

domibusConnector-4.0-RELEASE - InstallationGuide

Table of contents

TABLE OF CONTENTS	2
1. INTRODUCTION	4
1.1. SCOPE AND OBJECTIVE OF THIS DOCUMENT	4
1.2. THE DOMIBUSCONNECTOR AS A WEB APPLICATION	4
1.3. THE DOMIBUSCONNECTORCLIENT	4
1.4. THE GATEWAY	4
1.5. THE DOMIBUS-CONNECTOR-PLUGIN	5
2. PRECONDITIONS AND TECHNICAL REQUIREMENTS.....	6
2.1. THE DOMIBUSCONNECTOR DISTRIBUTION PACKAGE	7
2.2. SUPPORTED OPERATING SYSTEMS	6
2.3. JAVA RUNTIME	6
2.4. DATABASE	6
2.5. WEB CONTAINER	6
2.6. INTERNET CONNECTION.....	7
2.7. TECHNICAL SPECIFICATIONS	7
3. DATABASE INSTALLATION.....	9
3.1. SUPPORTED DATABASE VENDORS	9
3.2. NEW DATABASE / FRESH INSTALLATION.....	9
3.2.1. USING THE SCRIPTS.....	9
3.2.2. USING LIQUIBASE	9
3.3. DATABASE UPGRADE 3.5 TO 4.0	9
3.3.1. USING THE SCRIPT	10
3.3.2. USING LIQUIBASE	10
3.4. UPGRADE WITH LIQUIBASE	10
4. CONFIGURATION PROPERTIES	15
5. CERTIFICATE, KEY-STORES AND TRUSTSTORES.....	12
5.1. CONNECTOR BACKEND KEY STORE	13
5.2. CONNECTOR KEY STORE.....	13
5.3. EVIDENCE KEY STORE.....	13
5.4. CONNECTOR TRUSTSTORE.....	14
5.5. TLS KEY STORE (SYSTEM KEY STORE)	FEHLER! TEXTMARKE NICHT DEFINIERT.
6. DEPLOYMENT.....	16
7. IMPORT OF P-MODES	18
7.1. IMPORT OF A P-MODE FILE.....	18
7.2. DATATABLES	18
8. BACKEND CONFIGURATION	20

8.1.	BACKEND TYPES.....	20
8.1.1.	PUSH/PULL BACKEND.....	20
8.1.2.	PUSH/PUSH BACKEND.....	20
8.2.	ADDING THE BACKEND CLIENT KEYS TO THE CONNECTOR BACKEND KEY STORE.....	20
8.3.	CONFIGURING THE BACKEND AT THE DATABASE.....	21
8.3.1.	DOMIBUS_CONNECTOR_BACKEND_INFO	21
8.3.2.	DOMIBUS_CONNECTOR_BACK_2_S.....	21
8.3.3.	EXAMPLE SCRIPTS.....	22

1. Introduction

1.1. Scope and Objective of this document

This document is a technical guide to install and configure the domibusConnector 4.0-RELEASE. It can be used as a “go-through” installation guide. Readers should be able to install and configure the domibusConnector in their own environments without previously built know-how about the software.

The target audience of this document are technical personal or administrators that have experience in network environments and widely known software components like web servers or application servers.

A detailed knowledge of the own network structures and environment is a precondition.

The structure of this guide is built so that every step can be taken as listed in the document. That means all preconditions for a chapter should be given by the previous chapters.

As an InstallationGuide this document does not focus on features and functionalities on the usage of the domibusConnector. For more details on the usage please read the “domibusConnector_Technical-documentation-and-UserGuide” distributed together with the domibusConnector-4.0-RELEASE.

1.2. The domibusConnector as a web application

Starting with version 4.0-RELEASE, the domibusConnector is on the technical basis of a web application.

This means, that the domibusConnector itself is a “ready-to-use” software component that only needs to be configured, set-up and deployed in a web container.

Once installed and configured properly, the domibusConnector should run on its own.

1.3. The domibusConnectorClient

The domibusConnector web application offers different interfaces that can be used to approach the functionalities. Those interfaces can be used directly, if intended.

