        <responderParty name="receiveralias"/>
    </responderParties>
    <!-- no initiatorParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
...
```

17.9. Policy and certificates for OASIS

The receiver must include the certificate of the trusted authority(ies) in its truststore. It will only accept messages that were signed with certificates issued by the trusted authority(ies).

The sender truststore must include the SMP public certificate. This certificate is used by the AP to validate the identity of the used SMP (cf. §29 –Annex 1 - Usage of certificates in PEPPOL and OASIS for more information).

17.10. Message format for OASIS

When dynamic discovery is used, the "to" field should not be statically configured in the PMode (the "to" field may even be omitted in the message). The lookup is performed by C2 based on the **finalRecipient** message property.

Note 1: For OASIS clients: in the PMode "action" value, the document scheme must be included with the document ID (for PEPPOL client, busdox-docid-qns:: should be pre-appended to the document ID).

Note 2: For OASIS clients: the value of the “service@type” must be set to the “processIdentifier@scheme”.

Example of message using the **finalRecipient** for dynamic discovery:

```
<ns:UserMessage>
 <ns:PartyInfo>
  <ns:From>
   <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">senderalias</ns:PartyId>
   <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
```

```
   </ns:From>
   <ns:To>
   </ns:To>
  </ns:PartyInfo>
  <ns:CollaborationInfo>
   <ns:Service type="cenbii-procid-ubl">urn:www.cenbii.eu:profile:bii05:ver2.0</ns:Service>

          <ns:Action>'your_schema_name'::urn:oasis:names:specification:ubl:schema:xsd:CreditNote-
2::CreditNote##urn:www.cenbii.eu:transaction:biitrns014:ver2.0:extended:urn:www.peppol.eu:bis:pep
pol5a:ver2.0::2.1</ns:Action>
  </ns:CollaborationInfo>
   <ns:MessageProperties>
    <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</ns:Property>
    <ns:Property name="finalRecipient" type="iso6523-actorid-
upis">0007:9340033829test1</ns:Property>
   </ns:MessageProperties>
</ns:UserMessage>
```

18. MESSAGE PULLING

18.1. Setup

In order to configure message pulling, the process section should be configured with **mep** set to "oneway" and binding set to "pull" as shown in the following example:

```
<process name="tc1Process"
    agreement="agreementEmpty"
    mep="oneway"
    binding="pull"
    initiatorRole="defaultInitiatorRole"
    responderRole="defaultResponderRole">
    <initiatorParties>
        <initiatorParty name="initiatoralias"/>
    </initiatorParties >
    <responderParties>
        <responderParty name="receiveralias"/>
    </responderParties>
    <!-- no initiatorParties element -->
    <legs>
        <leg name="pushTestcase1tc1Action"/>
    </legs>
</process>
```

In the case of a pull process, the **initiatorParties** section contains the party that initiate the pull request. The **responderParties** section contains the parties that can be pulled from.

In domibus.properties configuration file, adapt the following properties to your needs. Note that domibus.msh.pull.cron and domibus.pull.queue.concurency are mandatory.

```
#Cron expression used for configuring the message puller scheduling.
#domibus.msh.pull.cron=0 0 0/1 * * ?

# Number of threads used to parallelize the pull requests.
#domibus.pull.queue.concurency=1-1

# Number of threads used to parallelize the pull receipts.
#domibus.pull.receipt.queue.concurrency=1-1

#Number or requests executed every cron cycle
#domibus.pull.request.send.per.job.cycle=1

#Time in second for the system to recover its full pull capacity when job schedule is one execution per second.
#If configured to 0, no incremental frequency is executed and the pull pace is executed at its maximum.
#domibus.pull.request.frequency.recovery.time=0

#Number of connection failure before the system decrease the pull pace.
#domibus.pull.request.frequency.error.count=10

#Pull Retry Worker execution interval as a cron expression
#domibus.pull.retry.cron=0/10 * * * * ?
```

If high frequency pulling is used (job configured every second), it is possible to configure the system to lower the pulling frequency in case the counterpart access point is unavalailable. Per default if the other access point returns errors 10 times in a row (domibus.pull.request.frequency.error.count) the number of pull requests per job cycle will fall to 1 per mpc. As from the moment, the counterpart access point is responding again, Domibus will take the amount of seconds configured within the domibus.pull.request.frequency.recovery.time property to recover the pulling pace configured within the domibus.pull.request.send.per.job.cycle property.

Per default, domibus.pull.request.frequency.recovery.time=0 which means that the throttling mechanism is off.

The following properties are used for dynamic pulling and are recommended to be used only with a custom authorization extension:

```
#Allow dynamic initiator on pull requests - 0 or multiple initiators are allowed in the PMode process
#domibus.pull.dynamic.initiator=false

#Allow multiple legs configured on the same pull process (with the same security policy)
#domibus.pull.multiple_legs=false

#Force message into READY_TO_PULL when mpc attribute is present
#domibus.pull.force_by_mpc=true

#Mpc initiator separator. This is used when the mpc provides information on the initiator: baseMpc/SEPARATOR/partyName
#domibus.pull.mpc_initiator_separator=PID
```

18.2. Configuration restriction

A correctly configured **one-way pull process** should only contain one party configured in the **initiatorParties** section.

Different **legConfiguration** with the same **defaultMpc** (highlighted in red in the following configuration) should not be configured in the same pull process or across different pull processes.

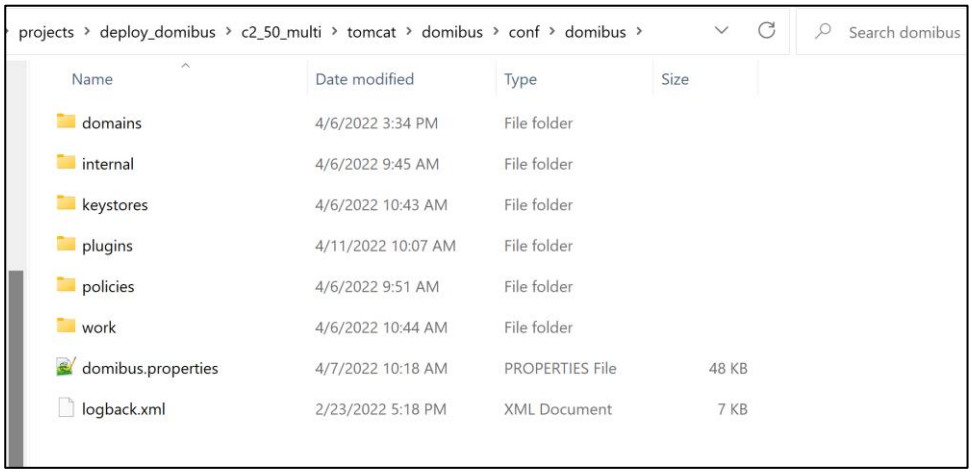
If those restrictions are not respected, the message will not be exchanged and a warning message will detail the configuration problem.

```
<legConfiguration name="pushTestcase1tc2Action"
                                                    service="testService1"
                                                    action="tc2Action"
                                                    defaultMpc="defaultMpc"
                                                    reliability="AS4Reliability"
                                                    security="eDeliveryAs4Policy"
                                                    receptionAwareness="receptionAwareness"
                                                    propertySet="eDeliveryPropertySet"
                                                    payloadProfile="MessageProfile"
                                                    errorHandling="demoErrorHandling"
                                                    compressPayloads="true"/>
```

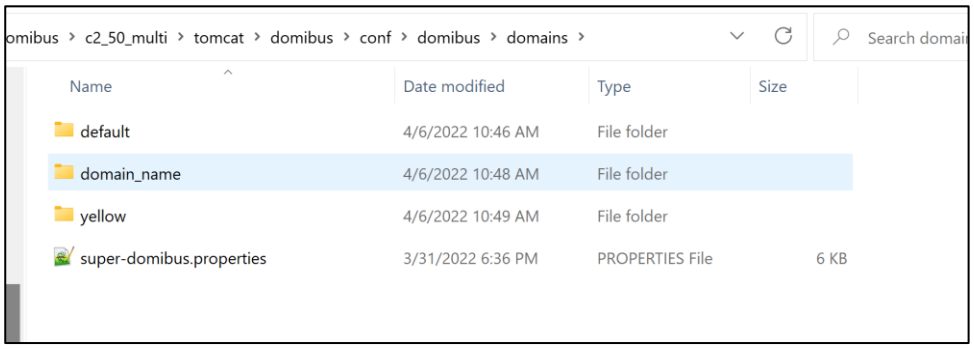
19. MULTITENANCY

Domibus supports multiple tenant domains configured in one Domibus instance. This means that each tenant domain has its own configuration (PMode, keystore, truststore and Domibus properties, etc). These multiple configurations allow one Domibus instance to process messages from multiple tenant domains simultaneously.

The global properties are located in the domibus.properties file, located in the edelivery_path/conf/domibus folder, along with the general logback.xml file, plugins and domains folders:

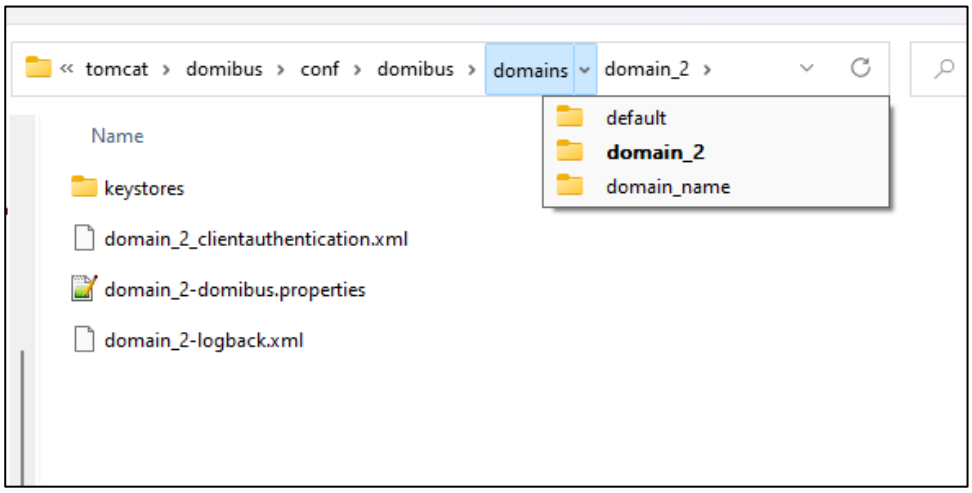


In the root folder there is also a folder called "domains", where the domain specific artefacts are located. The domain-specific artefacts are grouped in domain-specific folders like: *edelivery_path***/conf/domibus/domains/domain_name/.....**



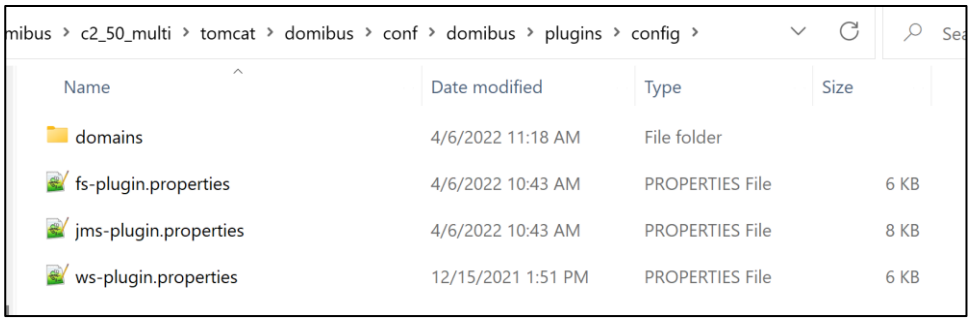
Inside each domain-specific folder there is:

- a "keystores" folder that contains the Domibus keystore, truststore and TLS truststore,
- the clientauthentication.xml file prefixed with the domain name with the description of the TLS TrustStore,
- the logback.xml file prefixed with the domain name and
- the domibus.properties file, also prefixed with the domain name.

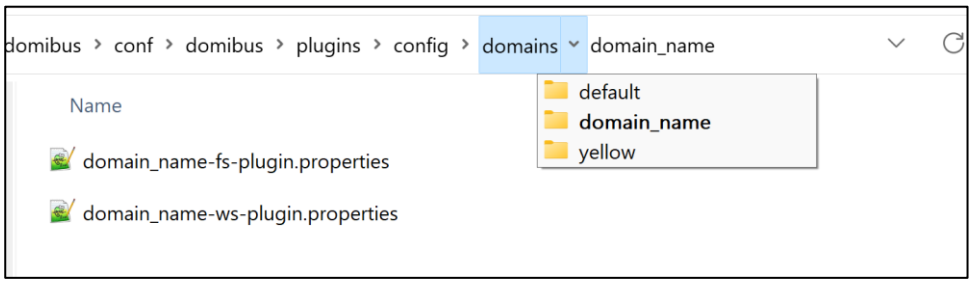


Domibus uses **Schema per tenant** strategy to implement Multitenancy, meaning that the data associated to a tenant domain will be saved in a database schema dedicated to that specific domain.

In case of plugins, the structure follows the same logic: the global properties files are located in the *edelivery_path***/conf/domibus/plugins/config/** folder:



while the domain specific properties are located under the “domains” folder, in files prefixed with the domain name as below:



19.1. Configuration

By default, Multitenancy is not activated. In order to activate Multitenancy, the following property that defines the database general schema needs to be configured in **domibus.properties.**

For Weblogic, this step can only be done after changing the Schema username and password as described in section §19.1.4.

domibus.database.general.schema=***general_schema***

Where *general_schema* is the database schema in which the association between users and domains is stored. The *general_schema* is not associated to any domain.

19.1.1. Database general schema

Configure the MySQL or Oracle datasource as indicated in §4.3.1 - Pre-Configured Single Server Deployment –point 2 (Prepare the database).

The *general_schema* needs to be initialized using the distributed database script **mysql-x.y.z-multitenancy.ddl** for MySQL or **oracle-x.y.z-multitenancy.ddl** for Oracle**.**

Please find below the steps needed to create the *general_schema* for MySQL and Oracle.

19.1.1.1. MySQL

1. Unzip **domibus-msh-distribution-X.Y.Z-sql-scripts.zip** in edelivery_*path*/**sql-scripts**
2. Open a command prompt and navigate to this directory: *edelivery_path*/**sql-scripts**
3. Execute the following MySQL commands at the command prompt:

mysql -h *localhost* -u *root_user* **--password=***root_password* **-e "drop schema if exists** *general_schema***;create schema general_*schema*;alter database** general_*schema* **charset=utf8mb4** collate= **utf8mb4_bin; create user *edelivery_user*@***localhost** **identified by '*edelivery_password*'; grant all on** general_*schema.** **to *edelivery_user*@***localhost***;"*

mysql -h *localhost* -u *root_user* **--*password*=***root_password* **-e "*grant xa_recover_admin on *.* to edelivery_user@localhost;*"**

The above script creates a schema (*general_schema*) and a user (*edelivery_user*) that has all the privileges on the *general_schema*.

Remark:

The edelivery_user creation can be skipped if the user already exists.

You need to make sure the user edelivery_user is granted full rights on all schemas used for all the domains.

mysql -h *localhost* -u *edelivery_user* **--*password*=***edelivery_password* general_*schema* **< mysql-x.y.z-multi-tenancy.ddl**

mysql -h *localhost* -u *edelivery_user* **--*password*=***edelivery_password* general_*schema* **< mysql-x.y.z-multi-tenancy-data.ddl**

The above command creates the required objects in ***general_schema***.

19.1.1.2. Oracle

1. Unzip **domibus-msh-distribution-X.Y.Z-sql-scripts.zip** in *edelivery_path*/**sql-scripts**

2. Open a command prompt and navigate to the following directory: *edelivery_path*/**sql-scripts**

3. Execute the following commands at the command prompt:

sqlplus sys as sysdba (password should be the one assigned during the Oracle installation)
==========
Once logged in Oracle:

```
CREATE USER <edelivery_general_user> IDENTIFIED BY <edelivery_general_password>
DEFAULT TABLESPACE <tablespace>
QUOTA UNLIMITED ON <tablespace>;
GRANT CREATE SESSION TO <edelivery_general_user>;
GRANT CREATE TABLE TO <edelivery_general_user>;
GRANT CREATE SEQUENCE TO <edelivery_general_user>;
GRANT CREATE JOB TO <edelivery_general_user>;
GRANT EXECUTE ON DBMS_XA TO <edelivery_general_user>;
GRANT SELECT ON PENDING_TRANS$ TO <edelivery_general_user>;
GRANT SELECT ON DBA_2PC_PENDING TO <edelivery_general_user>;
GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO <edelivery_general_user>;


