

domibusConnector - Technical documentation and User Guide

Version :	V1.0
Relates to domibusConnector Release :	domibusConnector-3.5-RELEASE
Date of release :	May 31 st 2017
Author(s):	Robert Behr (AT), Bernhard Rieder (AT), Ferdinand Rödlich (AT)

History

<i>Version</i>	<i>Date</i>	<i>Changes made</i>	<i>Modified by</i>
0.1	30/05/2017	First draft for internal review	Bernhard Rieder
1.0	31/05/2017	First official version	

Table of contents

HISTORY.....	2
TABLE OF CONTENTS	3
LIST OF ABBREVIATIONS	FEHLER! TEXTMARKE NICHT DEFINIERT.
1. INTRODUCTION	5
1.1. SCOPE AND OBJECTIVE OF THIS DOCUMENT	5
1.2. THE DOMIBUSCONNECTOR AS AN E-DELIVERY BUILDING BLOCK	5
1.3. AVAILABLE DISTRIBUTION PACKAGES	5
2. INSTALLATION	7
2.1. SUPPORTED OPERATING SYSTEMS	7
2.2. JAVA RUNTIME	7
2.3. PROPERTIES	7
2.4. DATABASE	7
2.4.1. PRECONDITIONS	8
2.4.2. SCRIPTS	8
2.5. GATEWAY	8
2.5.1. DOMIBUS E-DELIVERY GATEWAY	8
2.5.2. DOMIBUS-CONNECTOR-PLUGIN.....	8
2.5.3. START PARAMETER FOR GATEWAY SERVICE	9
3. DOMIBUSCONNECTOR-FRAMEWORK	10
3.1. PLACEMENT OF THE DOMIBUSCONNECTOR FRAMEWORK.....	10
3.2. INSTALLATION	10
3.2.1. INTEGRATION USING MAVEN	11
3.2.2. MANUAL INTEGRATION.....	11
3.3. INTERFACES.....	11
3.3.1. DOMIBUSCONNECTORNATIONALBACKENDCLIENT.....	12
3.3.2. DOMIBUSCONNECTORCONTENTMAPPER.....	13
3.3.3. DOMIBUSCONNECTORSECURITYTOOLKIT	13
4. DOMIBUSCONNECTOR-STANDALONE	15
4.1. INSTALLING AND STARTING THE STANDALONE DOMIBUSCONNECTOR.....	15
4.1.1. AVAILABLE STARTUP SCRIPTS.....	15
4.2. SENDING A MESSAGE WITH THE STANDALONE CONNECTOR.....	15
4.3. SENDING A MESSAGE WITH DETACHED SIGNATURE	17
4.4. RECEIVING A MESSAGE WITH THE STANDALONE CONNECTOR	18
4.5. STRUCTURE OF THE MESSAGE.PROPERTIES.....	19
4.6. THE DOMIBUSCONNECTORCONFIGURATOR	20



4.7.	THE DOMIBUSCONNECTORGUI	23
------	-------------------------------	----

1. Introduction

1.1. Scope and Objective of this document

This document is a technical documentation of the domibusConnector. In some areas it is technically detailed and scoped for technicians, administrators or developers.

Other areas are meant to describe users of the domibusConnector (mainly the standalone distribution) how to handle the domibusConnector.

It is intended to structure the chapters for different readers. Each chapter contains, in its introduction, a description on who the chapter is mainly scoped to.

1.2. The domibusConnector as an e-Delivery building block

e-CODEX was a large scale project in the domain of e-Justice that aimed to provide to citizens, enterprises and legal professionals an easier access to justice in cross border procedures and to make cross border collaboration of courts and authorities easier and more efficient by creating interoperability of the existing national ICT solutions.

To grant these functionalities, a system was built based on so called building blocks. Those building blocks should connect any existing national environment to another in any other participating member state. This is called e-Delivery.

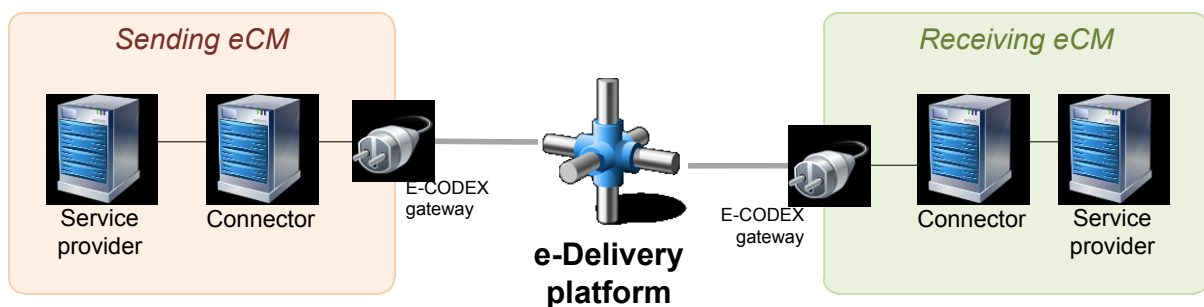


Figure 1: e-Delivery platform

Between the domibus gateway, which is a content agnostic point to point gateway solution following the AS4 standard, and the national service provider a missing link was discovered which helps the participants to produce, keep track, secure and map the messages that are exchanged over the gateways. This is covered by the domibusConnector.

1.3. Available distribution packages

To face the needs of existing national environments of participants, even if there is none existing yet, the domibusConnector is available in different distribution packages:

- DomibusConnectorDistribution-3.5-RELEASE-Framework:

The domibusConnector framework is an embeddable framework which must be integrated into an already existing (or new) national application that builds the shell around the framework and extends its functionalities with specific routines. It connects to/from other national backend applications and interacts with the domibus gateway to send and receive messages.

■ DomibusConnectorDistribution-3.5-RELEASE-standalone:

The Standalone connector's purpose is to serve participants that do not have existing national environments to interact with, or do not want to connect them. It interacts with some directories in the file system and works out of the box. For a better support it is shipped with a graphical user interface (GUI) that helps resolving received messages and also can trigger messages to be sent. Since there is no environment intended behind, it only expects and handles messages that are already structured properly for transmission.

2. Installation

This chapter describes common installation pre-conditions and guides through the setup of the database.

2.1. Supported Operating Systems

The domibusConnector was tested and prepared for the following operating systems:

- MS Windows: Version 7 onwards.
- Linux/Unix: Since the derivatives of Linux and Unix are wide spread, the support of those can only be guaranteed by providing startup-scripts. The domibusConnector was deeply tested with AIX, but every Unix System supporting Java Runtime should fit the need.

2.2. Java Runtime

Since the domibusConnector is a Java application, a Java Runtime needs to be installed on the target system. The Java Runtime is not distributed with the domibusConnector. Details on installation of Java can be found at the vendor's site. In its current release, the domibusConnector was compiled with Java 8.

Tested Runtimes:

- Oracle JDK 1.8.0
- IBM SDK 1.8.0

2.3. Properties

To give the domibusConnector environment specifics, you need to configure the properties first. An example of what those properties are can be found in the files "exampleConnectorProperties.properties" and "emptyConnectorProperties.properties". They can be edited with any text editor and need to be UTF-8 encoded.

With the standalone distribution of the domibusConnector there is also the possibility of editing the properties via [GUI](#).

Inside the GUI or the "exampleConnectorProperties.properties" every property that needs to be configured is also described.

How to properly configure the usage of those properties depends on the distribution package and is described in the "Installation" subchapters of the corresponding distribution.

2.4. Database

For every distribution of the domibusConnector an underlying database is needed to store information. Though every SQL conformant DBMS with a JDBC driver and Hibernate Dialect provided should work, the domibusConnector was only tested and set up with Oracle and MySQL.