To close the missing link between your own implementation and the provided web service of the domibusConnector, a domibusConnectorClient was implemented to support the connection to the domibusConnector.

The domibusConnectorClients and all its variants and usage are described in the document “domibusConnectorClient_Guide”.

1.4. The gateway

To establish a successful connection to e-CODEX network partners, it is necessary to have a gateway component for transmission of messages.

To be able to connect with e-CODEX partners the ebms3 standard of OASIS must be respected by the gateway. Additionally, there are different profiles on how the structure of ebms3 messages can be transmitted. In e-CODEX all partners agreed on using the e-SENS-AS4 pattern.

During the e-CODEX project an own gateway component was implemented as a building block that fulfils all mentioned requirements. This is the DOMIBUS Gateway.

Today the development and maintenance of the DOMIBUS Gateway lies at the CEF program of the European Commission.

Further details on the DOMIBUS gateway can be found at the CEF homepage:

<https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Domibus>

1.5. The domibus-connector-plugin

To have a connection between the domibusConnector and the Domibus Gateway, a plugin has been developed that can easily be installed on the gateway.

This plugin implements all interfaces that enable message transmission between the connector and the gateway. It is also available as a distribution package of e-CODEX on the Nexus repository server:

<https://secure.e-codex.eu/nexus/content/groups/public/eu/domibus/connector/domibus-connector-plugin/>

Details on how to install the plugin on the Domibus Gateway can be found in the documentation of the respective Domibus Gateway.

2. Preconditions and technical requirements

This chapter describes what has to be in place prior to install the domibusConnector. It also lists some technical specifications of the domibusConnector to give a more detailed insight.

2.1. Supported operating systems

The domibusConnector is a software that has been completely implemented using the JAVA programming language.

As JAVA is by definition a platform independent environment, every operating system with a proper JAVA installation should fit the needs of setting up the domibusConnector.

During implementation and testing phase of the domibusConnector, it was installed and tested on the following environments:

- Microsoft Windows 7
- Linux
- IBM AIX

2.2. Java Runtime

As the domibusConnector is a JAVA application, it also requires a proper installation of a Java Runtime to be able to run the software.

The recent version 4.0-RELEASE of the domibusConnector has been implemented and compiled with an Oracle JDK jdk-8u161. So at least this version or higher should be in place to avoid incompatibilities.

2.3. Database

The domibusConnector needs an underlying database to store information. Currently the following DBMS are supported:

- MySQL 5.5 and higher
- Oracle 12g and higher

The domibusConnector distribution package offers SQL scripts that are meant to either set up a completely new database for the connector, or to migrate an existing domibusConnector database to its current version.

Be aware that most web-servers need to have proper JDBC drivers installed to be able to set the connection to the database.

Details on the installation of the database can be found in the chapter [Database Installation](#).

2.4. Web container

Since the domibusConnector in its current version 4.0-RELEASE is a web application built as a WAR deployable, it needs a web-server or application server that it can be deployed at.

This can be any JAVA compliant product which supports at least Servlet 3.0 API level.

During implementation and testing of the domibusConnector the following web-servers were used:

- Apache Tomcat 8
- Oracle Bea WebLogic 12c

Be aware that those are neither requirements nor recommendations, but only listed for information.

This installation guide does not focus on specifics of web-server technologies in detail.

Details on how to deploy the domibusConnector on that web server products can be found in chapter [Deployment](#).

2.5. Internet connection

As the domibusConnector needs some sources from the internet for the security library features, also an internet connection from the installation point must be given. To be able to configure your environment the domibusConnector gives the opportunity to configure proxy settings in the connector properties described in chapter [Configuration properties](#).

2.6. Technical specifications

The main frameworks and technologies the domibusConnector was implemented with is listed here for your information:

- Java 8 (Oracle jdk-8u161)
- Spring framework 4.3.12.RELEASE
- Spring-boot 1.5.8.RELEASE
- Hibernate 5.0.12.FINAL
- Apache CXF 3.2.1
- Apache Maven 3

2.7. The domibusConnector distribution package

To get started, you first need to have downloaded and extracted the distribution package.