CONNECT < edelivery_general_user >
SHOW USER; (should return: edelivery_general_user)
@oracle-x.y.z-multitenancy.ddl
@oracle-x.y.z-multi-tenancy-data.ddl
EXIT
```

==========

Remarks:

1. Replace <edelivery_general_user> and <edelivery_general_password> with the corresponding values.

2. <tablespace> is created and assigned by your DBA; for local/test installations just replace it with users tablespace. The quota could be limited to a specific size.

3. DDL/SQL scripts must be run with the @ sign from the location of the scripts.

19.1.2. Creating new tenant domains

A new tenant domain can be created by adding a domain specific configuration file under the *edelivery_path***/conf/domibus/domains** directory. The domain configuration file name must start with the new tenant domain name (**domain_name**) using the following convention:

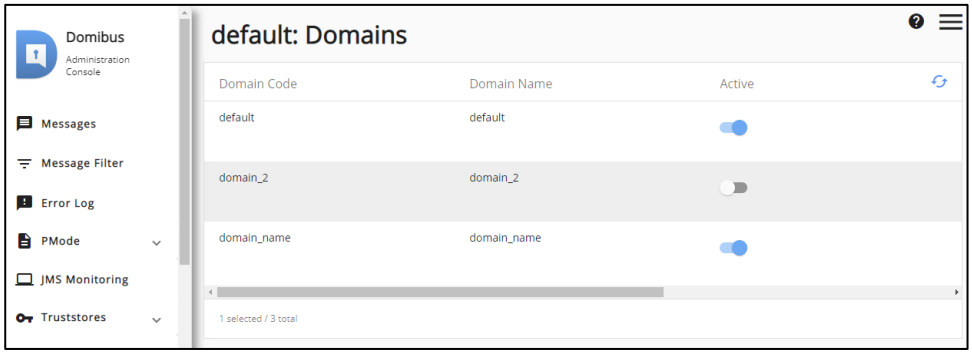
domain_name-domibus.properties

The tenant ***domain_name*** value is case-sensitive. It is a 50-character sequence of Unicode letters, digits or underscores characters. It must start with a letter and the subsequent characters may be letters, digits or underscore characters.

Each tenant domain uses its own dedicated schema which is configured in the domain configuration file and has its own keystore, Truststore configured.

All artefacts pertaining to a domain are located within its directory (keystores, TLS configuration file, properties file, logback, etc.).

It is also possible to add or remove a domain dynamically, without stopping the Domibus, using the **Domains** page of the admin console:



Please keep in mind that prior to adding a domain at runtime, you must create a folder for it in the "domains folder and add the needed artefacts into the folder, like properties file, keystores, etc. Once done, you must click on the **Refresh** button so that the new domain appears in the list as shown above. To activate or de-activate a domain, please use the button under **Active**. To add a domain at runtime, this domain should be active in the **Domains** section of the admin Console.

The tenant domain database schema, including the default domain, must be initialized using the distributed database script **mysql-x.y.z.ddl** or **oracle-x.y.z.ddl**. For more information on how to execute these scripts, go to §4.1-"*Database Configuration"*.

The database user used to connect to the ***general_schema*** schema must have the necessary privileges to access the database schemas for all the **configured tenant domains**. Please follow the steps below for each Database type:

19.1.2.1. MySQL

Execute the following MySQL commands at the command prompt:

If the user ***edelivery_general_user*** is the one having rights on general schema for a particular domain schema, just run:

mysql -h *localhost* -u *root_user* **--password=***root_password* **-e "grant all on *domain_schema*.* to** *edelivery_general_user@localhost***;"**

Repeat this command for all the other domains, including the default domain.

19.1.2.2. Oracle

1. Unzip **domibus-msh-distribution-X.Y.Z-sql-scripts.zip** in *edelivery_path***/sql-scripts**
2. Open a command prompt and navigate to this directory: *edelivery_path***/sql-scripts**.
3. Open a command line session, log in and execute the following commands to connect to current domain schema:

sqlplus s<domain_user>/<domain_password>@host:port/service

```
==========================================================================
Once logged in Oracle:
@oracle-4.2-multitenancy-rights.sql
==========================================================================
```

Before running this script, edit it and just replace domain_schema and general_schema values with the desired values. Repeat this command for each domain of the Multitenancy installation, including the default domain.

This script needs to be run after completing an update of domain Domibus schema (new objects - table, view, sequence – could be added in current domain schema).

Once Multitenancy is activated and with no other additional configuration, Domibus will use the tenant domain named **default** for the incoming and outgoing messages. The tenant domain **default** is configured in **default-domibus.properties.**

More information on how Multitenancy is implemented can be found in the **Domibus Software Architecture Document (c.f. [REF11]).**

19.1.3. Tomcat

The Domibus database in Tomcat is configured in the **domibus.properties** file.

Running Domibus in **Multitenancy** mode requires that some related database properties are adapted as shown in the example below.

Remark: when using Tomcat with Multitenancy, the user should tweak the number of threads defined in the variable domibus.taskExecutor.threadCount, depending on his configuration (see §5.2 – "Domibus Properties").

```
#General schema. Mandatory only if Domibus is configured in multi-tenancy mode.
#domibus.database.general.schema=general_schema

#Domibus schema. If Domibus is configured in multi-tenancy mode this property is used to define the schema for the default domain.
#Please comment this property, if Domibus is configured in single-tenancy mode with Oracle database.
domibus.database.schema=domibus

#Datasource
#MySQL
#Connector/J 8.0.x
#domibus.datasource.driverClassName=com.mysql.cj.jdbc.Driver

#domibus.datasource.url=jdbc:mysql://${domibus.database.serverName}:${domibus.database.port}/${domibus.database.schema}?useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC

#Oracle
#domibus.datasource.driverClassName=oracle.jdbc.OracleDriver
#domibus.datasource.url=jdbc:oracle:thin:@${domibus.database.serverName}:${domibus.database.port}/XE

#domibus.datasource.user=edelivery
```

```
#domibus.datasource.password=edelivery
```

19.1.3.1. domain_name-domibus.properties configuration

Within the tenant domain_name-domibus.properties file, the domain_name field must be replaced by the actual name of the tenant domain as shown in the following sample of the **dom50-domibus.properties** example, where **dom50** is the domain name created:

```
# ---------------------------------- GUI ------------------------
#The title shown in the Tab of Admin Console
#dom50.domibus.ui.title.name=windowTitle
#The name of the domain
#dom50.domain.title=domainTitle
#Number of console login attempt before the user is deactivated (default 5)
#dom50.domibus.console.login.maximum.attempt=5
#Time in seconds for a suspended user to be reactivated. (1 hour per default if property is not set, if 0 the user will not be reactivated)
#dom50.domibus.console.login.suspension.time=3600
#Max rows for CSV export
#dom50.domibus.ui.csv.max.rows=10000
# ---------------------------------- Keystore/Truststore ----------------------
#The location of the keystore
dom50.domibus.security.keystore.location=${domibus.config.location}/keystores/dom1_keystore.jks
#The type of the used keystore
dom50.domibus.security.keystore.type=jks
#The password used to load the keystore
dom50.domibus.security.keystore.password=test123
#Private key
#The alias from the keystore of the private key
dom50.domibus.security.key.private.alias=blue_gw
#etc...
```

19.1.4. WebLogic and WildFly

Most of the database configuration for WebLogic and WildFly is done in the application server. The datasources configured in the application server need to be configured with the user and password that has access to the *general_schema* schema and to all the domain schemas. At runtime the database schema will be changed based on the current domain.

19.1.5. WebLogic specific configuration

Activate the Multitenancy by configuring the following property in **domibus.properties**:

```
domibus.database.general.schema=general_schema
```

Disable basic authentication at the WebLogic level by setting the following property in **DOMAIN_HOME/config/config.xml** (End of the <security-configuration> tag):

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

Example:

```
<security-configuration>
....
<node-manager-password-
encrypted>{AES}hFKbHz7XZ19urplEtWmafYeUm9mr2yXEwyNC9ZpqJHY=</node-manager-password-
encrypted>
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
 </security-configuration>
```

Remark:

Weblogic might not start properly if **domibus.database.general.schema** is set before the **general_schema** username and password have been specified in the Weblogic console. This can be resolved using the following procedure:

1. Comment out (with a #) the domibus.database.general.schema=general_schema line
2. Start the Weblogic server and configure the weblogic server with the username and password of the general_schema
3. Remove the comment in the domibus.database.general.schema=general_schema
4. Restart the Weblogic server

19.2. PMode

In multi-tenant mode each domain, including the default domain, has its own PMode file. When you are logged in as super user, you can select the current domain from the dropdown found in the top right corner. Then follow the instructions from §7.1.4 – "Upload new Configuration" to upload the specific PMode file for each domain.

When C2 wants to send messages to a C3 running in Multitenancy mode, the endpoint URL of C3 configured in the C2 PMode can contain the domain name at the end, configured as an HTTP parameter to indicate the domain that will receive the message.

Example:

Let us suppose that C3 exposes the MSH endpoint under the URL: http://localhost:8080/domibus/service/msh. If C2 wants to send messages to C3 to the domain DIGIT, it will call the following MSH C3 endpoint URL:

http://localhost:8080/domibus/service/msh?domain=digit

In case C2 does not specify the domain in the endpoint URL, the message will be sent to the C3 **default** domain.

19.3. Tenant domain Properties

The properties listed in the table below are used to configure a domain. Some of them must be set here with a specific value for the tenant domain while for most it is not mandatory as they can fall back to the corresponding properties defined in domibus.properties. All the properties defined in a tenant domain property file (e.g. ***domain_name*-domibus.properties**) need to be prefixed by the domain name and override the properties from the **domibus.properties** file.

Example:

1. If the domain name is **digit,** the property file **digit-domibus.properties** is used to configure the **digit** domain.

2. Defining a property named **digit.domibus.msh.messageid.suffix** will override the property **domibus.msh.messageid.suffix** defined in **domibus.properties**.

For each domain, including the default domain, set the properties found in Keystore/Truststore section and also set domain_name.domibus.database.schema.

Domain configuration Property	Defaults to domibus.properties if not defined
domain_name.domibus.database.schema	no
*domain_name.*domibus.UI.title.name	yes
*domain_name.domibus.*ui.csv.max.rows	yes
domain_name.domibus.msh.messageid.suffix	yes
domain_name.domibus.msh.retry.cron	yes
*domain_name.*domibus.dynamicdiscovery.useDynamicDiscovery	yes
domain_name.domibus.smlzone	yes
domain_name.domibus.dynamicdiscovery.client.specification	yes
domain_name.domibus.dynamicdiscovery.peppolclient.mode	yes
domain_name.domibus.dynamicdiscovery.oasisclient.regexCertificateSubjectValidation	yes
domain_name.domibus.dynamicdiscovery.partyid.responder.role	yes
domain_name.domibus.dynamicdiscovery.partyid.type	yes
domain_name.domibus.dynamicdiscovery.lookup.clean.retention.hours	yes

domain_name.domibus.dynamicdiscovery.lookup.clean.retention.cron	yes
domain_name.domibus.dispatcher.allowChunking	yes
domain_name.domibus.dispatcher.chunkingThreshold	yes
domain_name.domibus.dispatcher.concurency	yes
domain_name.domibus.dispatcher.connectionTimeout	yes
domain_name.domibus.dispatcher.receiveTimeout	yes
domain_name.domibus.dispatcher.cacheable	yes
domain_name.domibus.msh.pull.cron	yes
domain_name.domibus.pull.queue.concurency	yes
domain_name.domibus.pull.request.send.per.job.cycle	yes
domain_name.domibus.pull.retry.cron	yes
domain_name.domibus.retentionWorker.cronExpression	yes
domain_name.message.retention.downloaded.max.delete	yes
domain_name.message.retention.not_downloaded.max.delete	yes
domain_name.domibus.sendMessage.messageIdPattern	no
domain_name.domibus.attachment.storage.location	no
domain_name.domibus.msh.retry.tolerance	yes
domain_name.domibus.security.keystore.location	no

domain_name.domibus.security.keystore.type	no
domain_name.domibus.security.keystore.password	Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" must be escaped in domibus.properties file
domain_name.domibus.security.key.private.alias	
domain_name.domibus.security.key.private.password	Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" must be escaped in domibus.properties file
domain_name.domibus.security.truststore.location	no
domain_name.domibus.security.truststore.type	no
domain_name.domibus.security.truststore.password	Accepted characters are: !\"#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ Please note that \\ \' and \" must be escaped in domibus.properties file
domain_name.domibus.receiver.certificate.validation.onsending	yes
domain_name.domibus.sender.certificate.validation.onsending	yes
domain_name.domibus.sender.certificate.validation.onreceiving	yes
domain_name.domibus.sender.trust.validation.onreceiving	yes
*domain_name.*domibus.sender.trust.validation.truststore_alias	yes
domain_name.domibus.sender.trust.validation.expression	yes
domibus.sender.trust.validation.allowedCertificatePolicyOIDs	yes
domain_name.domibus.sender.certificate.subject.check	yes

domain_name.domibus.alert.retry.cron	yes
domain_name.domibus.alert.cleaner.cron	yes
domain_name .domibus.alert.sender.email	
domain_name .domibus.alert.receiver.email	
domain_name.domibus.alert.cleaner.cron	0 0 0/1 * * ?
domain_name.domibus.alert.cleaner.alert.lifetime	20
domain_name.domibus.alert.active	TRUE
domain_name.domibus.alert.mail.sending.active	FALSE
domain_name.domibus.alert.mail.smtp.timeout	5000
domain_name.domibus.alert.queue.concurrency	1
domain_name.domibus.alert.retry.cron	0 0/10 * * * ?
domain_name.domibus.alert.retry.time	10
domain_name.domibus.alert.retry.max_attempts	2
domain_name.domibus.alert.msg.communication_failure .active	TRUE
domain_name.domibus.alert.msg.communication_failure .states	SEND_FAILURE
domain_name.domibus.alert.msg.communication_failure .level	HIGH
domain_name.domibus.alert.msg.communication_failure .mail.subject	Message status change
domain_name.domibus.alert.user.login_failure.active	TRUE

domain_name.domibus.alert.user.login_failure.level	LOW
domain_name.domibus.alert.user.login_failure.mail.subject	Login failure
domain_name.domibus.alert.user.account_disabled.active	TRUE
domain_name.domibus.alert.user.account_disabled.level	HIGH
domain_name.domibus.alert.user.account_disabled.moment	WHEN_BLOCKED
domain_name.domibus.alert.user.account_disabled.subject	Account disabled
domain_name.domibus.alert.cert.imminent_expiration.active	TRUE
domain_name.domibus.alert.cert.imminent_expiration.frequency_days	14
domain_name.domibus.alert.cert.imminent_expiration.level	HIGH
domain_name.domibus.alert.cert.imminent_expiration.mail.subject	Certificate imminent expiration
domain_name.domibus.alert.cert.expired.active	TRUE
domain_name.domibus.alert.cert.expired.frequency_days	7
domain_name.domibus.alert.cert.expired.duration_days	90
domain_name.domibus.alert.cert.expired.level	HIGH
domain_name.domibus.alert.cert.expired.mail.subject	Certificate expired
domain_name.domibus.dynamicdiscovery.transportprofileas4	yes

domain_name.domibus.dispatcher.connection.keepAlive	yes
domain_name.domibus.dispatcher.splitAndJoin.payloads.schedule.threshold	1000
domain_name.domibus.splitAndJoin.receive.expiration.cron	0 0/5 * * * ?
domain_name.domibus.pull.dynamic.initiator	yes
domain_name.domibus.pull.multiple_legs	yes
domain_name.domibus.pull.force_by_mpc	yes
domain_name.domibus.pull.mpc_initiator_separator	yes

Remark:

A tenant domain property is mandatory to be defined if it does not default to ***domain.properties.***

19.4. Super Properties

The properties that are sepecif to super users (ROLE_AP_ADMIN) are defined in a separate file called **super-domibus.properties,** a file that can be found along with the others. These properties are related to password policy and alert configuration for super users.

19.5. Logging

Domibus generates logs in 3 log files when running in non Multitenancy mode (**domibus.log, domibus-business.log and domibus-security.log**), that are configured in the **logback.xml** file. More information about what is being logged into those files can be found in §10.5 – "*Application Logging*".

In Multitenancy mode, the following should be expected:

- main files - **domibus.log**, **business.log** and **security.log** will contain only general logging information and not domain specific;

- 'per domain' files, e.g. domain1-domibus.log, domain1-business.log and domain1-security.log will contain logging entries only for the specific domain 'domain1';

- it is mandatory to add a **domain logback.xml** for each domain including the 'default' one. Pay attention that, if such file does not exist, the logging information may be lost for that domain.

When running in Multitenancy mode, the Domibus log configuration file **logback.xml** has to be modified as followed:

a. uncomment all the sections marked like this one:

```
<!-- multitenancy: uncomment this
<filter class="eu.domibus.logging.DomibusLoggerDomainFilter">
  <domain></domain>
  <OnMismatch>DENY</OnMismatch>
</filter>
-->
```

b. edit the file to include the log configuration for each domain. This is necessary to segregate the log statements per tenant domain, each tenant domain having its own set of the 3 logs files mentioned above:

```
<!-- multitenancy: start include domains config files here -->
<!--<include optional="true" file="${catalina.home}/conf/domibus/domain_name-logback.xml"/>-
->
<!-- multitenancy: end include domains config files here -->
```

c. add a domain config file for the 'default' domain.

In order to configure the logs per domain please follow the steps:

1. Customize the **domain_name-logback.xml** file distributed in each server configuration archive.

   a. Rename the **domain_name-logback.xml** file according to the domain name. E.g. if the domain name is **domain1**, the file should be renamed to **domain1-logback.xml.**

   b. Adapt the value of the **domainName** variable defined in the domain logback configuration file. The value should correspond to the name of the configured domain.