2.4.1. Preconditions

Before being able to set up the database for the domibusConnector, the database instance itself needs to be in place and a schema or user with sufficient privileges must exist. Also, the connection from the domibusConnector to the DBMS must be given.

Oracle is supported from version 10g onwards and MySQL from 5.5 onwards.

2.4.2. Scripts

Within this documentation package the scripts to set up the database can be found at “database-scripts/initial/”. Please choose the “*.sql” file of the DBMS you have installed and run them on the database.

This script contains the setup of the data model needed by the domibusConnector as well, as the tables needed by Quartz, which is used for the time-triggered jobs inside the domibusConnector.

There are also scripts to completely drop the tables for the domibusConnector at “database-scripts/drop/”

After setting up the data model, you need to run the script “database-scripts/data/insertData.sql”. This script sets some important values for configuration tables needed to be able to send and receive messages with the domibusConnector. By the time of this release only data scripts to run with e-Codex Use Cases are provided.

2.5. Gateway

Since the domibusConnector was developed for the e-Codex project, where also the domibus gateway was developed, it is only tested with domibus gateways. Anyway, there are other commercial vendors for AS4 compliant gateways. The domibusConnector should also be compatible to some of those gateways, but was not tested towards any other gateway than the domibus.

2.5.1. Domibus e-Delivery gateway

The domibus gateway is now under maintenance of the CEF programme of the European Commission.

Detailed information and guides as well as the distributions can be found at the CEF site following this link:

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

The domibusConnector in its current version is only tested with the domibus release 3.2.4.

2.5.2. Domibus-connector-plugin

To couple the domibusConnector with the domibus gateway, and to be able to react faster to future changes at the domibus gateway, the domibusConnector also provides a plugin that can be installed on the domibus gateway.

This plugin has an extra interface to the domibusConnector, following the structure of messages to be better integrate able.

To use the domibus-connector-plugin you need to at least have domibus 3.2.4 installed. The plugin will NOT work with any older domibus distributions!

The domibus-connector-plugin can be downloaded at

<https://secure.e-codex.eu/nexus/content/repositories/releases/eu/domibus/connector/>

Information and guidance on how to install the plugin on the domibus can be found at the CEF site.

2.5.3. Start parameter for gateway service

Since the domibusConnector support in its current release two ways to interact with the gateway, there needs to be a start environment parameter set.

This parameter needs to be set where the domibusConnector starts. This is in the startup scripts for the standalone distribution. For the framework distribution it depends whether the framework is embedded into a web application, or another standalone application. For web application the parameter needs to be set in the starting sequence of the webserver, for any other applications, when the application is called.

In any case this parameter needs to be set. If not set, the domibusConnector cannot start!

The startup scripts of the standalone distribution are already shipped with the default value "Webservice".

2.5.3.1. Webservice

In case an older distribution of domibus is used than 3.2.4 where the domibus-connector-plugin cannot be used, or if another gateway solution is in place, there is a webservice that already provides ebms3 conformant messages to the gateway. To enable this choice the parameter to be set is `gateway.routing.option=Webservice`

2.5.3.2. Plugin

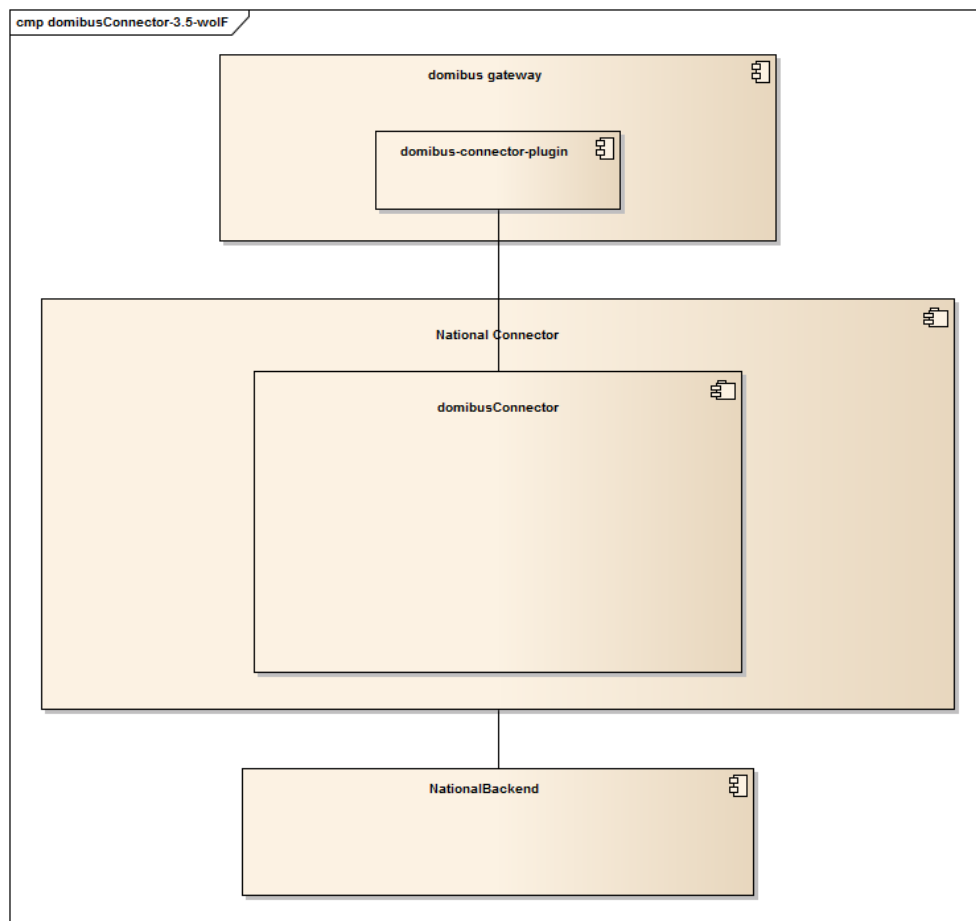
If the domibus gateway with the domibus-connector-plugin is in place, the parameter to be set is `gateway.routing.option=Plugin`

3. domibusConnector-Framework

This chapter describes the domibusConnector-framework distribution. It focuses on how to embed the framework into a national connector, which is an application that needs to be built by the participant to connect the domibusConnector to its existing backend application. It also describes the interfaces of the domibusConnector.

3.1. Placement of the domibusConnector framework

In its current release the domibusConnector is capable to communicate with the gateway over the new domibus-connector-plugin. Assuming this is in place, the framework settles in that environment



The domibusConnector framework is to be embedded into another application (web or standalone) which integrates the framework and overrides its interfaces.

3.2. Installation

The domibusConnector framework can be installed into a top application by two different ways:

- Maven integration
- Manual integration by libraries

In both ways the top application needs to configure and load the properties as described in chapter [2.3. Properties](#).

3.2.1. Integration using Maven

If the top application is using Apache Maven, it is the easiest way to let Maven do the job. The Nexus repository needs to be configured with the following URL:

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

Once the repository is configured, only the `domibusConnectorController` needs to be added as a dependency;

```
<dependency>
  <groupId>eu.domibus.connector</groupId>
  <artifactId>domibusConnectorController</artifactId>
  <version>3.5-RELEASE</version>
</dependency>
```

All other parts of the `domibusConnector`, and required libraries, will load automatically.

3.2.2. Manual Integration

In the `DomibusConnectorDistribution-3.5-RELEASE-framework.zip` contains all JAR files of all modules needed for the framework.

Every library the framework depends on that is not available via public download is also shipped in the framework package in the “lib” sub-folder.

3.3. Interfaces

The `domibusConnector` in its framework version has some interfaces that must be implemented by the top application. Other interfaces are optional to be implemented and have default implementations.