The domibusConnector provides different distribution packages all placed on the e-CODEX Nexus repository server at:

<https://secure.e-codex.eu/nexus/content/groups/public/eu/domibus/connector/domibusConnectorDistribution/4.0-RELEASE/>

- domibusConnectorDistribution-4.0-RELEASE.war
- domibusConnectorDistribution-4.0-RELEASE.zip

are the most important ones.

Whereas the “domibusConnectorDistribution-4.0-RELEASE.war” is only the deployable WAR package and intended to be used to simply upgrade already set up domibusConnector installations, the “domibusConnectorDistribution-4.0-RELEASE.zip” is of interest for setting up a new domibusConnector.

Once downloaded and extracted it has the following structure:

File/directory	Description
Webapp (directory)	This directory contains the application itself distributed as “domibusConnector-4.0-RELEASE.war”
Documentation/database-scripts (directory)	This directory contains all necessary database scripts to set up the database for the

	<p>domibusConnector. The scripts are prepared for the database vendors MySQL and Oracle.</p> <p>For more details see chapter Database Installation</p>
Documentation/databaseInitializer (directory)	<p>Contains the “domibusConnectorDatabaseInitializer.jar” which is a helper application to set up the database.</p> <p>For more details see chapter Database Installation</p>
Documentation/properties (directory)	<p>The “properties” folder contains example properties that show how to configure the domibusConnector. The log4j configuration is also contained as an example.</p>
domibusConnector_Monitoring_Interfaces.pdf	<p>A document that describes what monitoring interfaces the domibusConnector offers and how to approach them.</p>
domibusConnector-Technical-documentation-and-UserGuide.pdf	<p>This document merges the documentation for the domibusConnector for administrators and users. This document covers all distributions of the domibusConnector.</p>

3. Database Installation

The “domibusConnectorDistribution-4.0-RELEASE.zip” deliverable package contains database scripts to either create a new database or upgrade an existing database for domibusConnector 3.5(.1).

As a precondition a DBMS already needs to be in place. We recommend to create an own schema/user for the domibusConnector database.

3.1. Supported Database vendors

Tests for the domibusConnector have been done using the following databases:

- Mysql from version 5.5 onwards
- Oracle from version 12g onwards

Prepared database scripts exist only for those vendors. Though, any other SQL database can be used.

3.2. New Database / Fresh Installation

Starting with a new installation and therefore have an empty schema/user on the database system created, one has just to execute the provided scripts or use liquibase.

3.2.1. Using the scripts

The documentation contains a folder database-scripts/initial. This folder contains the following DDL/SQL scripts:

- “MySql_4_0_initial.sql” for MySQL
- “Oracle_4_0_initial.sql” for Oracle

Once those scripts are executed on the dedicated schema, the database is ready for usage for the domibusConnector.

3.2.2. Using liquibase

It is also possible to let liquibase create your database tables. Start reading the section “Upgrade with Liquibase” down below.

3.3. Database Upgrade 3.5 to 4.0

Upgrading an existing domibusConnector database for prior releases 3.5 or 3.5.1 is possible. But first of all we strongly recommend to create a backup of the existing database schema.

Then you can choose to upgrade your database schema manually by executing the scripts or let liquibase do the work.

Both methods are assuming that there are no changes or additional constraints, indexes added compared to the 3.5 database script.

3.3.1. Using the script

- Create a backup of your current database schema
- Drop/deactivate all foreign-key constraints (the script will create them again!)
- Execute the upgrade script which is located in the folder database-scripts/migration.

Use the script for your database vendor:

- “MySQL_Migrate_3.5_ConnectorDB_to_4.0.sql” for MySQL
- “Oracle_Migrate_3.5_ConnectorDB_to_4_0.sql” for Oracle

3.3.2. Using liquibase

If you want to use liquibase for database upgrade please continue with the next section “Upgrade with Liquibase”.

3.4. Upgrade with Liquibase

It is also possible to let liquibase upgrade or create your database. Liquibase is a tool which splits the database creation/upgrade into multiple changesets. In the future it will allow semi automatic database upgrades. You can also use liquibase to use unsupported databases like postgresql.