```
<included>
  <property name="domainName" value="domain1" scope="local" />
```

2. Include the domain configuration file into the main **logback.xml** file:

```
<configuration>
  <!-- start include domains config files here -->
  <include optional="true" file="${catalina.home}/conf/domibus/domain1-logback.xml"/>
```

In order to add some particular logging information per domain, the user must add in the **domain1-logback.xml** file the following section (example for ‘domain1’ domain):

```
<logger name=”${domainName}.eu.domibus.somepackage” level="DEBUG" additivity="false">
 <appender-ref ref="${domainName}-file"/>
 <appender-ref ref="stdout"/>
</logger>
```

In the example above, 'eu.domibus.somepackage' is the name of the package for setting DEBUG level, '${domainName}-file' is the appender of 'domain1'.

Note that this mechanism applies only to eu.domibus loggers. Loggers from third-party libraries cannot be configured independently for a tenant, they should be added to the main logback.xml file with the settings that apply to all tenants.

The line with 'stdout' is optional and it will print the DEBUG info on the server console.

19.6. Users

In Multitenancy mode there is a new user named **super** with role ROLE_AP_ADMIN which has the privileges to access all the available domains. The default password for the **super** user is written in the logs as “Default password for super user is”.

The first time a new tenant domain is created, the **super** user creates a new user in the **Domibus Administration Console** with role ROLE_ADMIN associated to the newly created domain. All normal users (ROLE_ADMIN, ROLE_USER) can be associated to only and only one domain. More details how to create users can be found in the help page of the **Users** page.

Afterwards the **super** user sends the credentials to the domain admin user. The domain admin logs into the **Domibus Administration Console** using the received credentials and has to change its password in the **Users** page. The domain admin has only access to his domain and he has the privileges to create only new users that are associated to his domain.

Remark:

Please note that user names need to be unique amongst existing tenant domains.

19.7. Plugins

When running in Multitenancy mode, the plugins security is activated by default, no matter if the property **domibus.auth.unsecureLoginAllowed** in the domibus.properties files is set to true or not. This is needed to identify the request performed by the user and associate it to a specific tenant domain. As a result, every request sent to Domibus needs to be authenticated.

Remark:

Please note that the **Default JMS Plugin** *requires the creation of additional JMS queues. More information on which queues need to be created can be found in the* JMS Plugin Interface Control Document (ICD) *(see* [REF12]).

More information on this topic can be found in the Domibus Software Architecture Document (SAD) (c.f. [REF11]).

19.7.1. Plugin Users

In Multitenancy mode, a plugin must use a configured plugin user associated to a specific tenant domain to authenticate every request sent to Domibus. The management of the plugin users is implemented in the **Plugin Users** page of **Domibus Administration Console**. More details about how to manage the plugin users can be found in the help page of the **Plugin Users** page (see also §10 – “*Administration Tools”*).

The **Default JMS Plugin** and the **Default FS Plugin** implement only authentication mechanism. The two previously mentioned plugins must use any configured plugin user to send requests to Domibus, no matter the role: ROLE_ADMIN or ROLE_USER. The request will be sent to the domain associated to the plugin user used for authentication.

The **Default WS Plugin** implements authentication and authorization mechanism.

For authentication the **Default WS Plugin** must use a configured plugin user to send requests to Domibus, the configuration being the same as for the **Default JMS Plugin** and the **Default FS Plugin**.

More details about how the authorization is implemented in the **Default WS Plugin** can be found in §6.1.2 *"WS Plugin"* and in the plugin cookbook document (cf.[REF6]).

Remark:

Please note that user names need to be unique amongst existing tenant domains.

19.8. Switching from non Multitenancy to Multitenancy mode

When switching an existing installation of Domibus to Multitenancy mode, the instructions described in §19.1 – *"Configuration"* have to be executed.

After the switch to Multitenancy mode is finished, the schema that was previously used in non Multitenancy mode will be used by a specific tenant domain. Additionally the **super** user must select the migrated tenant domain in Domibus Administration console and re-create the existing users present in the **Users** and **Plugin Users**. This step is required because in Multitenancy mode there is an automatic synchronization of domain users into the general schema. More info about the synchronization of tenant domain users can be found in the Domibus Software Architecture Document (SAD) (c.f. [REF11]).

20. ALERTS

20.1. Description

The purpose of the alert feature is to use different available media to notify the Domibus administrator in case of unusual behaviour. Those notifications are presented to the Domibus administrator under the form of configurable alerts. The alerts can be browsed in the **Domibus Admin Console** in the Alerts section and can be sent by **email**.

Currently, only email notification channel is available, but other communication media will be added in future releases.

Three topics are available for monitoring:

- Message status change
- Authentication issues
- Certificate expiration.

20.2. Main configuration

The properties, described below, can be configured in the domibus.properties configuration file.

By default, alerts are not activated. A single property can activate or deactivate the entire alert concept. In order to activate it, the following property should be set to true:

```
# ---------------------------------- Alert management ----------------------------------
#enable/disable the entire alert module. Pay attention to the fact that if the module is activated, all properties
#under the mandatory section should be configured.
domibus.alert.active=true
```

Once the alerts are activated, the SMTP server needs also to be configured. In that case, the following properties are mandatory:

```
# ----------------------------------Mandatory configuration start (if domibus.alert.active=true) -----------

#Smtp sever url for sending alert
#domibus.alert.sender.smtp.url=

#Smtp sever port
#domibus.alert.sender.smtp.port=

#Smtp sever user
#domibus.alert.sender.smtp.user=

#Smtp sever user password
#domibus.alert.sender.smtp.password=
```

```
#Alert sender email
#domibus.alert.sender.email=

#Alert email receiver.
#domibus.alert.receiver.email=
```

The first four properties are used to configure respectively the URL, the port, the user and the password to authenticate to the SMTP server.

The last two properties are needed to respectively set the emails of the alert sender and the alert receiver.

The following properties are already preconfigured with default values and therefore are not mandatory to be configured:

```
#The following properties can stay commented if no modifications to the default values are needed.

#Cron configuration for cleaning alerts.
#domibus.alert.cleaner.cron=0 0 0/1 * * ?

# Alerts lifetime in days of before cleaning.
#domibus.alert.cleaner.alert.lifetime=20

#Concurrency to process the alerts.
#domibus.alert.queue.concurrency=1

#Frequency of failed alerts retry.
#domibus.alert.retry.cron=0 0/1 * * * ?

#Elapsed time in minute between alert retry.
#domibus.alert.retry.time=1

#Number of retry for failed alerts.
#domibus.alert.retry.max_attempts=2
```

By default, Domibus will check every hour for expired alerts. The default lifetime for an alert is 20 days after which the alert is deleted from the system.

The concurrency property allows processing multiple alerts in parallel. Alerts can be configured with a retry in case of dispatch failure. By default Domibus will wait one minute between two alert dispatch attempts, and it will retry twice.

Multitenancy

In Multitenancy mode, the four SMTP properties should be configured in the main domibus.properties. Indeed only one SMTP server can be configured for all the tenants.

On the other hand, the sender and receiver properties must be configured in each domain configuration file.

Multitenancy also introduces the existence of a super user. Authentication alerts can be configured for it. Some specific global properties have been created for the super user. The following properties are documented with their default value. They can be overwritten in domibus.properties file:

```
# ---------------------------------- Super user Alert management ----------------------------------

#Cron configuration for cleaning alerts.
#domibus.alert.super.cleaner.cron=0 0 0/1 * * ?

#Lifetime in days of alerts before cleaning.
#domibus.alert.super.cleaner.alert.lifetime=20

#Enable/disable the entire alert module.
#domibus.alert.super.active=true

#Allow to disable alert mail sending.
#domibus.alert.super.mail.sending.active=false

#Frequency of failed alerts retry.
#domibus.alert.super.retry.cron=0 0/1 * * * ?

#Elapsed time in minutes between alert retry.
#domibus.alert.super.retry.time=1

#Maximum number of attempts for failed alerts
#domibus.alert.super.retry.max_attempts=2
```

20.3. Message status change alerts

Domibus is able to track Message status changes. All status changes can be tracked but it is advised not to track the status of frequently changing statuses (e.g.: From SEND_ENQUEUED to ACKNOWLEDGE) to avoid being spammed.

Each alert topic (Message status change, authentication and certificate expiration) can be activated or deactivated independently from each other. Pay attention that, in order for the alert feature to work, the main alert module must always be activated (see § 20.2-*"Main configuration"*).

By default, message status change alerts are not activated. In order to activate them, the following property should be set to true:

```
# ---------------------------------- Alert management: messaging module -----------------
#enable/disable the messaging alert module.
domibus.alert.msg.communication_failure.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to be configured:

```
#Message status change that should be notified by the messaging alert module. Comma-separated.
#domibus.alert.msg.communication_failure.states=SEND_FAILURE

#Alert levels corresponding to message status defined in previous
property(domibus.alert.msg.communication_failure.states) . Should be (HIGH, MEDIUM or LOW)
#domibus.alert.msg.communication_failure.level=HIGH

#Messaging alert module mail subject.
#domibus.alert.msg.communication_failure.mail.subject=Message status change
```

By default, Domibus will only track message status change to SEND_FAILURE. The level of the alert that will be triggered is HIGH. The last property allows configuring the subject of the mail sent.

If there is a need to track another message status change, a comma-separated list can be configured:

```
Eg: domibus.alert.msg.communication_failure.states=SEND_FAILURE,ACKNOWLEDGED
```

If there is a need to set an alert level per status change it can also be done with a comma-separated list:

```
domibus.alert.msg.communication_failure.level=HIGH,LOW
```

In the example above, an alert for a message being set in send_failure status will have a high level and an alert for a message being set to acknowledged status will have a low level.

20.4. Authentication Alerts

Domibus is able to track admin console login failure and user account disabling. The login failure alert will occur for each unsuccessful attempt. Note that if the username encoded is unknown to the system, no alert will be created. Only known user with invalid password will be tracked. The account disabled alert will occur either because the user did too many invalid login attempts or because an administrator disabled the account.

By default, login failure alerts are not activated. In order to activate them, the following property should be set to true:

```
# ---------------------------------- Alert management: Authentication module -----------------

#Enable/disable the login failure alert of the authentication module.
domibus.alert.user.login_failure.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Alert level for login failure.
#domibus.alert.user.login_failure.level=LOW

#Login failure mail subject.
#domibus.alert.user.login_failure.mail.subject=Login failure
```

Per default, the alert level for a login failure is low. The last property allows configuring the subject of the mail sent.

By default, account disabled alerts are not activated. In order to activate them, the following property should be set to true:

```
#Enable/disable the account disable alert of the authentication module.
domibus.alert.user.account_disabled.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Alert level for account disabled.
#domibus.alert.user.account_disabled.level=HIGH

#When should the account disabled alert be triggered.
# 2 possible values:
# AT_LOGON: An alert will be triggered each time a user tries to login to a disabled account.
# WHEN_BLOCKED: An alert will be triggered once when the account got disabled.
#domibus.alert.user.account_disabled.moment=WHEN_BLOCKED

#Account disabled mail subject.
#domibus.alert.user.account_disabled.subject=Account disabled
```

Per default, the alert level for an account disabled is high. The next property specifies when an account_disabled alert should be triggered. It can be only at disabling time or at every new login attempt after the account has been disabled. The default value WHEN_BLOCKED will therefore create only one alert when the account is disabled.

The last property allows configuring the subject of the mail sent.

Multitenancy

The following super user authentication alerts properties are documented with their default value. They can be overwritten in the domibus.properties file:

```
# ---------------------------------- Super user alert management:Authentication module -----------------

#Enable/disable the login failure alert of the authentication module.
#domibus.alert.super.user.login_failure.active=true

#Alert level for login failure.
#domibus.alert.super.user.login_failure.level=LOW

#Login failure mail subject.
#domibus.alert.super.user.login_failure.mail.subject=Super user login failure

#Enable/disable the account disable alert of the authentication module.
#domibus.alert.super.user.account_disabled.active=true

#Alert level for account disabled.
#domibus.alert.super.user.account_disabled.level=HIGH

#When should the account disabled alert be triggered.
# 2 possible values:
# AT_LOGON: An alert will be triggered each time a user tries to login to a disabled account.
# WHEN_BLOCKED: An alert will be triggered once when the account got disabled.
#domibus.alert.super.user.account_disabled.moment=WHEN_BLOCKED

#Account disabled mail subject.
#domibus.alert.super.user.account_disabled.subject=Super user account disabled
```

All that was mentioned earlier about console users is also true for the plugin users. There is an identical set of configuration properties for them:

```
# ---------------------------------- Alert management:Authentication module for Plugin users-----------------

#Enable/disable the login failure alert of the authentication module.
#domibus.alert.plugin.user.login_failure.active=true

#Alert level for login failure.
#domibus.alert.plugin.user.login_failure.level=LOW

#Login failure mail subject.
#domibus.alert.plugin.user.login_failure.mail.subject=Login failure

#Enable/disable the account disable alert of the authentication module.
#domibus.alert.plugin.user.account_disabled.active=true

#Alert level for account disabled.
#domibus.alert.plugin.user.account_disabled.level=HIGH

#When should the account disabled alert be triggered.
# 2 possible values:
# AT_LOGON: An alert will be triggered each time a user tries to login to a disabled account.
# WHEN_BLOCKED: An alert will be triggered once when the account got disabled.
#domibus.alert.plugin.user.account_disabled.moment=WHEN_BLOCKED

#Account disabled mail subject.
```

```
#domibus.alert.plugin.user.account_disabled.subject=Account disabled

#Account disabled mail subject.
#domibus.alert.super.user.account_disabled.subject=Super user account disabled
```

20.5. User Password alerts

Domibus is able to track user password expiration and imminent expiration. Obviously the user password expired alert occurs when a user password expires. The number of days the alert should be triggered after the expiration is configurable. The imminent expiration alert occurs for some time before the user password expiration. The number of days the alert should be triggered before expiration is configurable. The alert frequency for both trackers can be configured.

By default, imminent user password expiration alerts are not activated. In order to activate them, the following property should be set to true:

```
# ---------------------------------- Alert management:Password policy  -----------------

#Enable/disable the imminent password expiration alert
#domibus.alert.password.imminent_expiration.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Number of days before expiration as for how long before expiration the system should send alerts.
#domibus.alert.password.imminent_expiration.delay_days=15

#Frequency in days between alerts.
#domibus.alert.password.imminent_expiration.frequency_days=3

#Password imminent expiration alert level.
#domibus.alert.password.imminent_expiration.level=LOW

#Password imminent expiration mail subject.
#domibus.alert.password.imminent_expiration.mail.subject=Password imminent expiration
```

By default, Domibus will send user password imminent expiration alerts 15 days before the expiration. It will send alerts at a pace of one alert every 3 days. The level of the alert will be LOW. The last property allows configuring the subject of the mail sent.