Implementation-Classes of interfaces need to be configured in the properties. If no implementation is configured, a default implementation will be loaded. In case the interface is mandatory, the default implementation only throws an `ImplementationMissingException` as soon as the interface is called.

Mandatory interfaces are:

- `DomibusConnectorNationalBackendClient`

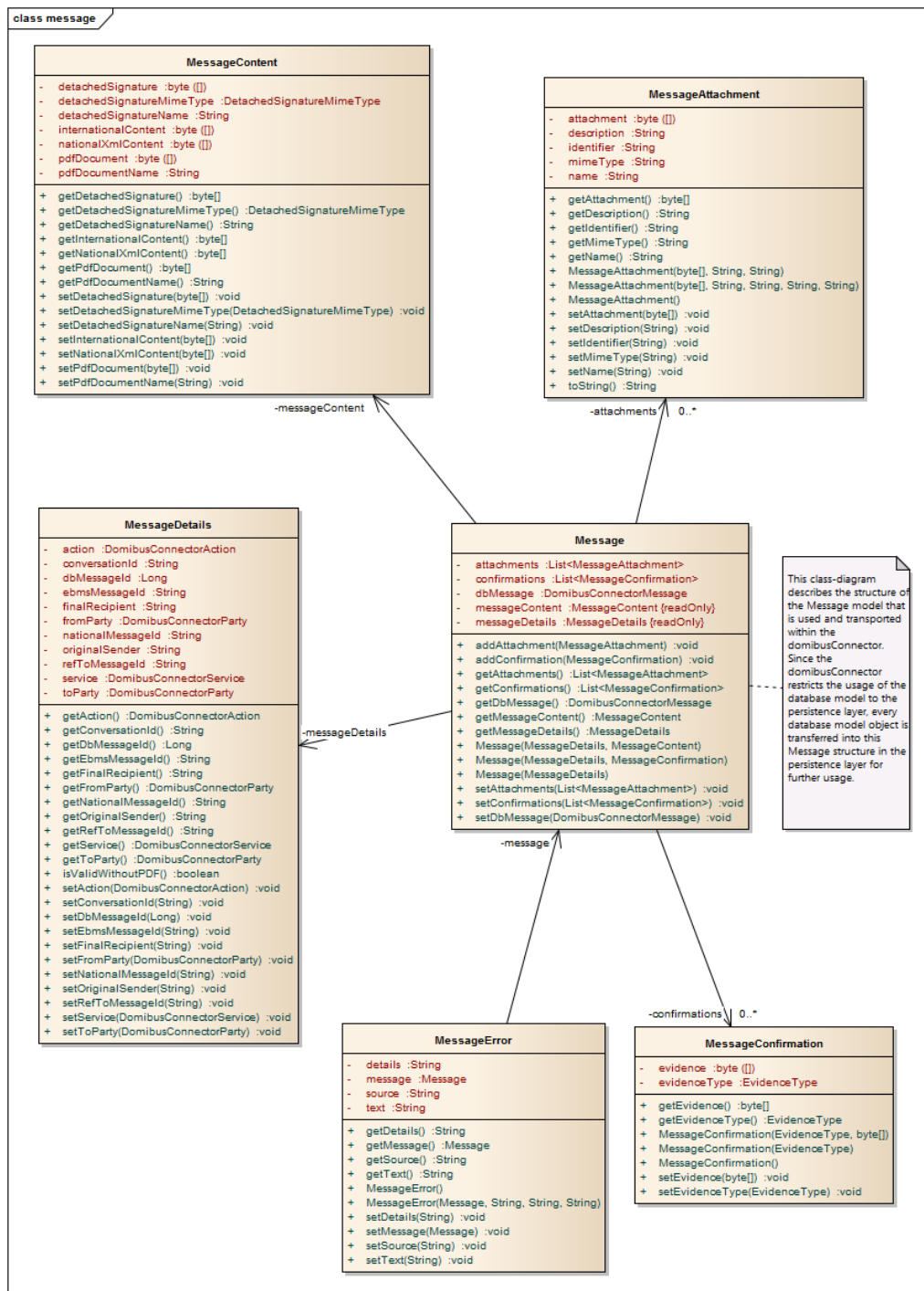
Optional interfaces are:

- `DomibusConnectorContentMapper`
- `DomibusConnectorSecurityToolkit`

3.3.1. DomibusConnectorNationalBackendClient

This interface is called every time the domibusConnector needs to communicate with the national backend. The interface is completely open to be implemented with every interface technology that suits the top application. This could be a webservice, message queues, file system interactions, whatever.

The methods of the interface are all well documented with javadoc. The domibusConnector deals with “Message” objects.



The data of those “Message” objects need to be filled as far as given and the domibusConnector extends it with every workflow step.

3.3.2. DomibusConnectorContentMapper

The DomibusConnectorContentMapper interface only has 2 methods:

```
package eu.domibus.connector.mapping;

import eu.domibus.connector.common.exception.ImplementationMissingException;

/**
 * Interface with methods to map national format XML to eCodex format and vice
 * versa. Inheritance must be configured.
 *
 * @author riederb
 */
public interface DomibusConnectorContentMapper {

    /**
     * Method to map international eCodex XML to national format. Must be
     * overridden when ContentMapper is used by configuration. The national xml
     * content will be written into the messageContent object.
     *
     * @param messageContent
     *      - a {@link MessageContent} object containing the eCodex xml
     *      Content.
     * @throws DomibusConnectorContentMapperException
     * @throws ImplementationMissingException
     */
    public void mapInternationalToNational(Message message) throws DomibusConnectorContentMapperException,
        ImplementationMissingException;

    /**
     * Method to map national XML to international eCodex format. Must be
     * overridden when ContentMapper is used by configuration. The eCodex xml
     * content will be written into the messageContent object.
     *
     * @param messageContent
     *      - a {@link MessageContent} object containing the national xml
     *      Content.
     * @throws DomibusConnectorContentMapperException
     * @throws ImplementationMissingException
     */
    public void mapNationalToInternational(Message message) throws DomibusConnectorContentMapperException,
        ImplementationMissingException;
}
```

The implementation of this interface should map the main content, transported as XML data, can be mapped from the national proprietary format into the common international format and back. If the content mapper is not used by configuration, the content will not be mapped, so the international content must already be put into the domibusConnector message in the right format.

3.3.3. DomibusConnectorSecurityToolkit

The default implementation that is already shipped with the domibusConnector builds and retrieves signed container which holds the messages’ human readable files and attachments. It also verifies the signature, a PDF file is signed with, generates a “TrustOKToken” in PDF and XML format to give the receiver of the message a verification result.

In fact it is very tricky to override this default implementation. One should be aware of the capabilities of the given security mechanisms.

4. domibusConnector-Standalone

4.1. Installing and starting the Standalone domibusConnector

To install the domibusConnector-Standalone, just download the distribution package “DomibusConnectorDistribution-3.5-RELEASE-Standalone.zip” from the e-Codex Nexus repository and unpack it anywhere on your file system. You then need to configure the properties for the domibusConnector following chapter [2.3. Properties](#) and store them as “connector.properties” inside the “conf” folder of your domibusConnector installation. After following the other [installation](#) steps described in this document you simply only have to run one of the startup scripts provided in the domibusConnector root folder.

4.1.1. Available startup scripts

There are 4 startup scripts available:

- DomibusStandaloneConnector.bat
This script is for MS Windows operating systems where you install the domibusConnector. It starts a console window where you can follow the output of the startup and also the logging output.
- DomibusStandaloneConnector.sh
This is the script for UNIX based systems. The output of the logger will appear following the execution of the script within the same session.
- DomibusStandaloneConnectorGUI.bat
Is doing exactly the same as “DomibusStandaloneConnector.bat” only with the difference, that it also starts the [GUI](#) of the domibusConnector too.
- DomibusStandaloneConnectorGUI.sh
Is doing exactly the same as “DomibusStandaloneConnector.sh” only with the difference, that it also starts the [GUI](#) of the domibusConnector too.