Liquibase is packaged into the jar named domibusConnectorDatabaseInitializer.jar which includes all the necessary database scripts:

- Database Migrate 3.5 to 4.0 DB ChangeLog: “db/changelog/v004/upgrade-3to4.xml”
- Database Create 4.0 DB ChangeLog: “db/changelog/install/initial-4.0.xml”

You can execute the scripts in your database by executing the jar:

```
java -jar domibusConnectorDatabaseInitializer.jar --changeLogFile=${changeLogFile} \
--driver=${sqlDriverName} \
--url=${databaseUrl} \
--username=${databaseUsername} \
--password=${databasePassword} \
--classpath=${jdbcDriverJar} \
upgrade
```

You have to provide the following parameters:

- “--driver=” the jdbc driver name.
 - com.mysql.jdbc.Driver for MySQL
 - oracle.jdbc.OracleDriver for Oracle
- “--url=” the jdbc url to access the database (consult the documentation of your jdbc driver)
 - Example: “jdbc: mysql://localhost/domibusconnector” for connecting to a local MySQL database named domibusconnector.
- “--username=” the username to access the database. The database user needs the permission to make schema modifications.
- “--password=” the password of the database user.

- “--classpath=” the path to an additional jar which contains the jdbc driver (the package already contains the mysql jdbc driver, so this parameter is only needed to provide the oracle jdbc driver jar)
- “--changeLogFile=” the change log liquibase should run against the database
- “-help” will show the liquibase help

4. Certificate, Key-Stores and Truststores

To ensure the highest reasonable level of security, the domibusConnector uses several certificates for different purposes:

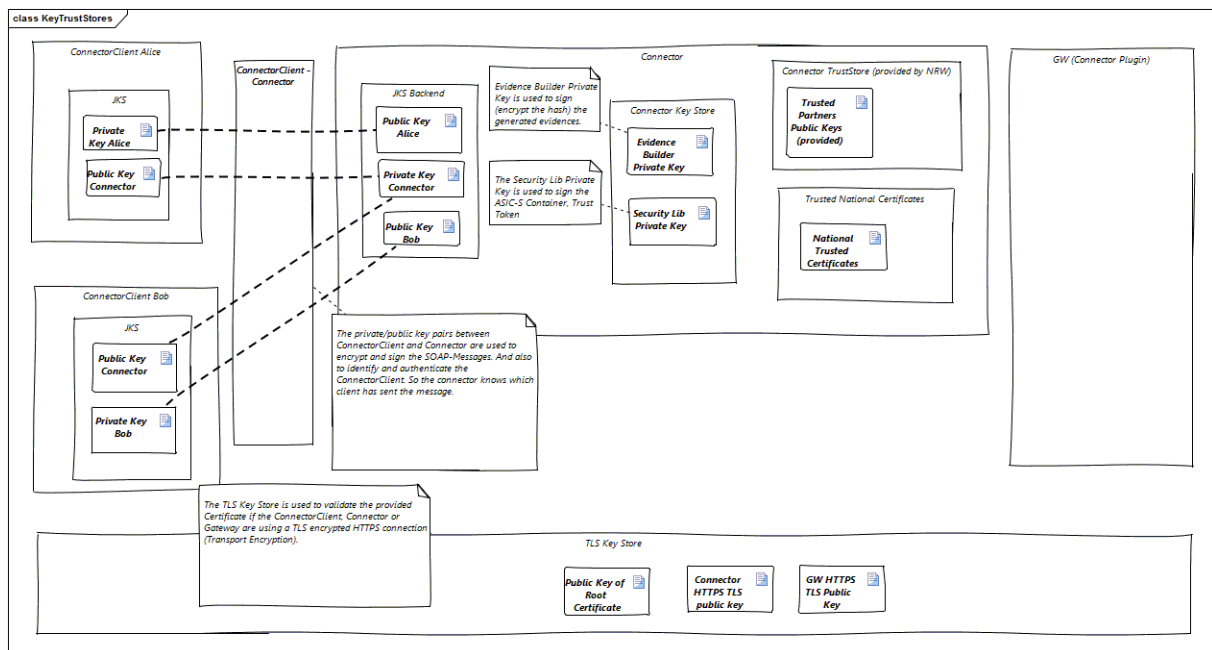
- Signing and Encrypting SOAP messages between the backend client and the Connector.
- Establishing Transport Security (TLS) between the backend client and the Connector.
- Signing and Encrypting SOAP messages between the Connector and the Gateway.
- Establishing Transport Security (TLS) between the Connector and the Gateway.
- Validating the signature of the main document (mostly a PDF) of the message (if configured).
- Validating the signature of the secure container (ASIC-S) received with incoming messages.
- Signing the secure container (ASIC-S) that is created by the Connector.
- Signing the ETSI-REM evidences.