By default, user password expired alerts are not activated. In order to activate them, the following property should be set to true:

```
#Enable/disable the certificate expired alert of certificate scanner module.
domibus.alert.password.expired.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Number of days after expiration as for how long the system should send alerts.
```

```
#domibus.alert.password.expired.delay_days=30

#Frequency in days between alerts.
#domibus.alert.password.expired.frequency_days=5

#Password expiration alert level.
#domibus.alert.password.expired.level=LOW

#Password expiration mail subject.
#domibus.alert.password.expired.mail.subject=Password expired
```

By default, Domibus will send user password expired alerts during 30 days after the expiration. It will send alerts at a pace of one alert every 5 days. The level of the alert will be LOW. The last property allows configuring the subject of the mail sent.

20.6. Plugin User Password alerts

Everything that was explained above about the console users alerts is also true for the plugin users. Their corresponding properties are listed below:

```
# ---------------------------------- Alert management: Plugin Password policy  -----------------

#Enable/disable the imminent password expiration alert
#domibus.alert.plugin_password.imminent_expiration.active=true

#Number of days before expiration as for how long before expiration the system should send alerts.
#domibus.alert.plugin_password.imminent_expiration.delay_days=15

#Frequency in days between alerts.
#domibus.alert.plugin_password.imminent_expiration.frequency_days=3

#Password imminent expiration alert level.
#domibus.alert.plugin_password.imminent_expiration.level=LOW

#Password imminent expiration mail subject.
#domibus.alert.plugin_password.imminent_expiration.mail.subject=Password imminent expiration

#Enable/disable the imminent password expiration alert
#domibus.alert.plugin_password.expired.active=true

#Number of days after expiration as for how long the system should send alerts.
#domibus.alert.plugin_password.expired.delay_days=30

#Frequency in days between alerts.
#domibus.alert.plugin_password.expired.frequency_days=5

#Password expiration alert level.
#domibus.alert.plugin_password.expired.level=LOW

#Password expiration mail subject.
#domibus.alert.plugin_password.expired.mail.subject=Password expired
```

20.7. Certificate scanner alerts

Domibus is able to track certificate expiration and imminent expiration. Obviously the certificate expired alert occurs when a certificate expires. The number of days the alert should be triggered after the expiration is configurable. The imminent expiration alert occurs for some time before the certificate expiration. The number of days the alert should be triggered before expiration is configurable. The alert frequency for both trackers can be configured.

By default, imminent certificate expiration alerts are not activated. In order to activate them, the following property should be set to true:

```
# ---------------------------------- Alert management: Certificate scanner -----------------

#Enable/disable the imminent certificate expiration alert of certificate scanner module.
domibus.alert.cert.imminent_expiration.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Number of days before revocation as from when the system should start sending alerts.
#domibus.alert.cert.imminent_expiration.delay_days=61

#Frequency in days between alerts.
#domibus.alert.cert.imminent_expiration.frequency_days=14

#Certificate imminent expiration alert level.
#domibus.alert.cert.imminent_expiration.level=HIGH

#Certificate imminent expiration mail subject.
#domibus.alert.cert.imminent_expiration.mail.subject=Certificate imminent expiration
```

By default, Domibus will send certificate imminent expiration alerts 61 days before the expiration. It will send alerts at a pace of one alert every 14 days. The level of the alert will be HIGH. The last property allows configuring the subject of the mail sent.

By default, certificate expired alerts are not activated. In order to activate them, the following property should be set to true:

```
#Enable/disable the certificate expired alert of certificate scanner module.
domibus.alert.cert.expired.active=true
```

The following properties are already preconfigured with default values and therefore are not mandatory to configure:

```
#Frequency in days between alerts.
#domibus.alert.cert.expired.frequency_days=7

#How long(in days) after the revocation should the system trigger alert for the expired certificate.
#domibus.alert.cert.expired.duration_days=92

#Certificate expired alert level.
#domibus.alert.cert.expired.level=HIGH
```

```
#Certificate expired mail subject.
#domibus.alert.cert.expired.mail.subject=Certificate expired
```

By default, Domibus will send certificate expired alerts during 92 days after the expiration. It will send alerts at a pace of one alert every 7 days. The level of the alert will be HIGH. The last property allows configuring the subject of the mail sent.

20.8. Configuration example

20.8.1. Example: domibus.properties

Below is shown only the section relevant to the alerts configuration in the **domibus.properties** configuration file, when the SMTP server is running in the same host as domibus (localhost):

```
…
…
# ---------------------------------- Alert management ----------------------------------
#Enable/disable the entire alert module. Pay attention to the fact that if the module is activated, all
properties
#under the mandatory section should be configured.
domibus.alert.active=true
#Allow to disable alert mail sending.
domibus.alert.mail.sending.active=true
domibus.alert.mail.smtp.starttls.enable=false
domibus.alert.mail.smtp.auth=false
#domibus.alert.mail.smtp.timeout=10000
# ----------------------------------Mandatory configuration start (if domibus.alert.active=true) ----------
#Smtp server url for sending alert.
domibus.alert.sender.smtp.url=localhost
#Smtp server port.
domibus.alert.sender.smtp.port=25
#Smtp server user.
#domibus.alert.sender.smtp.user=
#Smtp server user password
#domibus.alert.sender.smtp.password=
#Alert sender email.
domibus.alert.sender.email=sender@exemple.com
#Alert email receiver.
domibus.alert.receiver.email=mcb@gmail.com
# ----------------------------------Mandatory configuration end---------------------------------
#The following properties can stay commented if no modifications to the default values are needed.
#Cron configuration for cleaning alerts.
domibus.alert.cleaner.cron=0 0/1 * * * ?
#Lifetime in days of alerts before cleaning.
domibus.alert.cleaner.alert.lifetime=1
#Concurrency to process the alerts.
#domibus.alert.queue.concurrency=1

#Frequency of failed alerts retry.
#domibus.alert.retry.cron=0 0/1 * * * ?
#Elapsed time in minutes between alert retry.
```

```
#domibus.alert.retry.time=1
#Maximum number of attempts for failed alerts
#domibus.alert.retry.max_attempts=2
# ---------------------------------- Alert management:messaging module -----------------
#Enable/disable the messaging alert module.
#domibus.alert.msg.communication_failure.active=true
#Message status change that should be notified by the messaging alert module. Comma-separated.
domibus.alert.msg.communication_failure.states=SEND_FAILURE,WAITING_FOR_RETRY
#Alert levels corresponding to message status defined in previous
property(domibus.alert.msg.communication_failure.states) .
#Should be (HIGH, MEDIUM OR LOW)
#domibus.alert.msg.communication_failure.level=HIGH
#Messaging alert module mail subject.
domibus.alert.msg.communication_failure.mail.subject=Message status change MCB
# ---------------------------------- Alert management:Authentication module -----------------
#Enable/disable the login failure alert of the authentication module.
domibus.alert.user.login_failure.active=true
#Alert level for login failure.
#domibus.alert.user.login_failure.level=LOW
#Login failure mail subject.
domibus.alert.user.login_failure.mail.subject=Login failure MCB
#Enable/disable the account disable alert of the authentication module.
#domibus.alert.user.account_disabled.active=true
#Alert level for account disabled.
#domibus.alert.user.account_disabled.level=HIGH
#When should the account disabled alert be triggered.
# 2 possible values:
# AT_LOGON: An alert will be triggered each time a user tries to login to a disabled account.
# WHEN_BLOCKED: An alert will be triggered once when the account got disabled.
domibus.alert.user.account_disabled.moment=WHEN_BLOCKED,AT_LOGON
#Account disabled mail subject.
domibus.alert.user.account_disabled.subject=Account disabled MCB
# ---------------------------------- Alert management:Certificate scanner -----------------
#Enable/disable the imminent certificate expiration alert of certificate scanner module.
domibus.alert.cert.imminent_expiration.active=false
#Number of days before revocation as from when the system should start sending alerts.
domibus.alert.cert.imminent_expiration.delay_days=20000
#Frequency in days between alerts.
#domibus.alert.cert.imminent_expiration.frequency_days=14
#Certificate imminent expiration alert level.
#domibus.alert.cert.imminent_expiration.level=HIGH
#Certificate imminent expiration mail subject.
domibus.alert.cert.imminent_expiration.mail.subject=Certificate imminent expiration MCB
#Enable/disable the certificate expired alert of certificate scanner module.
domibus.alert.cert.expired.active=false
#Frequency in days between alerts.
#domibus.alert.cert.expired.frequency_days=7
#How long(in days) after the revocation should the system trigger alert for the expired certificate.
#domibus.alert.cert.expired.duration_days=90
#Certificate expired alert level.
#domibus.alert.cert.expired.level=HIGH
#Certificate expired mail subject.
domibus.alert.cert.expired.mail.subject=Certificate expired MCB
```

```
# ---------------------------------- Proxy settings ----------------------------------
….
```

20.8.2. Example: domain_name-domibus.properties

Below is shown only the section relevant to the alerts configuration in the **dom50-domibus.properties** configuration file, where dom50 is the name of a domain:

```
…
#Pull Retry Worker execution interval as a cron expression
#dom50.domibus.pull.retry.cron=0/10 * * * * ?
# ---------------------------------- Alert management ----------------------------------
#Enable/disable the entire alert module. Pay attention to the fact that if the module is activated, all
properties
#under the mandatory section should be configured.
dom50.domibus.alert.active=true
#Allow to disable alert mail sending.
dom50.domibus.alert.mail.sending.active=true
# ----------------------------------Mandatory configuration start (if domibus.alert.mail.sending.active=true) ---
#Alert sender email.
dom50.domibus.alert.sender.email=mcb@gmail.com
#Alert email receiver.
dom50.domibus.alert.receiver.email=mcb@gmail.com
# ----------------------------------Mandatory configuration end----------------------------------
#The following properties can stay commented if no modifications to the default values are needed
…
```

21. DSS EXTENSION CONFIGURATION

21.1. Overview

Domibus now offers the possibility to perform incoming messages certificate chain validation with the DSS library instead of the truststore. In order to achieve chain validation with DSS, Domibus security policy should be configured with a PKI path (see the file “eDeliveryAS4Policy_BST_PKIP.xml” in the distribution).
When PKI path is used, the full chain of certificates that contains the signing and its trust certificates is embedded in the security header of the SOAP message.
Domibus DSS extension will download and use per default the European list of trusted lists (LOTL).

Dominus can verify the trust anchor of any certificate chain having a certificate authority present within the LOTL.

The DSS extension also permits to configure custom trusted lists with additional certificate authorities.

DSS generates a validation report with different constraints and status. The DSS extension allows configuring the relevant constraints for the validation.

21.2. Installation

21.2.1. Enable Unlimited Strength Jurisdiction Policy

- Before Java 8 Update 151

For Java 8 Update 144 and earlier, you need to install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy files:

1. Download the unlimited strength JCE policy files from Oracle by clicking here
2. Extract the downloaded file
3. Replace the existing policy JAR files in $JAVA_HOME/jre/lib/security with the extracted unlimited strength policy JAR files

- Java 8 Update 151 and higher

The Unlimited Strength Jurisdiction Policy is included but not used by default. To enable it, you need to edit the java.security file in $JAVA_HOME/jre/lib/security (for JDK) or $JAVA_HOME/lib/security (for JRE). Uncomment (or include) the following line:

```
crypto.policy=unlimited
```

21.2.2. Download and install DSS extension

For this step, you will have to use the following resources (see section §3.1–*"Binaries repository"* for the download location):

- **domibus-msh-distribution-X.Y.Z-authentication-dss-extension.zip**

Unzip the artefact and copy the extensions directory under ${domibus.config.location}

21.2.3. Configure proxy

In order to refresh the EU LOTL, DSS needs to connect to the Internet. No white list can be configured at the proxy level, as changes in EU LOTL are dynamic. Therefore the DSS extension needs dynamic Internet access.

If a proxy is required, please configure the following properties within ${domibus.config.location}/extensions/config/authentication-dss-extension.properties:

```
# The https proxy host to use
#domibus.authentication.dss.proxy.https.host=

# The https proxy port to use
#domibus.authentication.dss.proxy.https.port=

# The https proxy user to use
#domibus.authentication.dss.proxy.https.user=

# The https proxy password to use
#domibus.authentication.dss.proxy.https.password=

# The https proxy excluded hosts. Allows multiple urls (separator ',', ';' or ' ')
#domibus.authentication.dss.proxy.https.excludedHosts=

# The http proxy host to use
#domibus.authentication.dss.proxy.http.host=

# The http proxy port to use
#domibus.authentication.dss.proxy.http.port=

# The http proxy user to use
#domibus.authentication.dss.proxy.http.user=

# The http proxy password to use
#domibus.authentication.dss.proxy.http.password=

# The http proxy excluded hosts. Allows multiple urls (separator ',', ';' or ' ')
#domibus.authentication.dss.proxy.http.excludedHosts=
```

Note: If the proxy server needs TLS authentication, please add the CA certificate of the proxy server in the java cacert or in the dss-tls-truststore described below.

21.2.4. DSS extension truststores

The DSS extension uses truststores for two reasons:

- Store TLS certificates of servers containing the trusted lists to download.
- Store public certificates to verify the xml signature of the trusted lists.

Separate truststores are used for xml signature verification and tls. The Dss extension distribution is provided with two truststores:

dss-tls-truststore.p12

Any extra TLS certificate (not present in the java cacert) needed to download custom or official trusted lists should be installed in the dss-tls-truststore.

ojkeystore.p12

The EU LOTL downloaded by DSS is signed, and to verify the signature, a truststore containing public certificates located at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG needs to be configured. Those certificates are packaged in the ojkeystore.

In case LOTL signing certificates need to be upgraded, please copy them from above url and add them to the ojkeystore.p12.

Please note, that if the DSS extension is configured with custom trusted lists, a third truststore should be configured to check the custom trusted list signature.

21.2.5. Configure LOTL truststore

Please copy truststore\ojkeystore.p12 to ${domibus.config.location}/keystores directory.

Please copy truststore\dss-tls-truststore.p12 to ${domibus.config.location}/keystores directory and add any required TLS certificate to it.

21.2.6. Configure custom trusted list

If a certificate chain with a CA not present in the LOTL needs to be used, DSS offers the possibility to configure custom trusted list. Please refer to the DSS documentation.

If a custom trusted list is required, please configure the following properties within ${domibus.config.location}/extensions/config/authentication-dss-extension.properties:

```
# Following properties should be used to add custom trusted list.
# Custom trusted list url
# domibus.authentication.dss.custom.trusted.lists.list1.url=
# Path of the keystore containing the certificate used to sign the custom trusted list
#domibus.authentication.dss.custom.trusted.list.keystore.path=

# The Keystore type
#domibus.authentication.dss.custom.trusted.list.keystore.type=

# The Keystore password
#domibus.authentication.dss.custom.trusted.list.keystore.password=
```

If multiple custom trusted lists are needed, please add the new url and increment the list number.

E.g.

```
# Custom trusted list url
#domibus.authentication.dss.custom.trusted.lists.list2.url=
```

As for EU LOTL, custom trusted lists are signed and DSS will verify the signature of the custom trusted lists before using it.

Please use domibus.authentication.dss.custom.trusted.list.keystore.path/type/password to configure a truststore containing the certificate needed to verify the custom trusted list signature. The recommendation is to add the custom trusted list truststore under ${domibus.config.location}/keystores.

21.2.7. Configure PMode policy

To perform certificate validation, the DSS extension expects to find the full signing certificate chain within the incoming AS4 message. To do so, Domibus should be configured with a security policy configured with WssX509PkiPathV1Token11 as described in the WS-SecurityPolicy document.

Remarks:

At startup, DSS generates stacktraces due to 2 old certificates which are wrongly encoded.
To avoid the exceptions, please configure your logger for the “eu.europa.esig.dss.tsl.service.TSLParser” class accordingly.