4.2. Sending a message with the Standalone Connector

In the connector.properties the folder for outgoing messages can be configured. When no folder is configured the domibusConnector listens at the default folder set, which is the „message/outgoing“ folder within the distribution package. To send a message first a message folder has to be prepared. The message folder needs to contain the following files:

- The message.properties -> see „Structure of the message.properties“
- A PDF file, the main document transported in the message (e.g. a form A of EPO)
- An XML file, containing the structured data representation of the main document (e.g. the structured data of a form A of EPO)

The following files can be added optionally:

- A file containing the detached signature with which the main document has been signed.
- Additional files to be attached to the message.

Please find here a screenshot with example files:

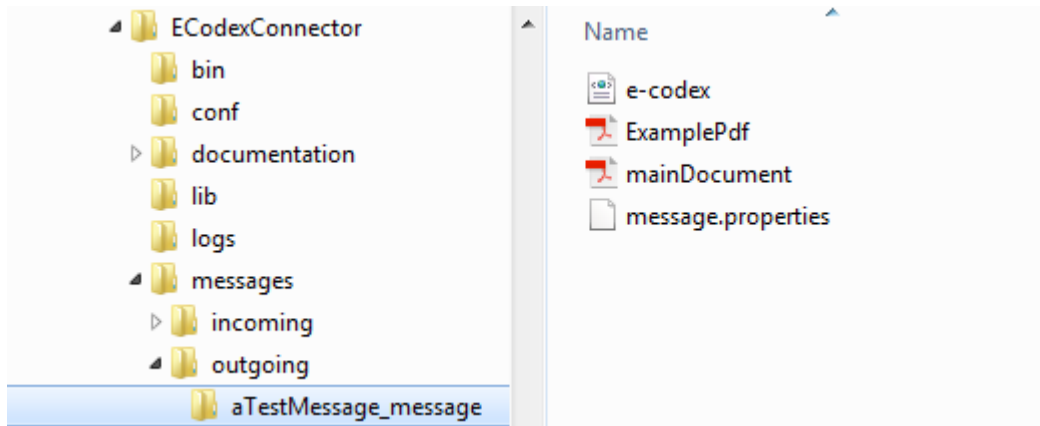


Figure: example files in outgoing folder

This folder holding an example message contains:

- e-codex.xml: The structured data representing the main document
- mainDocument.pdf: The main document of the message
- ExamplePdf: an additional file to be attached to the message
- message.properties: Holding the basic information for the message

In this example the message.properties file could be like this:

```
final.recipient=TargetCourtId
original.sender=LawyerId
from.party.id=AT
from.party.role=GW
to.party.id=DE
to.party.role=GW
service=EPO
action=Form_A
content.pdf.file.name=mainDocument.pdf
content.xml.file.name=e-codex.xml
```

Figure: property file example

Here the message goes from „LawyerId“ at the gateway AT-GW to the „TargetCourtId“ at the DE-GW gateway using EPO/Form_A.

The domibusConnector is triggered to process messages when inside the configured folder for outgoing messages there are folders ending with „_message“. The folders name before the ending „_message“ is not relevant. It can be, as here in the example „aTestMessage_message“, but only when it ends with „_message“ the connector will recognize it as a new message folder.

When the connector processed the message, the folder is renamed to the given national message id value plus „_sent“ postfix. In our example and for the case no national message id is given in the message.properties the connector creates one. This is done by generating a date/time value with the pattern „yyyyMMddhhmmssSSSS“ plus the original sender value. In our example this could be

„20150603091850133_LawyerId“. This national message id will then be saved to the message by the connector. The folder which contains the message will then be renamed with the national message id plus the „_sent“ postfix. So in our example the folder will then be „20150603091850133_LawyerId_sent“.

When the evidences to this message are produced or received, the DSAC identifies the corresponding message with the national message id and if the folder still exists, it automatically stores the evidences into the message folder. This looks like that:

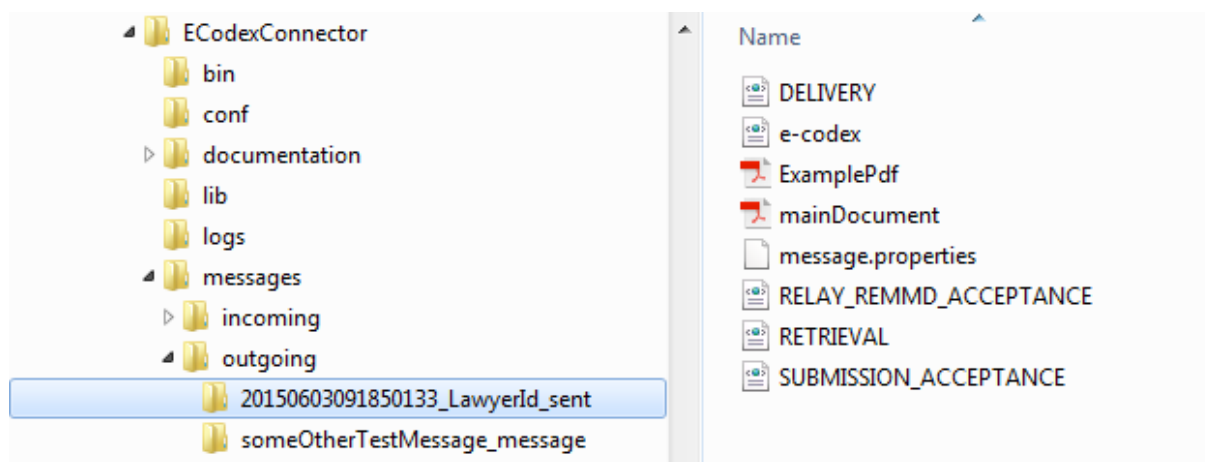


Figure: evidences in the folder of the sent message

As seen in the screenshot, the evidences are all stored in the message folder. If the folder does not exist anymore, the connector re-creates it.

Your message has now been sent successfully!

4.3. Sending a message with detached Signature

The only difference when sending a message with a detached signature is, that the name of the file holding the detached signature has to be set in the message.properties.

So if we got a message folder like this

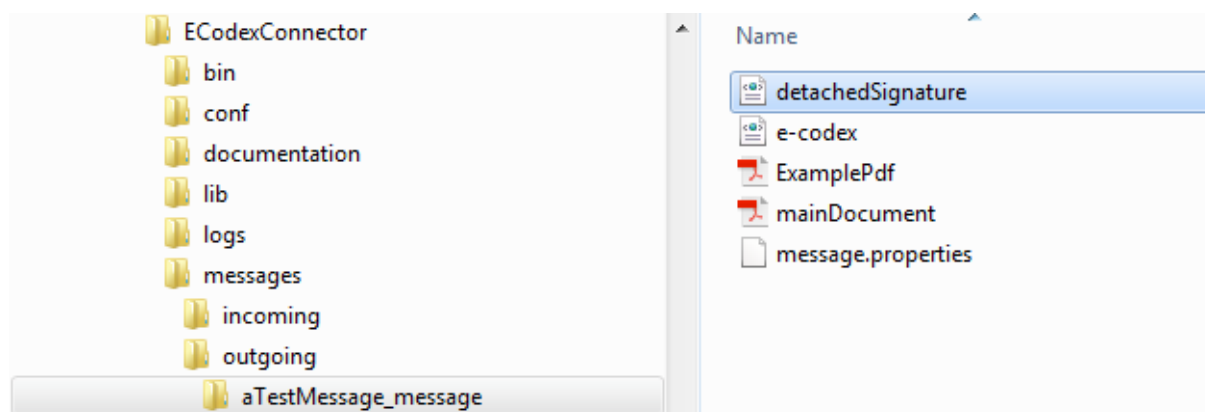


Figure: detached signature in message folder

the message.properties must be like this

```
final.recipient=TargetCourtId
original.sender=LawyerId
from.party.id=AT
from.party.role=GW
to.party.id=DE
to.party.role=GW
service=EPO
action=Form_A
content.pdf.file.name=mainDocument.pdf
content.xml.file.name=e-codex.xml
detached.signature.file.name=detachedSignature.xml
```

Figure: property file detached signature entry

The connector then recognizes the file as a detached signature file and it will be validated and sent together with the message.