In most of those cases the same certificate can be used, though we do not recommend that. For higher security it is more efficient to use different certificates.

In case of multiple backend clients that connect to the domibusConnector it is required to have an own certificate per backend.

Each certificate holds a private key, which should always stay inside your organization, and a public key.

The following graphic shows an example of what key is used at what point. It also shows the purpose of the different keys:



This guide focuses on the keys and stores that are required for a proper installation of the domibusConnector. Other keys and stores, for the backend client(s) for example, are explained in more detail in other documentations.

4.1. Connector Backend Key Store

The connector backend key store holds the private key of the connector which is used to decrypt and sign the messages which are sent to the connector clients. It also contains all public keys of all the backend clients to verify the signature of received messages. The common name (CN) of the certificate is also used to identify the backend clients name.

The configuration properties are:

```
#defines the location of the backend keyStore:
connector.backend.ws.key.store.path=
#defines the key store password:
connector.backend.ws.key.store.password=
#defines the key alias for the key which is used to sign the messages:
connector.backend.ws.key.key.alias=
#defines the key password:
connector.backend.ws.key.key.password=
```

4.2. Connector Key Store

The connector key store holds the private key for signing the ASIC-S container.

The configuration properties are:

```
#defines the key store location:
connector.security.keystore.path=
#defines the key store password
connector.security.keystore.password=
#defines the key alias
connector.security.key.alias=
#defines the key password
connector.security.key.password=
```

4.3. Evidence Key Store

The evidence key store holds the private key for signing the generated ETSI-REM evidences. This private key and key store can be the same for signing the ASIC-S container (Connector Key Store).

The configuration properties are:

```
#defines the key store location:
connector.evidences.keystore.path=
#defines the keystore password
connector.evidences.keystore.password=
#defines the key alias
connector.evidences.key.alias=
#defines the key password
connector.evidences.key.password=
```

4.4. Connector truststore

This truststore only holds public keys. The connector truststore (in configuration management called the “connectorstore”) is provided by the configuration management of the project and contains the public keys of the e-CODEX partners. They are used to verify the signature of the ASIC-S container received from an e-CODEX partner.

Additionally, if your organization uses signed documents (mostly PDF) as the main content of the message when sending a message to an e-CODEX partner, the public key of the certificate with which the document was signed with should be imported into this truststore. The security library uses this public key to verify the signature of the document then (configured as SIGNATURE_BASED).

The configuration properties are:

```
#defines the key store location:  
connector.security.ojstore.path=  
#defines the store password  
connector.security.ojstore.password=
```

5. Configuration properties

In order to give the domibusConnector the missing links about your environment, some properties have to be set in a property file. Usually this is called “connector.properties”.

There is also the possibility to adopt the logging configuration. This gives the opportunity to control where logs are written to and what to log.

Example properties and an empty property file, as well as an example for logging configuration can be found in the distribution package at “documentation/properties”.

The properties in those file are all well described on what is expected there.

The variants on how to include the properties into your web server environment is dependent on what product you have in place.

For the web server products Apache Tomcat and BEA Weblogic this is described exemplarily in the Chapter [Deployment](#).

6. Deployment

This chapter describes the steps to be taken to deploy the domibusConnector application on a web server. The e-CODEX community can provide more detailed information for the web server products Apache Tomcat and BEA Weblogic. Other Web Servers have not been subject to any tests.

It is not a requirement that one of the listed web server products must be used.

6.1. Deploy on Apache Tomcat

The suggested way to deploy the domibusConnector application on an Apache Tomcat server is to define a new context which

- configures a datasource as the connection to the domibusConnector database
- sets the parameters for the domibusConnector application
- loads the domibusConnector application.