21.2.8. Dss extension activation

In order to activate the DSS extension, please configure the following property:

domibus.extension.iam.authentication.identifier=DSS_AUTHENTICATION_SPI within ${domibus.config.location}/domibus.properties

21.3. DSS extension properties

Configuration Property	Default value	Purpose
domibus.authentication.dss.official.journal.content.keystore.type	PKCS12	Type of keystore containing the public certificate needed to validate the trusted list.
domibus.authentication.dss.official.journal.content.keystore.path	${domibus.config.location}/keystores/ojkeystore.p12	Path of the keystore containing the public certificate needed to validate the trusted list.
domibus.authentication.dss.official.journal.content.keystore.password	dss-password	Password of the keystore containing the public certificate needed to validate the trusted list.
domibus.authentication.dss.current.official.journal.url	https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG	Url: Official Journal URL where the EU trusted certificates are listed.
domibus.authentication.dss.current.lotl.url	https://ec.europa.eu/tools/lotl/eu-lotl.xml	Official EU URL of the list of trusted lists.

domibus.authentication.dss.lotl.country.code	EU	List of trusted list main code.
domibus.authentication.dss.lotl.root.scheme.info.uri	https://ec.europa.eu/information_society/policy/esignature/trusted-list/tl.html	Schema used to verify the OJ validity.
domibus.authentication.dss.cache.path	${domibus.config.location}/extensions/cache/dss/	Path where trusted lists are cached.
domibus.authentication.dss.refresh.cron	0 0 0/3 * * ?	Cron expression used to schedule DSS trusted list refresh. Default is every 3h.
domibus.authentication.dss.full.tls.refresh	false	If this property is true, the TL refresh job will force delete on all Trusted lists and download them again.
domibus.authentication.dss.constraints.constraint1.name	BBB_XCV_CCCBB	Name of the first constraint that will be validated against the DSS validation report. BBB_XCV_CCCBB checks whether the certificate chain can be built till the trust anchor.
domibus.authentication.dss.constraints.constraint1.status	OK	Constraint status needed to validate the certificate.
domibus.authentication.dss.constraints.constraint2.name		Empty value, giving the possibility to make a second DSS constraint validation.
domibus.authentication.dss.constraints.constraint2.status		Constraint status needed to validate the certificate.
domibus.authentication.dss.constraint1.name	BBB_XCV_ICTIVRSC	Name of the second constraint that will be validated against the DSS validation report. BBB_XCV_ICTIVRSC checks whether the current time is in the validity range of the signer's certificate.
domibus.authentication.dss.constraint1.status	OK	Constraint status needed to validate the certificate.
domibus.authentication.dss.enable.custom.trusted.list.for.multitenant	false	In multi-tenant configuration, custom DSS trusted lists are shared by all tenants. Therefore they are deactivated by default.
domibus.authentication.dss.exception.on.missing.revocation.data	false	Trigger an exception when no revocation data is accessible.
domibus.authentication.dss.check.revocation.for.untrusted.chains	false	Execute revocation check when anchor cannot be found.

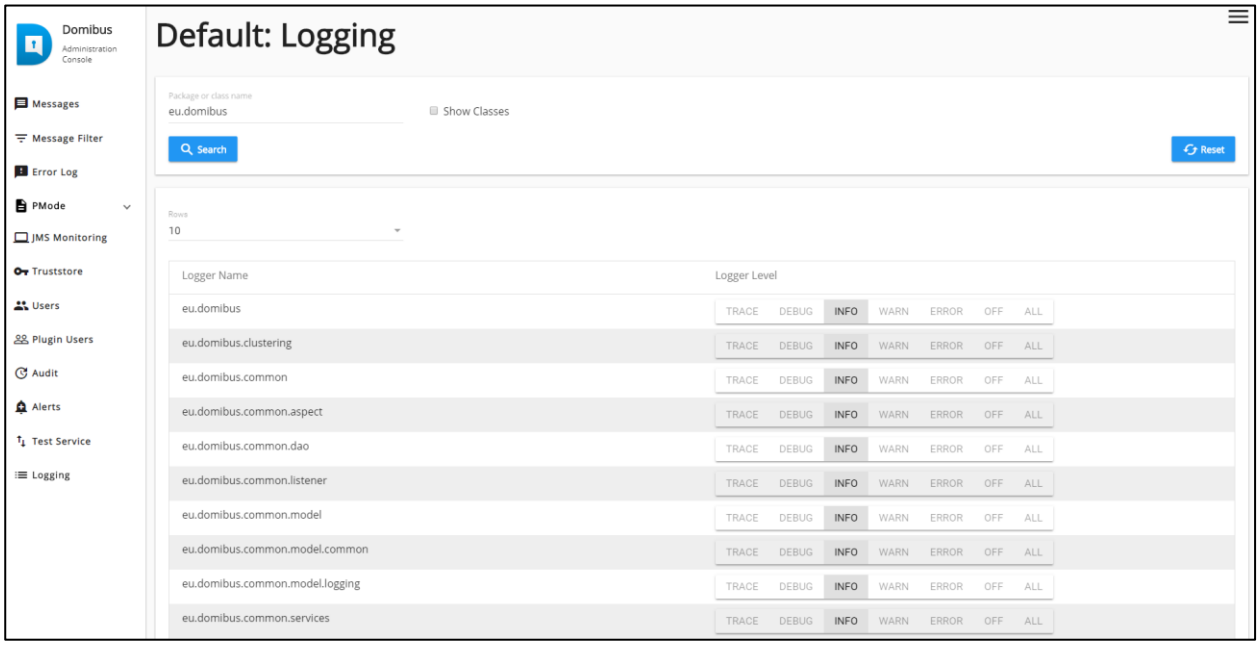
domibus.authentication.dss.custom.trusted.lists.list1.url=		Following properties should be used to add the first custom trusted list URL.
domibus.authentication.dss.custom.trusted.lists.list1.url		Following properties should be used to add the second custom trusted list URL.
domibus.authentication.dss.custom.trusted.lists.list3.url		Following properties should be used to add the third custom trusted list URL.
domibus.authentication.dss.custom.trusted.lists.list3.code		Following properties should be used to add the third custom trusted list code.
domibus.authentication.dss.custom.trusted.lists.list2.code		Following properties should be used to add the second custom trusted list code.
domibus.authentication.dss.custom.trusted.list.keystore.path		Path of the keystore containing the certificate used to sign the custom trusted list.
domibus.authentication.dss.custom.trusted.list.keystore.type		The custom trusted list Keystore type.
domibus.authentication.dss.custom.trusted.list.keystore.password		The custom trusted list Keystore password.
domibus.authentication.dss.proxy.https.host		The https proxy host to use.
domibus.authentication.dss.proxy.https.port		The https proxy user to use.
domibus.authentication.dss.proxy.https.user		The https proxy password to use.
domibus.authentication.dss.proxy.https.excludedHosts		The https proxy excluded hosts. Allows multiple URL's (separator ',', ';' or ' ').
domibus.authentication.dss.proxy.http.host		The http proxy host to use.
domibus.authentication.dss.proxy.http.port		The http proxy port to use.
domibus.authentication.dss.proxy.http.user		The http proxy user to use.
domibus.authentication.dss.proxy.http.password		The http proxy password to use.
domibus.authentication.dss.proxy.http.excludedHosts		The http proxy excluded hosts. Allows multiple URL's (separator ',', ';' or ' ').
domibus.authentication.dss.cache.name	dss-cache	Name of the ehcache configured for DSS.

domibus.dss.ssl.trust.store.path	${domibus.config.location}/keystores/dss-tls-truststore.p12	TLS truststore for dss dataloader. Should contain all the TLS certificates needed to download the EU LOTL and Custom trusted lists.
domibus.dss.ssl.trust.store.password	dss-tls	TLS truststore password for dss dataloader
domibus.dss.ssl.trust.store.type	JKS	TLS truststore type dss dataloader.
domibus.dss.perform.crl.check	False	Perform crl check within dss. False by default as it is perfomed by domibus.
domibus.dss.data.loader.socket.timeout	3000	Domibus data loader socket timeout in milliseconds.
domibus.dss.data.loader.connection.timeout	3000	Domibus data loader connection time out in milliseconds.

22. SETTING LOGGING LEVELS AT RUNTIME

22.1. Description

Admin and Super admin users can change the Logging levels at runtime for the Domibus application using the Admin Console 'Logging' menu:



Input elements include:

- A **Search box** where the user could freely enter the name of the package of classes desired to set the logging level. By default this is populated with 'eu.domibus' value.

  Note that wildcards are not accepted like 'domi*' are not recognised. Users must enter the full description of the item to be searched (e.g:'domibus' or 'apache')

- A **Show classes** check box allows level setting for each package. See the next picture

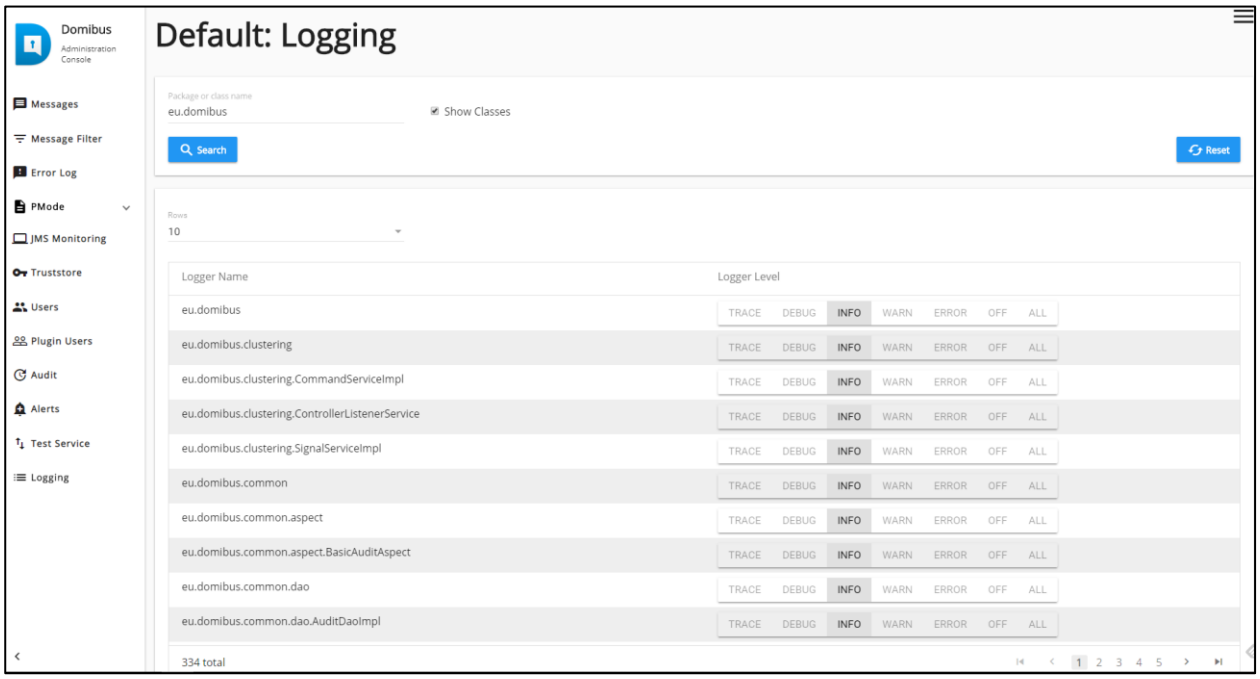
- A **Reset button** will reset all logging levels to the default values defined in logback.xml

- **Pagination** controls to change the number of rows to be shown per page

Remark:

- *In a multi-tenant environment, the names of the loggers are prefixed with the name of the tenant for which the selected logging level should be applied. For example, if the server has two tenants named tenantOne and tenantTwo, the user will be able to configure these loggers separately: "tenantOne.eu.domibus" and "tenantTwo.eu.domibus".*

- *Changing the logging levels only affects the currently running instance of Domibus and will not change or update the existing logging configuration file (logback.xml).*


Domibus
Administration Console
Messages
Message Filter
Error Log
PMode
JMS Monitoring
Truststore
Users
Plugin Users
Audit
Alerts
Test Service
Logging
Default: Logging
Package or class name
eu.domibus
Show Classes
Search
Reset
Rows
10
Logger Name
Logger Level
eu.domibus
eu.domibus.clustering
eu.domibus.clustering.CommandServiceImpl
eu.domibus.clustering.ControllerListenerService
eu.domibus.clustering.SignalServiceImpl
eu.domibus.common
eu.domibus.common.aspect
eu.domibus.common.aspect.BasicAuditAspect
eu.domibus.common.dao
eu.domibus.common.dao.AuditDaoImpl
TRACE DEBUG INFO WARN ERROR OFF ALL
334 total
1 2 3 4 5

23. EU LOGIN INTEGRATION

23.1. Description

Domibus is configured by default to use its own database for user authentication and authorization, as seen in previous chapters.

But Domibus could also be configured and installed to use EU Login for user authentication and authorization (even if this is not provided by default by EU Login).

EU Login[1] is the European Commission's central user authentication service. It allows authorised users to access a wide range of Commission resources, including websites, applications and services, using a single sign on based on (EC) email address, password, and if required, additional authenticating factors. More details could be found on internal Confluence page by clicking the following link: https://webgate.ec.europa.eu/CITnet/confluence/pages/viewpage.action?pageId=24641907

Domibus with EU Login integration is available only for Weblogic server.

23.2. Installation and Configuration

23.2.1. Installation

For installation of Domibus with EU Login, please follow the steps below:

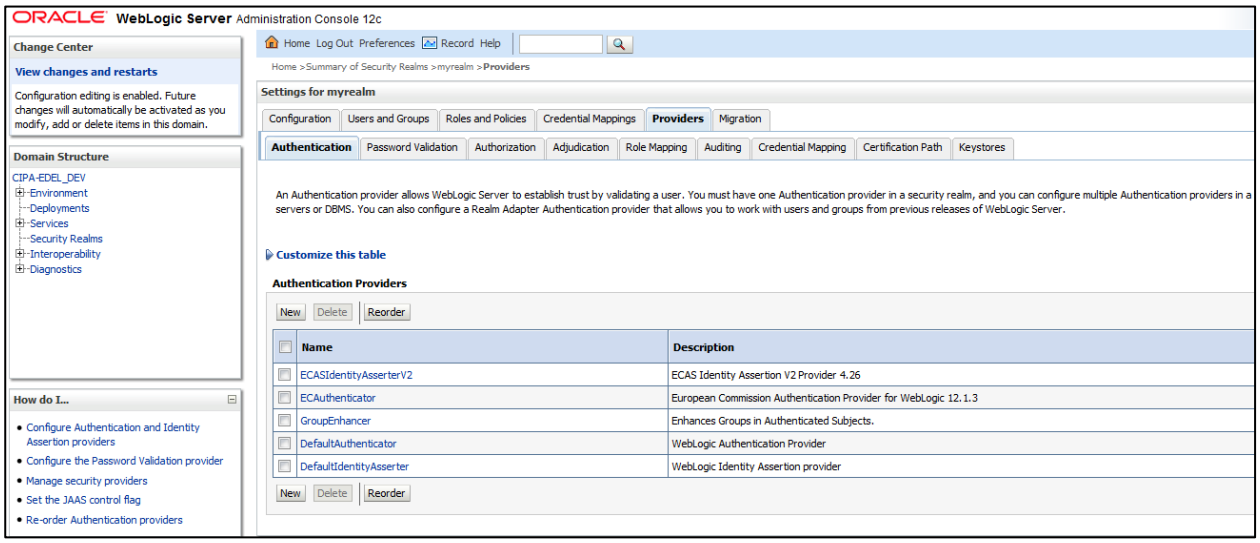
a. create DB schemas as per previous chapters for a single tenancy or Multitenancy installation

b. download domibus-msh-distribution-xyz-weblogic-ecas-configuration.zip and domibus-msh-distribution-xyz-weblogic-ecas-war.zip

c. install and configure Domibus war and configuration files into Weblogic server – follow Weblogic guidelines as per previous chapters

d. check that the WebLogic server has latest compatible ECASIdentityAsserter installed: go to the Weblogic Server console -> Security Realms -> myrealm -> Providers:

[1] Click here for more information on EU Login.



e. Configure ecas-config-domibus.xml file and install it in the classpath of Weblogic server

An example of ecas-config-domibus.xml file is as shown below:

```
<client-config xmlns="https://www.cc.cec/cas/schemas/client-config/ecas/1.8"
        xmlns:cas="https://www.cc.cec/cas/schemas/client-config/cas/2.0">


  <ecasBaseUrl>https://ecasa.cc.cec.eu.int:7002</ecasBaseUrl>

  <groups>
    <group>*</group>
  </groups>

  <acceptStrengths>
    <strength>STRONG</strength>
                <strength>STRONG_SMS</strength>
    <strength>CLIENT_CERT</strength>
  </acceptStrengths>

  <assuranceLevel>LOW</assuranceLevel>
  <!-- renew is false only for local in order to speedup the development-->
  <cas:renew>true</cas:renew>
  <requestingUserDetails>true</requestingUserDetails>

</client-config>
```

For more details about steps d. and e., please refer to EU Login documentation in the Confluence pages provided above.