4.4. Receiving a message with the Standalone Connector

Let's assume that the receiving gateway of our message above also uses an `domibusStandaloneConnector`. Then this connector will receive the message. The message will then be stored into the configured incoming message folder. If no incoming message folder is configured, it will be, by default, the „messages/incoming“ sub folder of the DSAC package.

Inside this incoming messages folder the connector creates a new folder for the received message. This message will be named with a generated date/time pattern „yyyyMMddhhmmssSSSS“ plus the gateway Id it has been received from. In our example this would be like „20150603093844987_AT“.

In this folder all the message files are stored.

For our above example this will be

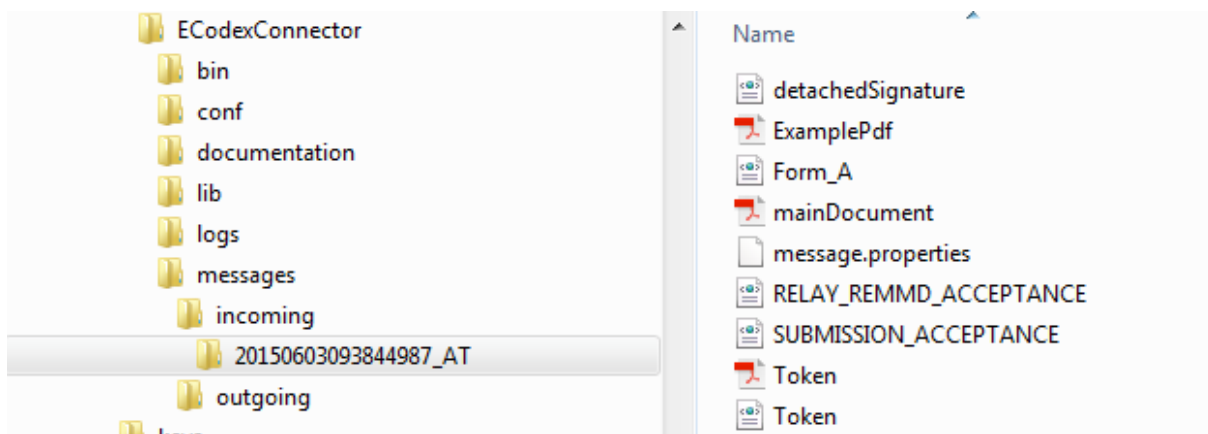


Figure: message received folder

Additionally to the message files already known from the sending side, the folder also contains the message.properties which are then generated by the connector. In our example they now look like

```
#Mon Jun 03 09:38:44 CEST 2015
to.party.id=DE
from.party.role=GW
national.message.id=20150603093844987_AT
original.sender=LawyerId
action=Form_A
content.xml.file.name=Form_A.xml
ebms.message.id=327fafcc-da53-46a9-9b3b-57854e37a9a4
content.pdf.file.name=mainDocument.pdf
service=EPO
to.party.role=GW
from.party.id=AT
detached.signature.file.name=detachedSignature.xml
final.recipient=TargetCourtId
```

Figure: property file received

Besides the data received from the sending side, these properties now also contain the ebms message id which is unique over all the gateways and generated by the sending gateway.

There are also the following files:

- Form_A.xml: This is the structured data file. As no name for this file is transported with the message, the connector names it after the action attribute the message was sent along with
- RELAY_REMMD_ACCEPTANCE.xml and SUBMISSION_ACCEPTANCE.xml: these are the two evidences already created for that message at that time.
- Token.pdf: When the sending connector validates the signature of the main document this Token document is generated. It is the human readable representation of the validation result.
- Token.xml: The structured representation of the validation result.

As the standalone connector treats received messages which are successfully stored as delivered, it automatically generates and sends the evidences DELIVERY and RETRIEVAL to the sending party.

4.5. Structure of the message.properties

In order to cover some data needed for message processing not included in the message files themselves there needs to be a file containing the needed data.

This file is built as a properties file with key/value pairs.

It is necessary to provide such a properties file for every message. The name of the message properties file can be configured within the connector.properties. By default it is „message.properties“.

Contents examples:

```
# the national message id. This is optional,
# if no national message id is given, the connector creates one
national.message.id=
# the finalRecipient and originalSender of the message
final.recipient=the final recipients name
original.sender=the original senders name
# The sending gateway party
from.party.id=
from.party.role=GW
# The receiving gateway party
to.party.id=
to.party.role=GW
# The service used
service=EPO
# The action used
action=Form_A
# The name of the PDF file representing the messages main document
content.pdf.file.name=
# The name of the XML file containing the structured data
# representing the message
content.xml.file.name=
# optional - if the main document PDF was signed with a
# detached signature this is provided by this file
detached.signature.file.name=
```