The delivered distribution package contains an example application context, which has been tested on Microsoft Windows 7 with an Apache Tomcat 8.

6.1.1. Tested Tomcat Version

The deployment has been tested with the following versions:

- Apache Tomcat 8.5.23 on Windows 7

6.1.2. Deployment steps

- Copy the file "domibusConnector-4.0-RELEASE.war" into your Apache Tomcat installation "<path_to_tomcat>/conf/domibusConnector".
- Copy the adopted "connector.properties" and "log4j.properties" file into "<path_to_tomcat>/conf/domibusConnector".
- Copy the example context file "domibusConnectorWebAppModule.xml" from the distribution package "config/tomcat" into "<path_to_tomcat>/conf/Catalina/localhost".

The context file needs to be adapted, as the example file needs the missing parameters:

```
<Context docBase="<path_to_tomcat>/conf/domibusConnector-4.0-RELEASE.war">
  <Parameter name="spring.datasource.jndi-name"
    value="jdbc/domibusWebConnectorDS" override="false" />
  <Parameter name="connector.config.location"
    value="<path to the folder containing the application.properties"
    override="false" />
  <Resource name="jdbc/domibusWebConnectorDS" auth="Container"
    type="javax.sql.DataSource"
    driverClassName="<jdbcDriverClass>"
    url="<databaseUrl>"
    username="<username>"
    password="<password>"
    maxActive="20"
    maxIdle="10"
    maxWait="-1"/>
</Context>
```

Finally start/restart your Apache Tomcat. The application should be deployed automatically.

6.2. Deploy on BEA Weblogic

The connector application is a spring boot application which should run on a weblogic application

server.

In its default configuration it expects a jndi DataSource configured with the name “domibusWebConnectorDS”. Please configure a datasource with this name and deploy the application to your weblogic server. You should also set the “connector.config.location” parameter to your “connector.properties” so the spring boot application can load the configured settings. For that purpose you need to create a custom deployment descriptor.

7. Import of p-modes

The “p-modes” that are distributed by the configuration management for the domibus gateway can also be used for the domibusConnector database. To have necessary data to support business use cases in your domibusConnector database.

If the set up domibusConnector is not a fresh new one but a migrated one from a previous version, this step should not be necessary.

Once the domibusConnector is successfully deployed in a web container and running, the pages of the domibusConnector can be reached.

The default login already stored in the database for the web user interface is “admin” with the password “admin”.

7.1. Import of a p-mode file

The requested functionality is reachable by clicking on “DataTables” in the left menu. At the “DataTables” view next to “Select PMode File:” a button to choose the p-mode is placed. Once the p-mode file is selected, it can be uploaded and processed by hitting “Import”.

Every ACTION, PARTY and SERVICE that is found in the p-modes and which do not exist in the database already, will be created. The domibusWebAdmin will neither change any existing database items (service, action or party), nor will it delete any of those. If the p-mode-file cannot be processed an error message will appear.