23.2.2. Configuration

When a user is authenticated against EU Login he or she has specific LDAP groups associated with him/her. These groups will be used for Domibus to map:

- User roles: AP_ADMIN, ADMIN and USER

- Default domain

The mapping of these groups is performed in **domibus.properties** which needs to be changed accordingly. Look for the section related to EU Login mappings and update it:

domibus.security.ext.auth.provider.group.prefix=DIGIT_DOM

This is the prefix of EU Login LDAP groups that Domibus will take into account.

domibus.security.ext.auth.provider.user.role.mappings=DIGIT_DOMRUSR=ROLE_USER;DIGIT_DOMRADM=ROLE_ADMIN;DIGIT_DOMRSADM=ROLE_AP_ADMIN;

This property will map each EU Login LDAP group to a corresponding Domibus user role. If one user has more than one LDAP group/role associated, the role with the broader rights will be chosen.

domibus.security.ext.auth.provider.domain.mappings=DIGIT_DOMDDOMN1=domain1;

This property will map an EU Login LDAP group to a Domibus domain: it is useful in a Multitenancy installation, as in single tenancy all users are mapped to Default domain.

If the current user has no roles/LDAP groups or domain associated, he/she could still authenticate but he or she will not have the privileges to use the Domibus console.

```
# ---------------------- EU Login mappings -------------------------------------
# all EU Login groups used by Domibus should have this prefix
domibus.security.ext.auth.provider.group.prefix=DIGIT_DOM

# pairs of strings separated by semicolons to map Domibus user roles to EU Login LDAP groups
# the format is
LDAP_GROUP_USER=ROLE_USER;LDAP_GROUP_ADMIN=ROLE_ADMIN;LDAP_GROUP_AP_ADMIN=ROLE_AP_ADMIN;
# last semicolon is mandatory
domibus.security.ext.auth.provider.user.role.mappings=DIGIT_DOMRUSR=ROLE_USER;DIGIT_DOMRADM=ROLE_ADMIN;DIGIT_DOMRSADM=ROLE_AP_ADMIN;

# pairs of strings separated by semicolons to map Domibus domain codes to EU Login LDAP groups
# the format is LDAP_GROUP_DOMAIN1=domain1;LDAP_GROUP_DOMAIN2=domain2;
# last semicolon is mandatory
domibus.security.ext.auth.provider.domain.mappings=DIGIT_DOMDDOMN1=domain1;
```

23.3. Domibus UI changes

When the user first tries to access Domibus at the address http://server:port/domibus, he/she will be redirected to the EU Login page where he/she will fill in the username and password. After successfully entering his/her credentials, he/she will be redirected to the Domibus User Interface.

Non super administrator users that can manage multiple domains will see a domain dropdown that will allow them to switch between all their available domains.

His/her username will appear on the right corner on the Domibus Admin console but the some options will be greyed out (not accessible anymore):

- Change Password (from top right menu), as the password change is managed by the EU Login

- Users (from left menu): adding or editing existing users will not be possible

24. DOMIBUS STATISTICS

Dropwizard library has been added to Domibus allowing administrators to monitor Domibus with JVM and custom metrics.

24.1. Metrics type

24.1.1. JVM metrics

Memory usage

A set of gauges for JVM memory usage, including stats on heap vs. non-heap memory, plus GC-specific memory pools.

Memory metrics can be added or removed by modifying the following domibus property:

```
#Activate drop wizard memory metrics
domibus.metrics.monitor.memory=true
```

Garbage collector

Contains a set of gauges for the counts and elapsed times of garbage collections.

Garbage collector metrics can be added or removed by modifying the following domibus property:

```
#Activate drop wizard gc metrics
domibus.metrics.monitor.gc=true
```

Threads

Thread metrics can be added or removed by modifying the following domibus property:

```
#Activate drop wizard cached threads metrics
domibus.metrics.monitor.cached.threads=true
```

24.1.2. Custom metrics

Custom metrics to monitor messages exchange are also available for the following flows:

- Incoming UserMessage
- Incoming UserMessage receipt
- Incoming PullRequest
- Incoming PullRequest receipt
- Outgoing UserMessage
- Outgoing PullRequest

- Outgoing PullRequest receipt

  Each of them will have a Dropwizard counter and timer metrics configuration. Please refer to Dropwizard documentation. (https://metrics.dropwizard.io/3.1.0/manual/core/#timers, https://metrics.dropwizard.io/3.1.0/manual/core/#counters).

24.1.3. JMS Queues count metrics

This metrics will monitor the count of JMS queues.

In order to enable it, please set the following Domibus property to true:

```
#Activate drop wizard JMS Queues metrics
domibus.metrics.monitor.jms.queues=true
```

The following property will establish the interval (in seconds) upon which the JMS count are recalculated:

```
# how long (in seconds) the JMS count will be cached
# defaults to 0 - the count isn't cached
domibus.metrics.monitor.jms.queues.refresh.period=0
```

The last property to set: by default only DLQ queue count is shown. Set to false to add metrics for all JMS queues:

```
# show counts only for DLQ queue
domibus.metrics.monitor.jms.queues.show.dlq.only=true
```

24.2. Metrics access

24.2.1. Log file

In order to log the metrics under the statistics.log file, please set the following property to true (default):

```
#Enable slf4j reporter for dropwizard metrics.
domibus.metrics.slf4j.reporter.enable=true
```

In case of upgrade, please follow the upgrade procedure to add the relevant appender and logger within the logback.xml file.

24.2.2. Servlet

Statistics can also be visualized within the browser under the following URL, once the user has authenticated:

<server url>/domibus/metrics

The types of metrics available can be seen in this example JSON, generated afer receiving 3 messages on C2:


```
{
  "version" : "4.0.0",
```

```
 "gauges" : { },
 "counters" : {
  "eu.domibus.core.ebms3.sender.AbstractUserMessageSender.outgoing_user_message.counter" : {
   "count" : 0
  },
  "eu.domibus.core.ebms3.sender.MessageSenderListener.onMessage.counter" : {
   "count" : 0
  },
  "eu.domibus.core.ebms3.sender.MessageSenderService.sendUserMessage.counter" : {
   "count" : 0
  },
  "eu.domibus.core.ebms3.sender.client.MSHDispatcher.dispatch.counter" : {
   "count" : 0
  },
  "eu.domibus.core.message.MessagingServiceImpl.storeMessage.counter" : {
   "count" : 0
  },
  "eu.domibus.core.message.UserMessageHandlerServiceImpl.persistSentMessage.counter" : {
   "count" : 0
  },
  "eu.domibus.core.message.retention.MessageRetentionDefaultService.retention_deleteExpiredMessages.counter" : {
   "count" : 0
  },
  "eu.domibus.core.participant.FinalRecipientDao.findEndpointUrl.counter" : {
   "count" : 0
  },
  "eu.domibus.core.plugin.handler.DatabaseMessageHandler.submit.counter" : {
   "count" : 0
  },
  "eu.domibus.core.plugin.notification.BackendNotificationService.notifyOfMessageStatusChange.counter" : {
   "count" : 0
  },
  "eu.domibus.core.plugin.notification.PluginAsyncNotificationListener.onMessage.counter" : {
   "count" : 0
  },
  "eu.domibus.core.pmode.validation.validators.MessagePropertyValidator.submissionValidate.counter" : {
   "count" : 0
  }
 },
 "histograms" : { },
 "meters" : { },
 "timers" : {
  "eu.domibus.core.ebms3.sender.AbstractUserMessageSender.outgoing_user_message.timer" : {
   "count" : 3,
   "max" : 1.7963081150000002,
   "mean" : 0.7172844155520971,
   "min" : 0.190011653,
   "p50" : 0.304142917,
   "p75" : 1.7963081150000002,
   "p95" : 1.7963081150000002,
   "p98" : 1.7963081150000002,
   "p99" : 1.7963081150000002,
   "p999" : 1.7963081150000002,
   "stddev" : 0.7116186875056694,
   "m15_rate" : 0.18606482682841233,
   "m1_rate" : 0.06769308502134848,
   "m5_rate" : 0.16103966480361412,
   "mean_rate" : 0.035642575683050286,
   "duration_units" : "seconds",
   "rate_units" : "calls/second"
  },
  "eu.domibus.core.ebms3.sender.MessageSenderListener.onMessage.timer" : {
   "count" : 3,
   "max" : 1.9540381580000001,
   "mean" : 0.8001299801693523,
   "min" : 0.228544151,
   "p50" : 0.365267078,
   "p75" : 1.9540381580000001,
   "p95" : 1.9540381580000001,
   "p98" : 1.9540381580000001,
   "p99" : 1.9540381580000001,
   "p999" : 1.9540381580000001,
   "stddev" : 0.7614390255941889,
```

```
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.03557590138340944,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.ebms3.sender.MessageSenderService.sendUserMessage.timer" : {
    "count" : 3,
    "max" : 1.942905847,
    "mean" : 0.7966011396382428,
    "min" : 0.228307575,
    "p50" : 0.36503053,
    "p75" : 1.942905847,
    "p95" : 1.942905847,
    "p98" : 1.942905847,
    "p99" : 1.942905847,
    "p999" : 1.942905847,
    "stddev" : 0.7564498977737351,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.035580559611544674,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.ebms3.sender.client.MSHDispatcher.dispatch.timer" : {
    "count" : 3,
    "max" : 0.8656259420000001,
    "mean" : 0.295635655301653,
    "min" : 0.040702159,
    "p50" : 0.056150771,
    "p75" : 0.8656259420000001,
    "p95" : 0.8656259420000001,
    "p98" : 0.8656259420000001,
    "p99" : 0.8656259420000001,
    "p999" : 0.8656259420000001,
    "stddev" : 0.37512321358066775,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.03594801379121541,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.message.MessagingServiceImpl.storeMessage.timer" : {
    "count" : 3,
    "max" : 0.026732020000000002,
    "mean" : 0.012253196838249663,
    "min" : 0.0051686480000000005,
    "p50" : 0.0069916810000000005,
    "p75" : 0.026732020000000002,
    "p95" : 0.026732020000000002,
    "p98" : 0.026732020000000002,
    "p99" : 0.026732020000000002,
    "p999" : 0.026732020000000002,
    "stddev" : 0.009453424577937073,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.035416798866656655,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.message.UserMessageHandlerServiceImpl.persistSentMessage.timer" : {
    "count" : 3,
    "max" : 0.26435188600000004,
    "mean" : 0.08830605751878227,
    "min" : 0.011123899000000001,
    "p50" : 0.016145264,
    "p75" : 0.26435188600000004,
    "p95" : 0.26435188600000004,
    "p98" : 0.26435188600000004,
```

```
    "p99" : 0.26435188600000004,
    "p999" : 0.26435188600000004,
    "stddev" : 0.11458684335112779,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.03542814747227987,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.message.retention.MessageRetentionDefaultService.retention_deleteExpiredMessages.timer" : {
    "count" : 2,
    "max" : 0.39966172,
    "mean" : 0.12145134037886698,
    "min" : 0.008339441000000001,
    "p50" : 0.008339441000000001,
    "p75" : 0.39966172,
    "p95" : 0.39966172,
    "p98" : 0.39966172,
    "p99" : 0.39966172,
    "p999" : 0.39966172,
    "stddev" : 0.17739477011981494,
    "m15_rate" : 0.1810306636758757,
    "m1_rate" : 0.0499815368634729,
    "m5_rate" : 0.1486564048912939,
    "mean_rate" : 0.019115378637402845,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.participant.FinalRecipientDao.findEndpointUrl.timer" : {
    "count" : 3,
    "max" : 0.014485389000000001,
    "mean" : 0.006205980140272974,
    "min" : 0.002342032,
    "p50" : 0.002979658,
    "p75" : 0.014485389000000001,
    "p95" : 0.014485389000000001,
    "p98" : 0.014485389000000001,
    "p99" : 0.014485389000000001,
    "p999" : 0.014485389000000001,
    "stddev" : 0.005435187471446011,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.03594146185268624,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.plugin.handler.DatabaseMessageHandler.submit.timer" : {
    "count" : 3,
    "max" : 2.035293721,
    "mean" : 0.6650845139203682,
    "min" : 0.076706179,
    "p50" : 0.092151826,
    "p75" : 2.035293721,
    "p95" : 2.035293721,
    "p98" : 2.035293721,
    "p99" : 2.035293721,
    "p999" : 2.035293721,
    "stddev" : 0.8917317293691389,
    "m15_rate" : 0.1850397103474366,
    "m1_rate" : 0.06271339961046699,
    "m5_rate" : 0.15842190815053878,
    "mean_rate" : 0.03472384283942923,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
  },
  "eu.domibus.core.plugin.notification.BackendNotificationService.notifyOfMessageStatusChange.timer" : {
    "count" : 6,
    "max" : 0.141240953,
    "mean" : 0.048801831975242856,
    "min" : 0.002696167,
    "p50" : 0.026940219,
```

```
    "p75" : 0.08250924100000001,
    "p95" : 0.141240953,
    "p98" : 0.141240953,
    "p99" : 0.141240953,
    "p999" : 0.141240953,
    "stddev" : 0.04690508878004811,
    "m15_rate" : 0.37212965365682465,
    "m1_rate" : 0.13538617004269696,
    "m5_rate" : 0.32207932960722824,
    "mean_rate" : 0.07091295935446128,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
   },
   "eu.domibus.core.plugin.notification.PluginAsyncNotificationListener.onMessage.timer" : {
    "count" : 3,
    "max" : 0.031067928,
    "mean" : 0.011473757179216823,
    "min" : 0.0027278890000000003,
    "p50" : 0.003280162,
    "p75" : 0.031067928,
    "p95" : 0.031067928,
    "p98" : 0.031067928,
    "p99" : 0.031067928,
    "p999" : 0.031067928,
    "stddev" : 0.012895512138375831,
    "m15_rate" : 0.18606482682841233,
    "m1_rate" : 0.06769308502134848,
    "m5_rate" : 0.16103966480361412,
    "mean_rate" : 0.03641137464343127,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
   },
   "eu.domibus.core.pmode.validation.validators.MessagePropertyValidator.submissionValidate.timer" : {
    "count" : 3,
    "max" : 0.004634629,
    "mean" : 0.0013561917783451183,
    "min" : 8.16E-6,
    "p50" : 9.29E-6,
    "p75" : 0.004634629,
    "p95" : 0.004634629,
    "p98" : 0.004634629,
    "p99" : 0.004634629,
    "p999" : 0.004634629,
    "stddev" : 0.002101785471641864,
    "m15_rate" : 0.1850397103474366,
    "m1_rate" : 0.06271339961046699,
    "m5_rate" : 0.15842190815053878,
    "mean_rate" : 0.03472885660070646,
    "duration_units" : "seconds",
    "rate_units" : "calls/second"
   }
  }
}
```


Gauges

The size of the resources such as the JMS queues. If the property domibus.metrics.monitor.jms.queues.show.dlq.only is true, then only the queues whose names contain the string DLQ will be considered. Otherwise all destinations will be shown, except the ones listed in eu.domibus.core.jms.JMSManagerImpl#SKIP_QUEUE_NAMES.

Note that in a cluster environment, only the queues that belong the the current node will be monitored. The user should check the metrics on each managed server to get a full picture.

The gauges refresh interval is configured by the property domibus.metrics.monitor.jms.queues.refresh.period: when this is set to 0, the queues will be assessed for each request.

Counters

With this metric, you can count how many threads are currently executing a given method that was annotated with eu.domibus.core.metrics.Counter.

Example: when dealing with 2 outgoing messages at the same time, during the processing the counter eu.domibus.core.ebms3.sender.MessageSenderListener.onMessage.counter is 2 and once the processing is done, the count is 0.

Timers

Timer implementation uses dropwizard timer that measures the methods annotated with eu.domibus.core.metrics.Timer.

A timer measures both the rate that a particular piece of code is called and the distribution of its duration. Please review the documentation on the dropwizzard framework portal: https://metrics.dropwizard.io/3.1.0/manual/core/#timers .