Figure: property file explanations

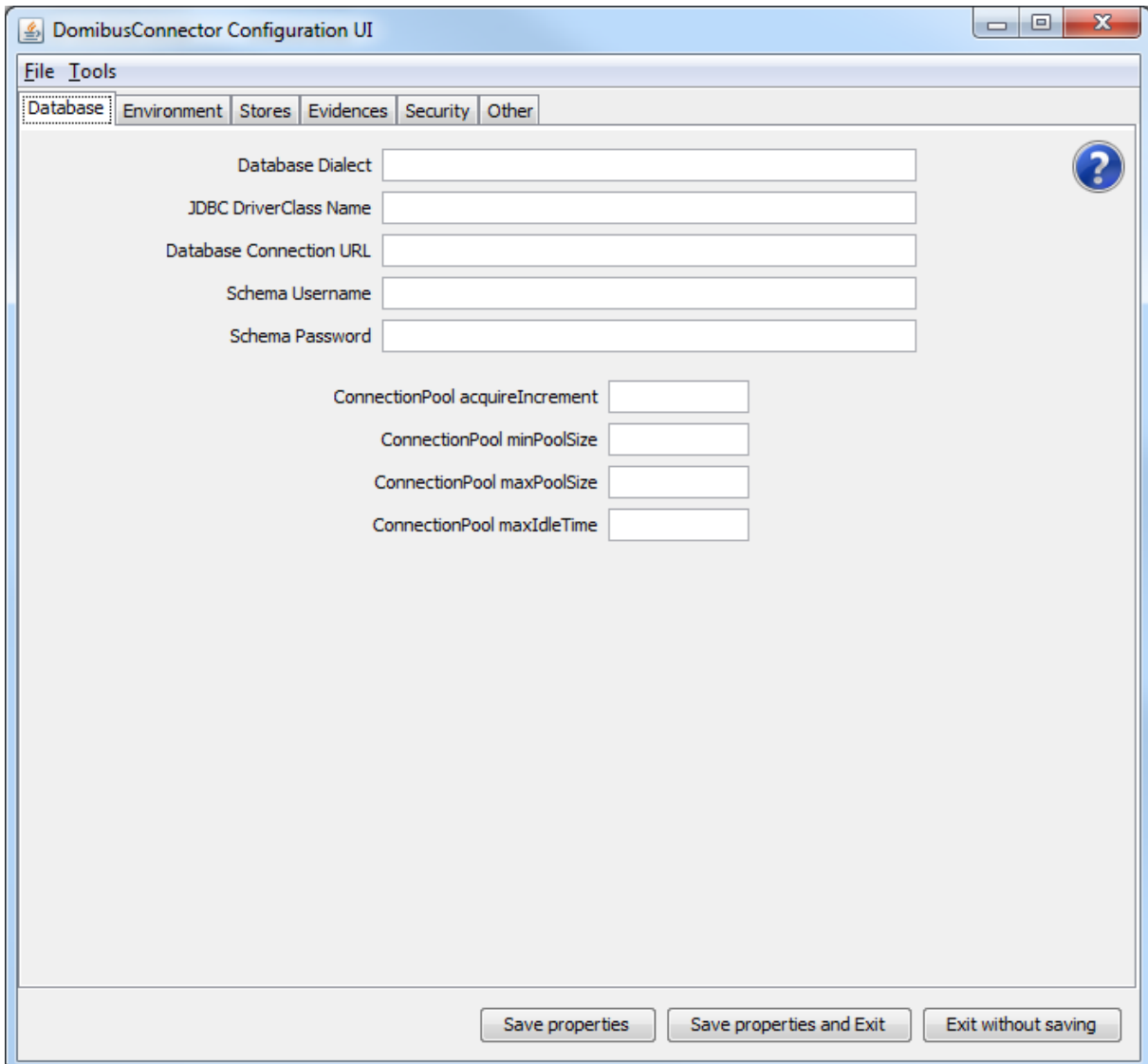
4.6. The domibusConnectorConfigurator

The domibusConnectorConfigurator is a graphical user interface (GUI) that helps to edit the connector.properties that are used connector-wide.

The connector.properties are necessary to run the domibusConnector as all settings for the domibusConnector environment are included in the file.

The domibusConnectorConfigurator is started automatically by the domibusStandaloneConnector if no „connector.properties“ file can be found in the „conf“ subfolder of the domibusStandaloneConnector installation directory.

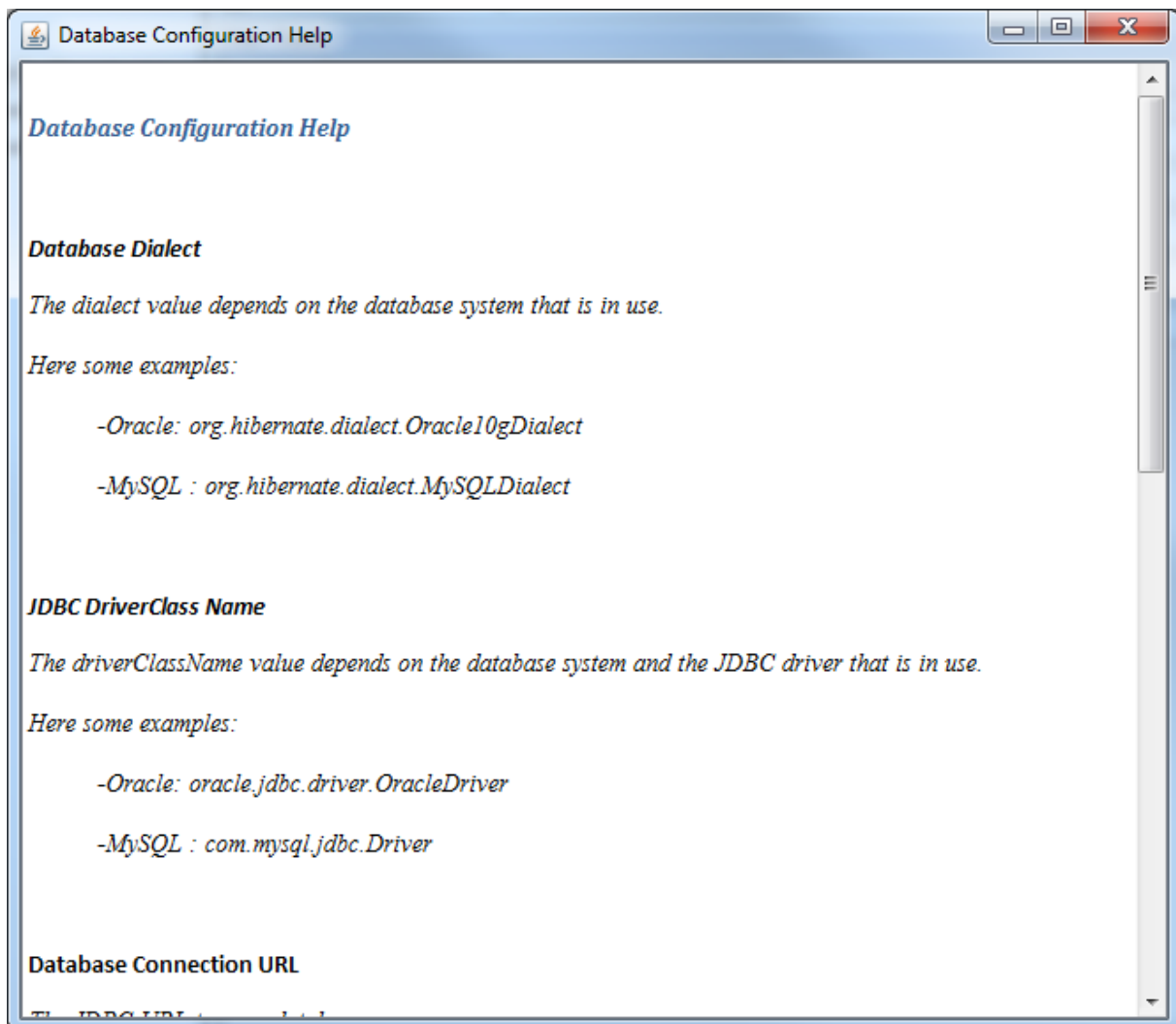
If the domibusConnectorConfigurator must be started manually (if, for example, properties are missing or need to be changed) this can be done by simply executing the „domibusConnectorConfigurator.jar“ file that is in the installation path of the domibusStandaloneConnector.



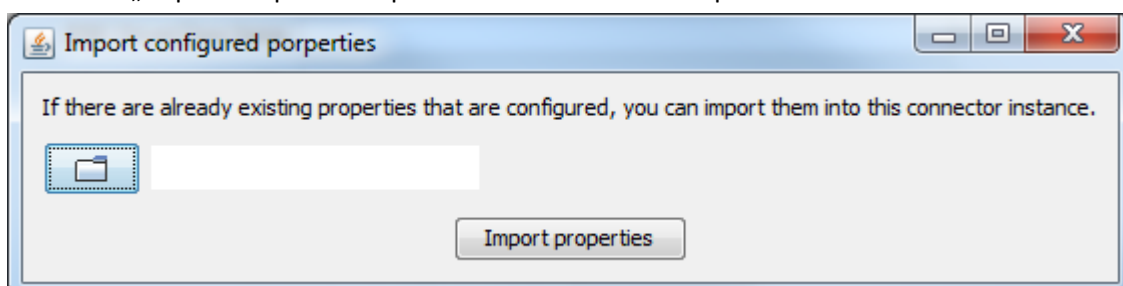
The starting screen of the domibusConnectorConfigurator shows

- The menu: This has „File“ and „Tools“ items. Within the „File“ section you can find 3 different, self explaining options on how the domibusConnectorConfigurator can be finished. The „Tools“ section gives the options to export or import the properties. Within every window that opens when selecting one option there are explanations on how to use the fields.
- The main section: This is splitted by several tabs that represent different groups of settings that must be set.
- The Button Bar: There you can see 3 different Buttons that give the possibilites of how to save/exit the domibusConnectorConfigurator.