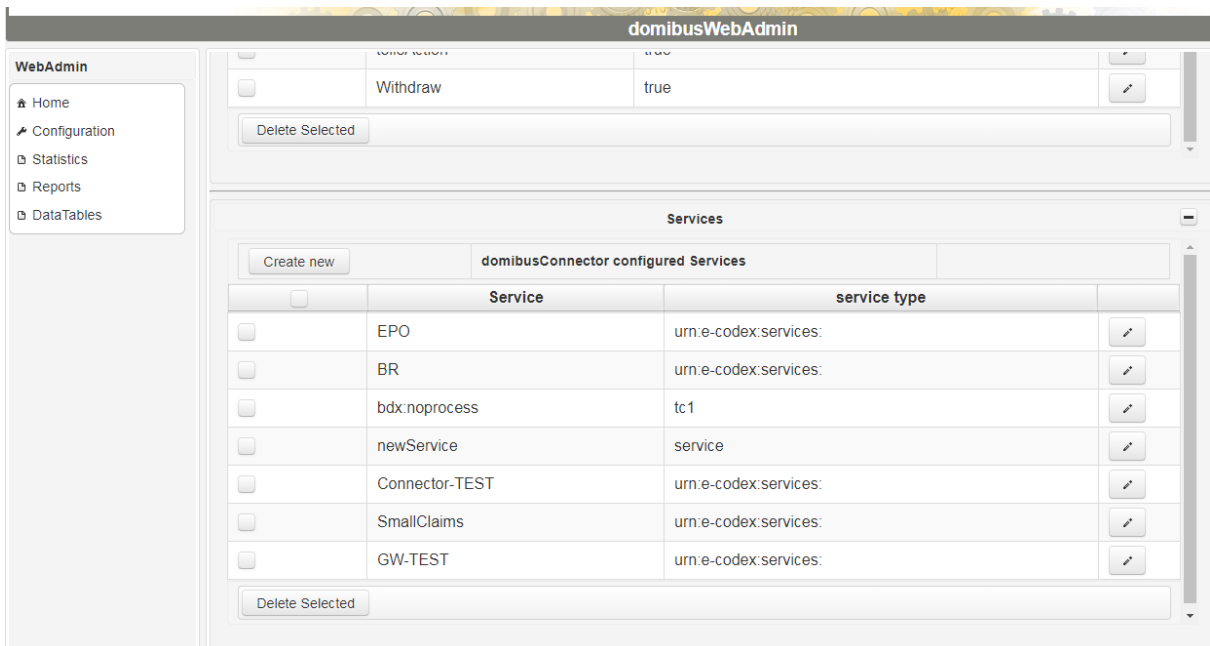
7.2. DataTables

Once the p-modes are imported, there is also the capability to modify, create and delete items (parties, actions and services) that are persisted in the database of the domibusConnector. This new feature is provided by the “DataTables” view. It is not possible to delete items that are referenced by a stored message. This means if a message has already been sent to or received from a party or has used a service or action, that item is not allowed to be deleted.

The following list gives an overview of what is editable:

- Party
 - Party id (not editable)
 - Party id type (editable)
 - Role (not editable)

- Action
 - Action (not editable)
 - Pdf required (editable)
- Service
 - Service (not editable)
 - Service type (editable)



The screenshot shows the 'domibusWebAdmin' interface. On the left is a sidebar with navigation links: Home, Configuration, Statistics, Reports, and DataTables. The main content area is titled 'domibusWebAdmin' and contains a 'Services' section. At the top of this section is a 'Create new' button and a label 'domibusConnector configured Services'. Below this is a table with columns: 'Service' and 'service type'. The table lists several services: EPO, BR, bdx:noprocess, newService, Connector-TEST, SmallClaims, and GW-TEST. Each row has a checkbox in the first column and a pencil icon in the last column. Below the table is a 'Delete Selected' button.

	Service	service type	
<input type="checkbox"/>	EPO	urn:e-codex:services:	
<input type="checkbox"/>	BR	urn:e-codex:services:	
<input type="checkbox"/>	bdx:noprocess	tc1	
<input type="checkbox"/>	newService	service	
<input type="checkbox"/>	Connector-TEST	urn:e-codex:services:	
<input type="checkbox"/>	SmallClaims	urn:e-codex:services:	
<input type="checkbox"/>	GW-TEST	urn:e-codex:services:	

Pressing “Create new” starts the process of creating a new service. A dialog appears where the required information can be filled in.

By ticking the checkboxes in the left column and hitting “Delete Selected” a confirmation dialog appears. The deletion process is started after confirmation. If there are any references to processed messages deletion fails and an error message appears.

Editing is started by pressing the button with the pencil symbol on it. A dialog box opens in which the item can be changed. Editing can be stopped by pressing “Cancel”. Changes are committed by pressing “Save”.

8. Backend configuration

The domibusConnector can be accessed by multiple clients. It does not matter, if the clients use the distributed domibusConnectorClient (either one of the libraries for integration, or the standalone client), as long as the backend interfaces of the domibusConnector are implemented properly and the security needs can be met.

This guide focuses on the configuration needed to add new backend clients to connect to the domibusConnector. Details on configuration steps needed on the client side can be found in the domibusConnectorClient documentation.