The timer will wrap a method execution and give you the following information about its execution:

```
count : 23 = number of time the method has been executed since the
server started.
      max : 1.178201075 = max time execution for the method
      mean : 0.10270451496827847 = mean time execution for the method
      min : 0.001535626 = min time execution for the method
      p50 : 0.050753843 = execution time in 50% of the cases
      p75 : 0.065817394 = execution time in 25% of the cases
      p95 : 0.165156044 = execution time in 5% of the cases
      p98 : 1.178201075 = execution time in 2% of the cases
      p99 : 1.178201075 = execution time in 1% of the cases
      p999 : 1.178201075 = execution time in 0.1% of the cases
stddev : 0.22737038231350143 = Execution standard deviation
      m15_rate : 3.9591133710591 = 15 minutes mean average number
      of execution
      m1_rate : 3.431343092532076 = 1 minute mean average number  of
execution
      m5_rate : 3.8786722524995447 = 5 minutes mean average
      number of execution
      mean_rate : 1.3049182430991542 = 1 second mean average
      number of execution
      duration_units" : "seconds",
 rate_units : "calls/second"
```

The following counters and timers are currently defined:

Class	Name	Description of what it counts or what it times
ReceiptDao	deleteMessages	Deleting a batch of Receipt messages
SignalMessageDao	deleteMessages	Deleting a batch of Signal messages
SignalMessageRawEnvelopeDao	deleteMessages	Deleting a batch of Signal Message Raw entries
UserMessageDao	deleteMessages	Deleting a batch of User Messages
UserMessageRawEnvelopeDao	deleteMessages	Deleting a batch of User Message Raw entries
MessageAttemptDao	deleteMessages.deleteAttemptsByMessageIds	Deleting a batch of Message Attempt entries
ErrorLogServiceImpl	deleteMessages.deleteErrorLogsByMessageIdInError	Deleting a batch of Error Log entries
MessageAcknowledgementDao	deleteMessages.deleteMessageAcknowledgementsByMessageIds	Deleting a batch of Message Acknowledgements
SignalMessageLogDao	deleteMessages.deleteMessageLogs	Deleting a batch of Signal Message Log entries
UserMessageLogDao	deleteMessages.deleteMessageLogs	Deleting a batch of User Message Log entries
JMSPluginImpl	deliverMessage	The method in the JMS plugin that delivers the UserMessage to backend.
MSHDispatcher	dispatch	Dispatching a message to another access point.
UserMessageDao	dropPartition	Dropping of a partition
EArchivingRetentionService	earchive_cleanStoredBatches	Cleaning the E-archiving storage for a domain
EArchiveBatchDispatcherService	earchive_createBatch	Running the E-archive batch for a domain and type

Class	Name	Description of what it counts or what it times
EARKSIPFileService	earchive_createDataFile	Writing an E-archive data file
EArchivingFileService	earchive_getArchivingFiles	Getting the archiving files for an entity id
EArchiveListener	earchive_process_1_batch	Running an E-archive batch for a batch id and entity id
EArchivingDefaultService	earchive1_getEArchiveBatch	Running an E-archive batch for a batch entity id
FileSystemEArchivePersistence	earchive2_createEArkSipStructure	Creating an E-archive structure for a batch of user messages
EArchivingFileService	earchive24_getBatchFileJson	Writting an E-archive batch JSON file
EArchivingDefaultService	earchive3_executeBatchIsExported	Exporting an E-archive batch
FinalRecipientDao	findEndpointUrl	Searching for a final recipient endpoint
UserMessageDao	findPotentialExpiredPartitions	Searching for potentially expired Oracle partitions
AS4ReceiptServiceImpl	generateReceipt	The first method to be called to generate and AS4 receipt.
UserMessageHandlerServiceImpl	handleNewUserMessage	The first method to be called to handle a new incoming UserMessage.
IncomingPullRequestHandler	incoming_pull_request	The first method to be called (after CXF interceptors) on an incoming PullRequest.
IncomingPullReceiptHandler	incoming_pull_request_receipt	Handling an incoming pull request receipt
MSHWebservice	incoming_user_message	The first method to be called (after CXF interceptors) on an incoming UserMessage or SignalMessage.

Class	Name	Description of what it counts or what it times
		This method in the entry point of the MSH endpoint.
IncomingUserMessageReceiptHandler	incoming_user_message_receipt	Handling an incoming user message receipt
SoapUtil	logMessage	Logging a SOAP message
BackendNotificationService	notifyMessageReceived	Notifying the backend of a new incoming message.
BackendNotificationService	notifyMessageReceivedFailure	Sending a backend notification for message receive failure
BackendNotificationService	notifyMessageResponseSent	Sending a backend notification for message receive failure
BackendNotificationService	notifyOfMessageStatusChange	Sending a backend notification for message receive failure
BackendNotificationService	notifyOfSendSuccess	Sending a notification of status change for a user message
PluginMessageSendSuccessNotifier	notifyPluginSendSuccess	Sending a notification for a successful sending of a user message
MessageSenderListener	onMessage	The method that picks a message from the JMS send queue to send the message to another access point.
PluginAsyncNotificationListener	onMessage	Sending a plugin notification about processing a message
RetentionListener	onMessage.deleteMessages	Deleting messages
PullReceiptListener	outgoing_pull_receipt	Sending a pull receipt for a

Class	Name	Description of what it counts or what it times
		pulled message
PullMessageSender	outgoing_pull_request	Processing a pull request
AbstractUserMessageSender	outgoing_user_message	The method used to send a message to another access point.
UserMessageHandlerServiceImpl	persistReceivedMessage	Validating and persisting a received message
MessageSubmitterHelper	persistSentMessage	Persists a sent user message and user message log
AbstractIncomingMessageHandler	processMessage	The first method to be called (after CXF interceptors) on an incoming UserMessage. It includes: the load of the policy, the validation of the parties, the validation of the certificates, the validation of the receipt.
JMSPluginImpl	receiveMessage	The method in the JMS plugin that receives a UserMessage from the backend
MessageRetentionDefaultService	retention_deleteExpiredMessages	Deleting and archiving expired messages based on the retention policy
MessageRetentionPartitionsService	retention_deleteExpiredMessages	Dropping a partition with no ongoing messages based on the retention policy
UserMessageDefaultService	scheduleSending	Sending user messages to the appropriate MSH JMS queue
JMSManagerImpl	sendMessageToQueue_map1	Sending a user message of type MAP_MESSAGE to the internal JMS queue

Class	Name	Description of what it counts or what it times
JMSManagerImpl	sendMessageToQueue_map2	Sending a user message of type MAP_MESSAGE to the internal JMS queue
JMSManagerImpl	sendMessageToQueue_text	Sending a user message of type TEXT_MESSAGE to the internal JMS queue
MessageSenderService	sendUserMessage	Validating and sending an AS4 message to C3
MessagingServiceImpl	storeMessage	Storing a received message or scheduling the storing of a message to be sent
MessagingServiceImpl	storePayloads	Storing the payloads of a message
MessagePropertyValidator	submissionValidate	Validating the message properties of a submission. (Sent by the backend)
DatabaseMessageHandler	submit	The first method to be called to validate and store a UserMessage coming from the backend plugin.
MessageSubmitterImpl	submit	Validating and submitting a message
MessagePropertyValidator	validate	Validating the message properties of a user message (received on the msh endpoint).

24.2.3. Jmx

To access the metrics via jmx, please set the following property to true:

```
#Enable jmx reporter for dropwizard metrics. The following warning:
#We do not recommend that you try to gather metrics from your production environment. JMX's RPC
```

```
API is fragile.
#For development purposes and browsing, though, it can be very useful.
domibus.metrics.jmx.reporter.enable=false
```

25. PAYLOAD ENCRYPTION

Data at rest is not encrypted by default in Domibus. This means that the payloads are stored in C2 exactly as they were received from C1. The same for payloads received from C2 and stored in C3.

The payloads stored in C2 and C3 are not accessible to third parties. Nevertheless, it is a good practice to encrypt the payloads to increase the security level.

Data at rest encryption can be activated using the property *domibus.payload.encryption.active=true.* Once activated, Domibus encrypts the payloads stored in C2 and C3 using symmetric encryption with *AES/GCM/NoPadding* algorithm. Domibus generates the symmetric key used to encrypt payloads the first time the payload encryption is activated. The generated symmetric key is stored in the Domibus database. A symmetric key is generated for each domain in case of multitenancy.

Encrypting data at rest is transparent for C1/C4, so if C4 downloads a message from C3, it will receive the payloads un-encrypted as they were sent by C1.

26. MESSAGE PRIORITIZATION

26.1. Introduction

When Domibus C2 receives concurrently from C1 a lot of *UserMessages* to be sent, it cannot keep the pace of sending *UserMessages* to C3. Consequently, JMS messages start accumulating in the *SendMessageQueue*.

As the JMS messages from the *SendMessageQueue* are processed in a random order, for some *UserMessages* there might be a big delay between the time C1 submits a message to C2 for sending and the actual sending of the *UserMessage* from C2 to C3.

Moreover, in some use cases there is a need to assign a high priority to some *UserMessages*. Due to their urgency, these high priority messages must be sent as soon as possible regardless of when they have been submitted to C2.

26.2. Solution overview

Domibus assigns a priority to each *UserMessage* based on service and action when the message is submitted by C1. All *UserMessages* are scheduled for sending using the existing *SendMessageQueue.*

There are two options for processing messages from the *SendMessageQueue:*

1. Using the underlying JMS infrastructure if it supports message priority on a message queue
2. Using dedicated JMS listeners (with a specific concurrency) for each configured message priority that consumes only JMS messages having the configured priority using a JMS selector. This solution can also take advantage on the JMS infrastructure support for message priority

26.3. Solution detail

Domibus C2 assigns a priority to each *UserMessage* it receives from C1 to implement message prioritization. The *UserMessage* priority is determined based on the service and action values of the *UserMessage*. The priority varies from 1 to 9, 1 for low priority messages and 9 for high priority messages.

For instance, for the following service and action values from the *UserMessage*:

```
<eb:CollaborationInfo>

    <eb:Service type="tc1">bdx:noprocess</eb:Service>

   <eb:Action>TC1Leg1</eb:Action>

</eb:CollaborationInfo>
```

In order to assign a priority to the above *UserMessage* a priority rule name must be defined first in *domibus.properties* configuration file.

For instance, one can define a priority rule named **medium**:

domibus.dispatcher.priority.medium=Medium priority messages

Once the rule name is defined, other properties, like service, action, priority and concurrency can be also defined using the rule name. As we will see in the next sections the *concurrency* property is optional. For instance:

domibus.dispatcher.priority.medium.service= bdx:noprocess

domibus.dispatcher.priority.medium.action= TC1Leg1, TC1Leg2, TC1Leg3

domibus.dispatcher.priority.medium.value=5

domibus.dispatcher.priority.medium.concurrency=10-15

The action property configured for a specific rule supports a list of action values separated by comma. The action property will match if any of the list of actions will match. In the example above we have configured for instance three actions values separated by commas.

When a *UserMessage* having a service/action combination is matching a service/action combination configured for a priority rule, the priority configured for the matching rule will be assigned to the *UserMessage.*

It is not mandatory to configure both service and action for a priority rule. Only the service or only the action can be configured, in which case the priority will match if the service or the action configured will match.

Note: service/action combinations configured for routing rules must be unique.

After the priority of the *UserMessage* has been determined, C2 schedules the *UserMessage* for sending it to C3. This is performed by sending a JMS message to the *SendMessageQueue* containing the message id of the message to be sent and the message priority using *JMSPriority* header. For the example above the priority assigned to the message will be 5.

Once the priority has been determined for each *UserMessage/JMSMessage*, there are two options for processing messages from the *SendMessageQueue:*

26.3.1. Using the underlying JMS infrastructure

This solution can be used if the underlying JMS infrastructure supports message priority on a message queue. Such infrastructure will guarantee the priority delivery of high priority messages using the *JMSPriority* header value. This approach is suited when there are low to medium number of high priority messages processed by the system.

In this case there is only one JMS listener that is consuming JMS messages from the *SendMessageQueue.* This JMS listener is the default listener that is used by Domibus to process all JMS messages from the *SendMessageQueue,* regardless if message prioritization is used. The default JMS listener can be configured in *domibus.properties* in the *Dispatcher* section.

Please find below an example about how to configure a rule for medium priority messages, the *concurrency* property is not used:

domibus.dispatcher.priority.medium.service= bdx:noprocess

domibus.dispatcher.priority.medium.action= TC1Leg1, TC1Leg2, TC1Leg3

domibus.dispatcher.priority.medium.value=5

In case the *SendMessageQueue* is flooded with high priority messages only the high priority messages will be consumed (most of JMS brokers will try to deliver high priority messages first), leaving the lower priority messages in the queue potentially for long periods of time (in extreme cases even days). For this specific case, the solution from the next section is more suited.

26.3.2. Using dedicated JMS listeners (with a specific concurrency) for each configured message priority

This approach is suited in case a finer level of granularity is desired to JMS message consumption or for tackling the case mentioned above when high priority messages are flooding the system. In this scenario we have a quality of service which gives a chance to lower priority messages to be consumed.

In this scenario, dedicated JMS listeners (with specific concurrency) consume only the JMS messages with a specific priority given by the *JMSPriority* header. This is performed using a JMS selector filtering messages for each configured message priority. This solution can also take advantage o the JMS infrastructure support for message priority.

At start up Domibus reads all the priority rules configured in *domibus.properties*. For each configured priority rule with a defined *concurrency* property, Domibus creates programmatically a JMS listener with a specific JMS selector listening to the *SendMessageQueue*.

Please find below an example about how to configure a rule for medium priority messages with a JMS selector, the *concurrency* property is mandatory:

domibus.dispatcher.priority.medium.service= bdx:noprocess

domibus.dispatcher.priority.medium.action= TC1Leg1, TC1Leg2, TC1Leg3

domibus.dispatcher.priority.medium.value=5

domibus.dispatcher.priority.medium.concurrency=10-15

Multiple JMS listeners are listening the *SendMessageQueue* using a JMS selector that takes into account the message priority. JMS listeners that are processing messages with high priority can have a higher concurrency assigned, meaning multiple threads are assigned to process concurrently high priority messages. This way high priority messages can be processed faster than messages with lower priority.

UserMessages not matching any priority rule will be scheduled on the same *SendMessageQueue* and handled by a default JMS listener configured with a specific concurrency. This way it is not mandatory to define a priority rule for all messages. The default JMS listener serves as a catch all messages if they do not match any priority rule.

To understand the solution, the following example contains a configuration with 3 JMS listeners for handling messages with low, medium and high priority:

#low priority

domibus.dispatcher.priority.low=Low priority messages

domibus.dispatcher.priority.low.service= service1

domibus.dispatcher.priority.low.action= action2

domibus.dispatcher.priority.low.value= 1

domibus.dispatcher.priority.low.concurrency=**2-5**

#medium priority

domibus.dispatcher.priority.medium=Medium priority messages

domibus.dispatcher.priority.medium.service= service2

domibus.dispatcher.priority.medium.action= action2

domibus.dispatcher.priority.medium.value= 4

domibus.dispatcher.priority.medium.concurrency=**10-15**

#high priority

domibus.dispatcher.priority.medium=High priority messages

domibus.dispatcher.priority.medium.service= service3

domibus.dispatcher.priority.medium.action= action3

domibus.dispatcher.priority.high.value= 9

domibus.dispatcher.priority.medium.concurrency=**30-50**

#default priority for messages not matching any priority rule above

domibus.dispatcher.concurency=5-20

27. SSL OFFLOADING

In this section you will find more details about how to configure SSL offloading and when to actually use it. SSL offloading only makes sense in the context of dispatching Domibus messages to secure endpoints (i.e. a receiving PMode party having its URL configured using the “https:// scheme, instead of the “http://” one).

When dispatching to a secure endpoint, Domibus creates a secure SSL connection to the receiving party within the application. This is sometimes not desired, for example in the case when Domibus is running behind a forward SSL proxy installed as a DMZ proxy. The DMZ proxy may handle connection from applications other that Domibus, making it the central node responsible for relaying communication outside the trusted network it is serving. In this scenario, the DMZ proxy is usually configured for setting the SSL connections itself, having all the required configuration like truststores deployed in it. This is problematic, since the SSL connection cannot be initiated twice: the creation of the SSL connection needs to be offloaded from Domibus to the DMZ proxy.