In the main section, where the tabs with the grouped settings are, a question mark symbol can be found in every tab. When clicking on the symbol a help window opens that contains a supporting text for every field contained in the tab.



In the „Tools“ section of the menu two options for import/export of the connector.properties. When the „Import Properties“ option is chosen a window opens.



There the connector.properties from anywhere on the disk can be chosen to import it into the „conf“ folder of the domibusStandaloneConnector installation. After the file is copied, the domibusConnectorConfigurator refreshes itself with the values of the imported file.

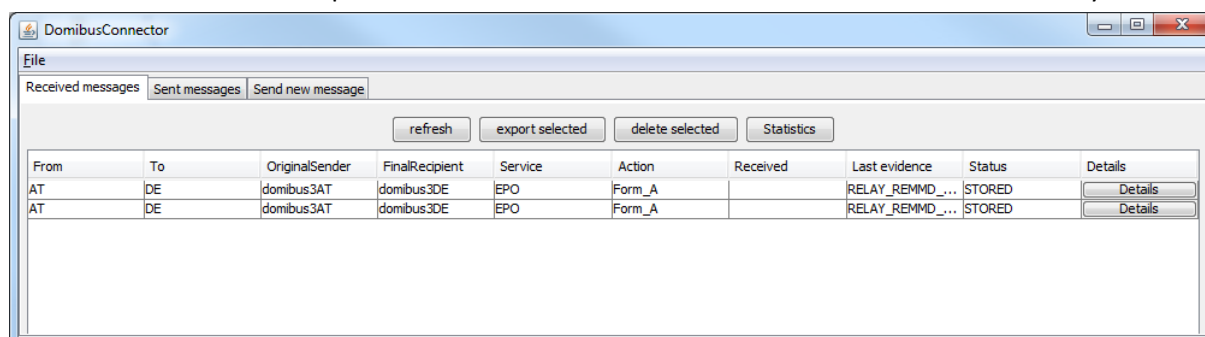
4.7. The domibusConnectorGUI

This GUI is designed to give a visualization and support for the functionalities of the domibusStandaloneConnector.

The GUI only relies on information of the file system structure of the domibusStandaloneConnector and is NOT using information of the domibusConnector database!

When the domibusStandaloneConnector starts it first initializes the domibusConnector framework components and all of its functionalities. If no „connector.properties“ can be found in the „conf“ folder of the domibusStandaloneConnector the framework does not start and The domibusConnectorConfigurator starts.

After the framework startup the domibusStandaloneConnector starts the GUI automatically:



The menu only shows a „File“ item that only gives the option to shutdown the domibusConnector. Be aware that this not only closes the GUI, but also shuts down the entire domibusConnector framework as well.

In the main section there are 3 tabs:

- Received messages: It shows a listing of all the messages that can be resolved from the configured „incoming.messages.directory“ from the connector.properties.
- Sent messages: Shows a listing of all the messages that can be resolved from the configured „outgoing.messages.directory“ from the connector.properties.
- Send new message: Gives the possibility to create a new message for the connector to send.

Received messages

If the property „incoming.messages.directory“ is set properly the GUI displays every message in that folder. The representation of the messages must stick to the description above in this Guide in chapter Receiving a message with the Standalone Connector.

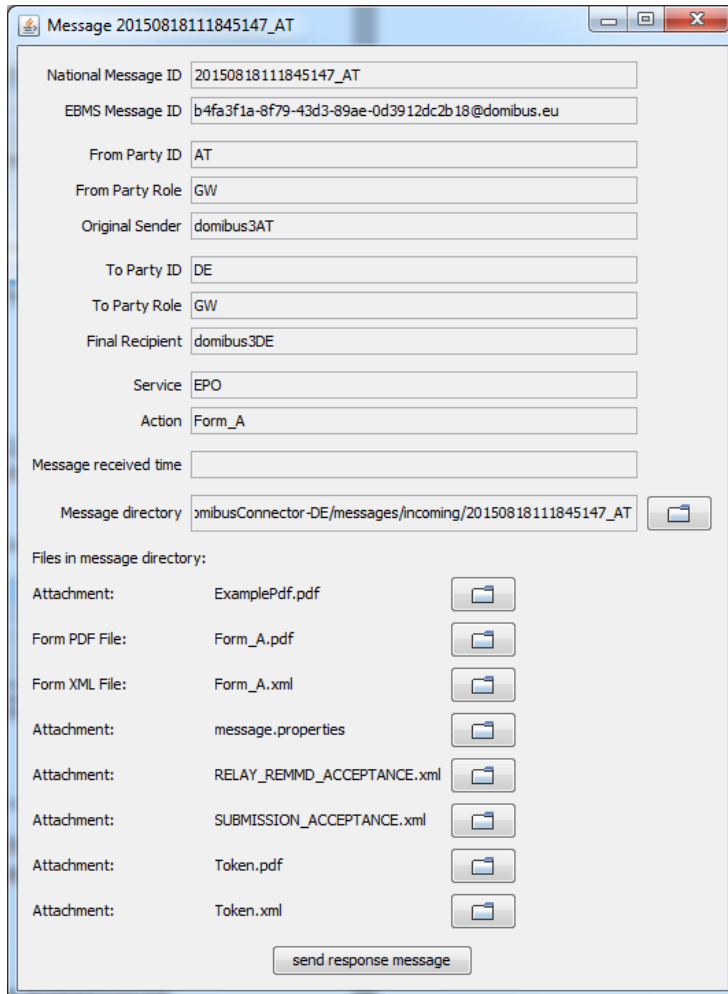
The information displayed in the list are read from the files inside the message directory. If there is no „message.properties“ file or if it is incomplete, no information can be displayed.

The buttons above of the listing give the following options:

- Refresh: simply refreshes the listing if there were any changes on the file system (for example: a new message was received by the domibusStandaloneConnector).
- Export selected: To use this functionality one or more messages in the listing must be selected. It give the possibility to export those selected messages to a place of choice. There is also a possibility to export the messages as zipped archives.

- Delete selected: To use this functionality one or more messages in the listing must be selected. It gives the possibility to delete the underlying message folders.
- Statistics: This only opens an information that statistical information can only be retrieved by the domibusWebAdmin.

Every message can be opened for details with the „Details“ button:



Message 20150818111845147_AT

National Message ID: 20150818111845147_AT

EBMS Message ID: b4fa3f1a-8f79-43d3-89ae-0d3912dc2b18@domibus.eu

From Party ID: AT

From Party Role: GW

Original Sender: domibus3AT

To Party ID: DE

To Party Role: GW

Final Recipient: domibus3DE

Service: EPO

Action: Form_A

Message received time:

Message directory: xmbusConnector-DE/messages/incoming/20150818111845147_AT

Files in message directory:

Attachment:	ExamplePdf.pdf	
Form PDF File:	Form_A.pdf	
Form XML File:	Form_A.xml	
Attachment:	message.properties	
Attachment:	RELAY_REMMD_ACCEPTANCE.xml	
Attachment:	SUBMISSION_ACCEPTANCE.xml	
Attachment:	Token.pdf	
Attachment:	Token.xml	

It shows more information from the „message.properties“ file, gives the option to open the message directory, lists all files contained in the message folder and opens the files if selected.

There is an additional functionality „send response message“. It creates a new message to be sent and uses information from the message details of this message for pre-filling of some fields.

Sent messages

If the property „outgoing.messages.directory“ is set properly the GUI displays every message in that folder. The representation of the messages must stick to the description above in this Guide in chapter Sending a message with the Standalone Connector.

The information displayed in the list are read from the files inside the message directory. If there is no „message.properties“ file or if it is incomplete, no information can be displayed.