This chapter follows an example in which 2 new backend clients should be configured in the domibusConnector:

- Alice
- Bob

8.1. Backend types

Whereas it is given in version 4.0-RELEASE of the domibusConnector that the backend interfaces only can be reached via SOAP webservises, it can be configured if the backend client supports to be called as an active service.

8.1.1. Push/pull backend

This type of backend client does not support an active webservice itself. It can only be seen as a passive client. The domibusConnector can receive messages from this type of client at any time, but cannot actively send messages to the client.

That means that the client itself has to call the webservice of the domibusConnector to receive messages that are stored inside the connector until the client calls them.

Mostly this is done by having time triggered jobs running on the client side that connector to the domibusConnector to receive its messages.

The domibusConnectorClient Standalone variant is of that kind.

8.1.2. Push/push backend

This type of backend client does support an active webservice. This webservice must run in a web container itself and must implement the delivery web service of the domibusConnector. In that case the domibusConnector does not need to wait until the clients connects, but can push messages to the client by itself.

8.2. Adding the backend client keys to the Connector Backend Key Store

Every new backend client needs its own certificate.

The certificate of a backend client has the following purpose:

- Identifies the backend client when it connects to the domibusConnector
- Encrypts/Decrypts messages between the backend client and the Connector.

So let's assume, following the example of "alice" and "bob", that there are 2 certificates. What we need to configure the backend clients properly, are the public keys of those certificates.

Those 2 public keys need to be added to the "Connector Key Store" described and configured in chapter 5.1. To keep it transparent the public keys are imported into the store with the alias names "alice" and "bob".

8.3. Configuring the backend at the database

For the domibusConnector-4.0-RELEASE the adding of the backend must be done manually by accessing the domibusConnector database. The backend information must be stored in two database tables:

8.3.1. DOMIBUS_CONNECTOR_BACKEND_INFO

Name	Description
ID	a unique technical id
BACKEND_NAME	The name of the backend this name must match the common name (CN) field of the assigned certificate
BACKEND_KEY_ALIAS	The key alias in the connector backend keystore for the certificate to use to encrypt messages for the connectorClient
BACKEND_KEY_PASS	If the key is encrypted this column contains the password
BACKEND_SERVICE_TYPE	Not used yet, will later define the type of the backend, is it push/pull, push/push over webservices, push/push over jms
BACKEND_ENABLED	Is the backend enabled, must be true if the connector should send messages to this backend
BACKEND_DEFAULT	The default backend will receive all messages which aren't delivered to another backend first
BACKEND_DESCRIPTION	A description of the backend, can be used by the admin to store information
BACKEND_PUSH_ADDRESS	If the backend is a push backend, push address must be defined here

8.3.2. DOMIBUS_CONNECTOR_BACK_2_S

Contains the routing information, which backend will receive the message. The routing decision is based on the name of the business use case, which is called SERVICE in e-CODEX.

The table contains the following fields:

DOMIBUS_CONNECTOR_SERVICE_ID	References the service
DOMIBUS_CONNECTOR_BACKEND_ID	References the backend

8.3.3. Example scripts

Now we will have a look back to our example, where we want to add “alice” and “bob” as our new backend clients.

The following SQL statement will add an connectorClient named bob with the key alias bob and expects that the common name of the certificate is bob. Bob will also be the default backend!

```
INSERT INTO domibus_connector_backend_info  
(ID, BACKEND_NAME, BACKEND_KEY_ALIAS, BACKEND_ENABLED, BACKEND_DEFAULT)  
VALUES ('11', 'bob', 'bob', TRUE, TRUE);
```

The following statement will add “alice” to the backend configuration:

```
INSERT INTO domibus_connector_backend_info  
(ID, BACKEND_NAME, BACKEND_KEY_ALIAS, BACKEND_ENABLED, BACKEND_DEFAULT)  
VALUES ('12', 'alice', 'alice', TRUE, FALSE);
```

This statement will assign the epo messages to the connectorClient with the id 12 in the database. In this case this will be the connectorClient alice.

```
INSERT INTO domibus_connector_back_2_s  
(DOMIBUS_CONNECTOR_SERVICE_ID, DOMIBUS_CONNECTOR_BACKEND_ID)  
VALUES ('EPO', '12');
```