27.1. Configuration

In the current setup, Domibus uses CXF to dispatch messages between its corners - named further below as C2 and C2. Internally, CXF uses java.net.URL for creating the connection between C2 and C3, with the possibility to use an optional HTTP/SOCKS proxy.

In the case the receiving party has a secure HTTPS endpoint, the java.net.URL is responsible for creating the SSL socket and starting the SSL handshake (see Figure 9).

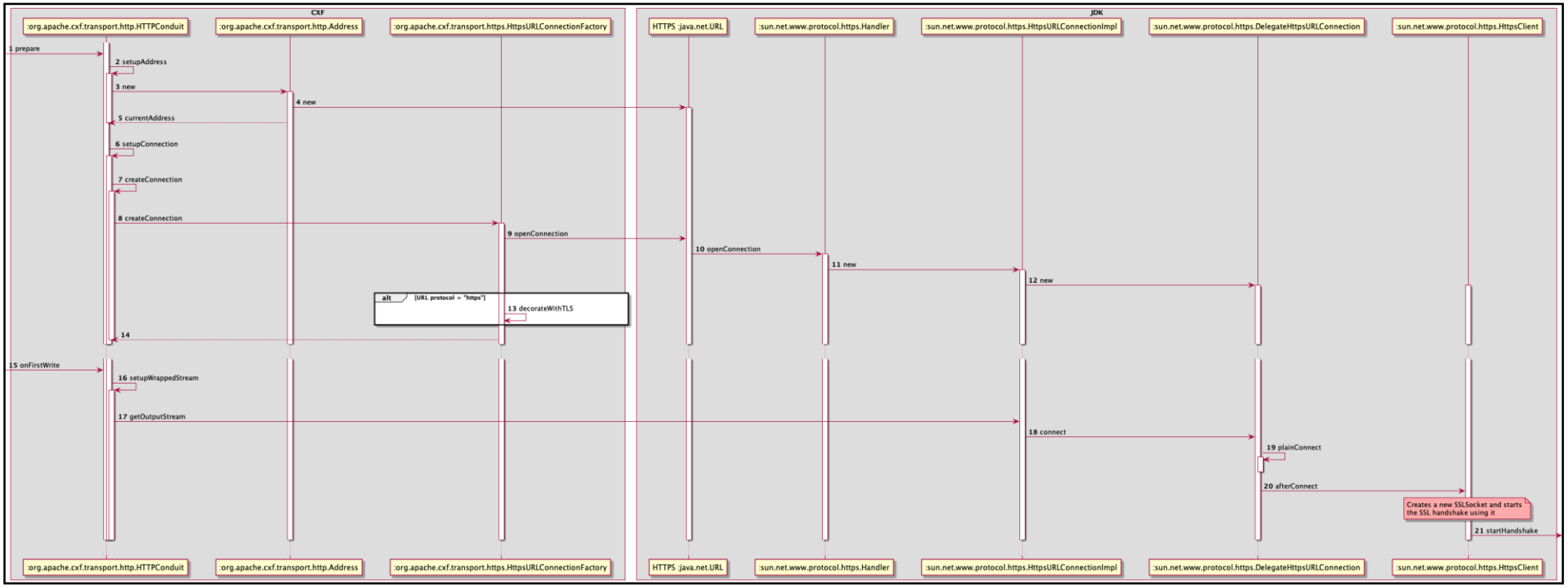


Figure 9 – SSL handshake

In order to offload the SSL to another application (e.g. SSL forward proxy), we need to prevent the SSL handshake to happen in Domibus, in the C2 initiating corner. A new domibus.connection.cxf.ssl.offload.enable Domibus property has been added to prevent this SSL handshake from happening within Domibus, even when the C3 endpoint uses an HTTPS URL.

When this parameter is set to true, Domibus will replace the default HTTPS URL with a URL created from the HTTP version of the endpoint address (see Figure 10). This new URL will create a plain HTTP connection and will not trigger an SSL handshake anymore.

In order to allow the SSL forward proxy to identify the correct endpoint address, the protocol of the new HTTP URL is set back to HTTPS. The end result is that this HTTP URL will trigger a plain HTTP connection on the HTTPS endpoint address.

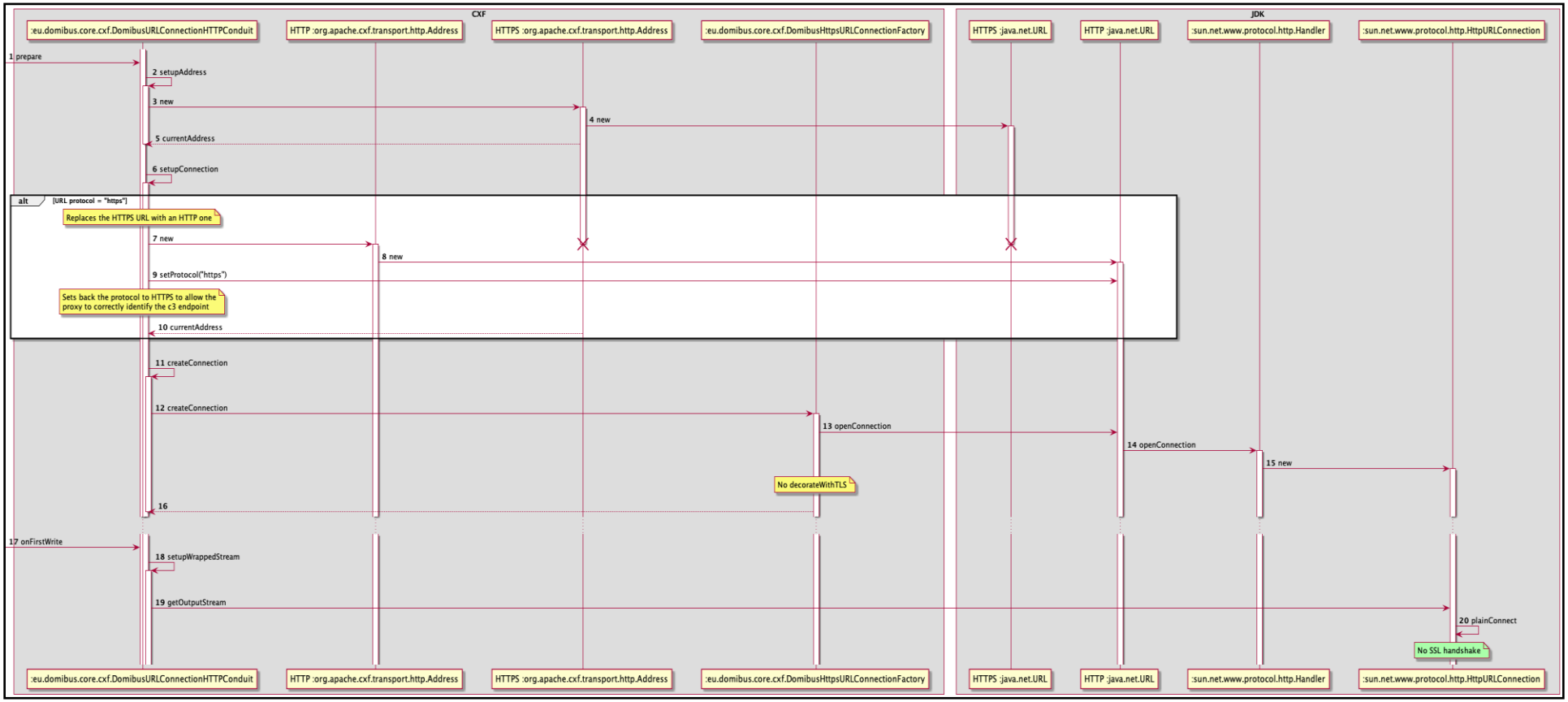


Figure 10 – Endpoint address

28. OPERATIONAL GUIDELINES

In this section you will find some recommendations about how to administer Domibus efficiently. The following topics are tackled: JMS Queue management, log management, capacity planning, database management and the monitoring of message life cycle.

28.1. JMS Queue Management

Domibus provides following out-of-the-box features to manage the JMS Queues used in Domibus (see also §10.7- *"Queue Monitoring"*):

- Inspecting and filtering the messages from a queue based on the contents of Source, Period, JMS Type or Selector
- Move message from the DLQ (Dead Letter Queue) to the original Queue
- Delete stuck or pending message(s) from Queues

It is recommended to monitor the Queue size and number of messages in the different Queues. If some messages are stuck in any of the Queue then alerts must be sent to the Domibus Administrator.

Please pay special attention to the dead letter queue (DLQ). Messages stuck in this queue is a signal that there is some issue in Domibus that needs to be analysed and an alert should be sent to the Domibus Administrator.

Important:

The 'ListPendingMessages' operation on WS Plugin browses the JMS queue. Max count is limited to destination MaxBrowsePageSize which can be changed via the 'domibus.listPendingMessages.maxCount' Domibus property.

If the received messages are not returned by the webservice listPendingMessages method, you should:

1. increase the value of the 'domibus.listPendingMessages.maxCount'property;
2. delete the messages from the domibus.notification.webservice queue with selector NOTIFICATION_TYPE=MESSAGE_SEND_SUCCESS using JMX tools: http://activemq.apache.org/how-can-i-monitor-activemq.html .

28.2. Log Management

28.2.1. Log Level

It is recommended that the log level is correctly set in all the environments:

- The log level should be set to INFO/DEBUG in all the test environments for debugging purpose.
- The log level should be set to ERROR/WARN in production environment (keeping log level to INFO in production environment will degrade the performance of Domibus).

28.2.2. Log Rotation and Archiving

It is recommended that log rotation and archiving logic is implemented.

Domibus provides by default log rotation, but Domibus administrator should manage Domibus archiving logic.

28.2.3. Log Monitoring

It is recommended to monitor continuously Domibus logs. It can be done using an automated script which looks for keywords like "ERROR", WARNING", etc. and reports all the errors and warnings to the Domibus administrator.

28.3. Capacity Planning

28.3.1. JVM Memory Management

Hereafter some recommendations:

- the JVM memory parameters must first be tested in a test environment with the load expected in production
- the JVM parameters i.e. heap size must be monitored with the help of automated scripts and any abnormal hikes in heap size must be reported to the administrator.

28.3.2. CPU, IO operations and Disk Space Monitoring

CPU, IO operations and disk space must be continuously monitored using automated scripts. Any abnormal hikes must be reported to Domibus administrator and further investigated.

28.4. Database Management

28.4.1. Database Monitoring

It is important to monitor the database size.

The Payload of the message is deleted from the sending Access Point. Only the metadata of the message stays in the table. The Payload from the receiving Access Point is deleted based on the retention policy defined in the PMode settings.

Domibus uses approximately 40 MB of table space to store the metadata of 1000 messages.

28.4.2. Database Archiving

Since the Database contains AS4 receipts that are used for non-repudiation purposes, they should be archived before purging the database.

The metadata of the database can be purged if it is no longer required.

28.4.3. Monitor Message Life Cycle

It is recommended to monitor the message status in the TB_MessageLog table. Automated scripts can be used to count different status in the table.

Please pay special attention to the following statuses:

- WAITING_FOR_RETRY: this means that there is some issue between C2 and C3 that must be resolved.
- SEND_FAILURE: this means that that there is some issue between C2 and C3 that must be resolved.
- SEND_ENQUEUED: this is part of the successful message life cycle, however abnormal increase in the count of messages with this status means that there is an issue. Further investigation is recommended.

28.5. Domibus Monitoring/Domibus IsAlive AP

28.5.1. Database Monitoring

Monitoring or IsAlive external service, checks the status of Domibus database, JMS Broker and Quartz Triggers using REST API.

28.5.2. Check Domibus DB, JMS Broker and Quartz Trigger isAlive

This REST endpoint will get the monitoring details of Domibus by checking its DB, JMS Broker and Quarter Trigger with or without the filters.

HTTP method: GET

```
http://localhost:8080/domibus/ext/monitoring/application/status
```

Response: HTTP 200 status with body:

```
{
    "services": [
        {
            "name": "Database",
            "status": "NORMAL"
        },
        {
            "name": "JMSBroker",
            "status": "NORMAL"
        },
        {
            "name": "Quartz Trigger",
            "status": "NORMAL",
            "quartzTriggerInfos": []
        }
    ]
}
```

28.5.3. DataBase Monitoring:

Domibus checks the database connection by fetching the user details from the TB_USER table. If Domibus user details are successfully fetched from the database without any exception, API returns the DB status as NORMAL, otherwise the API returns the status as ERROR.

HTTP method: GET NORMAL’

```
http://localhost:8080/domibus/ext/monitoring/application/status?filter=db
```

Response: HTTP 200 status with body:

```
{
    "services": [
        {
            "name": "Database",
            "status": "NORMAL"
        }
    ]
}
```

28.5.4. JMS Monitoring:

Domibus tries to get the number of JMS messages in the first queue that contains pull in the JMS Broker. If there is no exception, API returns the JMS Broker status as NORMAL, otherwise the API returns the status as ERROR.

HTTP method: GET

```
http://localhost:8080/domibus/ext/monitoring/application/status?filter=jmsBroker
```

Response: HTTP 200 status with body:

```
{
    "services": [
        {
            "name": "JMSBroker",
            "status": "NORMAL"
        }
    ]
}
```

28.5.5. Quartz Trigger Monitoring:

Domibus tries to get the Quartz triggers in BLOCKED or ERROR status. If there are no triggers in ERROR status or BLOCKED for longer than 10 minutes, the API returns the Quartz trigger status as NORMAL, otherwise the API returns the status as ERROR with the list of Quartz jobs in ERROR status.

HTTP method: GET

```
http://localhost:8080/domibus/ext/monitoring/application/status?filter=quartzTrigge
r
```

Response: HTTP 200 status with body:

```
{
    "services": [
        {
            "name": "Quartz Trigger",
            "status": "NORMAL",
            "quartzTriggerInfos": []
        }
    ]
}
```

Response: HTTP 200 status with Triggers with blocked and error state:

```
{
    "services": [
        {
            "name": "Quartz Trigger",
            "status": "ERROR",
            "quartzTriggerInfos": [
                {
                    "jobName": "alertCleanerJob",
                    "domainName": "Default",
                    "triggerStatus": "BLOCKED"
                },
                {
                    "jobName": "errorLogCleanerJob",
                    "domainName": "Default",
                    "triggerStatus": "ERROR"
                }
            ]
        }
    ]
}
```

More API details of these external services are present in Domibus REST Service Open API documentation which is part of the Domibus distribution artefacts (see [REF1]).

29. Annex 1 - Usage of certificates in PEPPOL and OASIS

		C2		C3	
		Keystore	**Truststore**	**Keystore**	**Truststore**
PEPPOL	Certificate:	Sender's (issued by CA)	Empty	Receiver's	CA's
	Note:	C2 signs the message with its private key	C2 discover C3 public certificate from the SMP	C3 signs the receipt with its private key	The receiver trusts all senders whose certificates were issued by these CA's
OASIS	Certificate:	Sender's (issued by CA)	SMP's	Receiver's	CA's
	Note:	C2 signs the message with its private key	C2 discover C3 public certificate from the SMP To trust the SMP, the sender needs its public certificate	C3 signs the receipt with its private key	The receiver trusts all senders whose certificates were issued by these CA's

30. List of Figures

Figure 1 - Diagram representing the Deployment of Domibus in a Cluster on WebLogic 28
Figure 2 - Diagram representing the Deployment of Domibus in a Cluster on Tomcat 45
Figure 3 - Diagram representing the Deployment of Domibus in a Cluster on WildFly 61
Figure 4 - Message Service Handler diagram 68
Figure 5 - State machine of Corner 2 (sending access point) 143
Figure 6 - State machine of Corner 3 (receiving access point) 143
Figure 7 - Domibus – Error Log page 147
Figure 8 – PMode page 148
Figure 9 – SSL handshake 277
Figure 10 – Endpoint address 278

List of Tables

Table 1 - Domibus Properties 111
Table 2 – Super-domibus Properties 112
Table 3 - Domibus PMode configuration to ebMS3 mapping 126
Table 4 - Queue Monitoring 151
Table 5 - SMP Entry Mapping 209

31. CONTACT INFORMATION

eDelivery Support Team

By email: EC-EDELIVERY-SUPPORT@ec.europa.eu

SUPPORT Service: 8am to 6pm (Normal EC working Days)