DomibusConnector

File

Received messages | **Sent messages** | Send new message

refresh export selected delete selected Statistics

From	To	OriginalSender	FinalRecipient	Service	Action	Sent	Last evidence	Status	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	PROCESSING	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	FAILED	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	READY TO BE SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	test	test	EPO	Form_A		unknown	READY TO BE SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		unknown	READY TO BE SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		unknown	NEW	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		unknown	NEW	Details
DE	AT	domibus3DE	domibus3AT	EPO			unknown	NEW	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RELAY_REMMD_...	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	PROCESSING	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	FAILED	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	READY TO BE SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		SUBMISSION_RE...	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RELAY_REMMD_...	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		RETRIEVAL	SENT	Details
AT	DE	domibus3AT	domibus3DE	EPO	Form_A		DELIVERY	SENT	Details

A description of the buttons above of this listing can be found in the previous section of this document.

Every message in the list can be opened with the button „Details“.

If the message is in status „NEW“ the opening window gives all the information to the message known yet, to fill out additional information, add attachments and to send the message:

Message 20160519094421746_AT

National Message ID: 20160519094421746_AT

EBMS Message ID:

From Party ID: AT

From Party Role: GW

Original Sender: domibus3AT

To Party ID: DE

To Party Role: GW

Final Recipient: domibus3DE

Service: EPO

Action: Form_A

Message directory: busConnector-AT/messages/outgoing/20160519094421746_AT_new

Form XML File: e-codex.xml

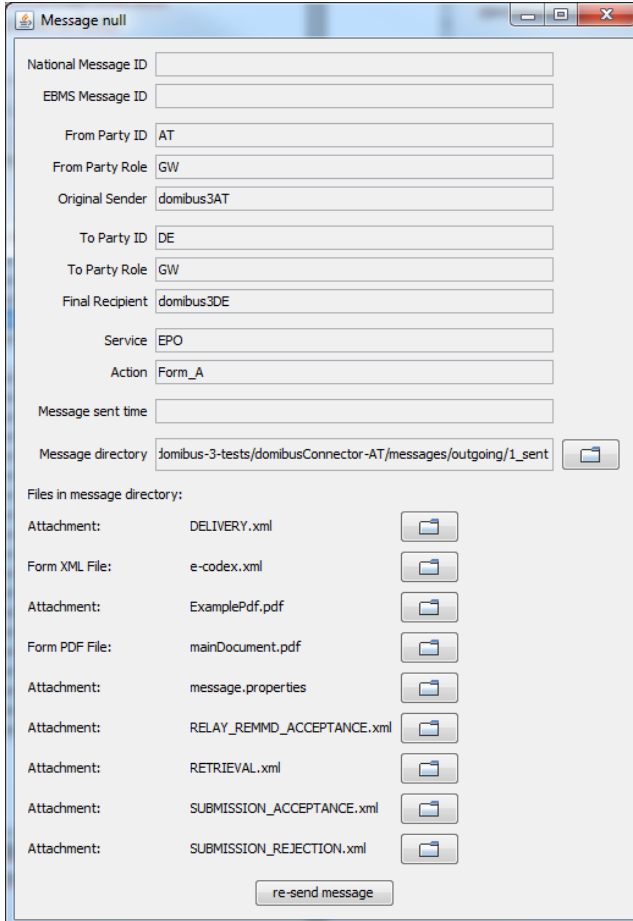
Form PDF File: mainDocument.pdf

Attachment 1: ExamplePdf.pdf

Attachment 2:

send message

With any other status than „NEW“ the message details are displayed:



The 'Message null' window displays the following details:

- National Message ID: [empty]
- EBMS Message ID: [empty]
- From Party ID: AT
- From Party Role: GW
- Original Sender: domibus3AT
- To Party ID: DE
- To Party Role: GW
- Final Recipient: domibus3DE
- Service: EPO
- Action: Form_A
- Message sent time: [empty]
- Message directory: domibus-3-tests/domibusConnector-AT/messages/outgoing/1_sent

Files in message directory:

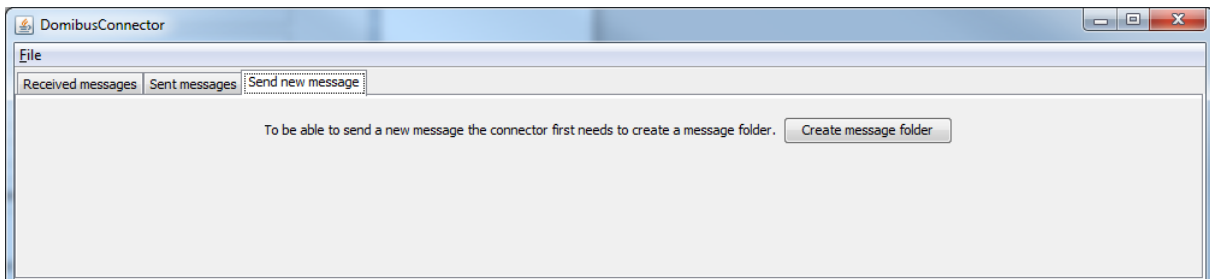
Attachment:	File Name	Icon
Attachment:	DELIVERY.xml	[Folder icon]
Form XML File:	e-codex.xml	[Folder icon]
Attachment:	ExamplePdf.pdf	[Folder icon]
Form PDF File:	mainDocument.pdf	[Folder icon]
Attachment:	message.properties	[Folder icon]
Attachment:	RELAY_REMMD_ACCEPTANCE.xml	[Folder icon]
Attachment:	RETRIEVAL.xml	[Folder icon]
Attachment:	SUBMISSION_ACCEPTANCE.xml	[Folder icon]
Attachment:	SUBMISSION_REJECTION.xml	[Folder icon]

re-send message

This window is almost the same as that of the details of a received message. But instead of sending a response message it gives the possibility to „re-send message“.

If selected a new message will be created with all the information and attachments of the previous message.

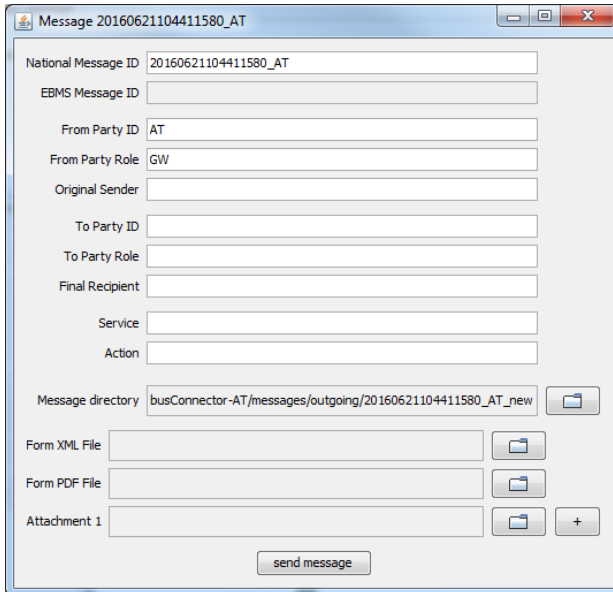
Send new message



The DomibusConnector window shows the 'Send new message' tab. It contains the following text and button:

To be able to send a new message the connector first needs to create a message folder. [Create message folder](#)

To be able to send a new message, a message folder must be created.



Then the message window opens with the information filled that the domibusConnector itself can fill. It is a generically generated National Message ID, the From Party information and the Message directory that is already created.

All the other fields must be filled manually.

Also the necessary files for the message must be selected.

The Form XML File and the Form PDF File must be present and syntactically correct. They cannot be created or edited by the domibusConnector.

After the „send message“ Button is clicked the message folder is prepared for the domibusStandaloneConnector to be picked up and the message sent.