

# domibus Standalone Connector - Guide

This is a guide to show how the Standalone Connector (DSAC) works and can be used.

## Table of contents

Preconditions: .....	1
Configuring the Standalone Connector .....	1
Sending a message with the Standalone Connector .....	2
Sending a message with detached Signature .....	4
Receiving a message with the Standalone Connector .....	4
Structure of the message.properties .....	6
The domibusConnectorConfigurator .....	7
The domibusConnectorGUI .....	9

## Preconditions:

The pre-requisite for setting up the DSAC is the setup of a Domibus Gateway. Please refer to the documentation which comes together with the Gateway for doing this.

Supported operating systems are Windows based or Unix based systems. The domibusConnector-Standalone.zip package has to be downloaded and unzipped to any place in the file system. A database must be set up. Supported database vendors are MySQL (5) and Oracle (10, 11). Prepared database scripts can be found in the documentation section of the package.

## Configuring the Standalone Connector

To configure the Standalone Connector the following steps have to be done:

- Database setup: You need to set up a database for the connector. There are prepared scripts for MySQL and Oracle included in the DSAC package documentation folder. Running the initial scripts of your DB vendor on the database should do the job.
- Connector.properties: A properties file is needed to set up the domibusConnector environment. This file can be already edited manually and imported or completely set up new. This can be done The domibusConnectorConfigurator. If no properties can be found in the “conf” folder of the domibusStandaloneConnector installation, the domibusConnector automatically starts the domibusConnectorConfigurator by itself.
- Logging.properties: The underlying logging framework used by the connector is slf4j which is an abstraction level for java logging. In the particular case of the connector log4j is used as the implementor for logging. Therefore the corresponding configuration has to be set up to

run properly. This is done by default in „conf/log4j.properties“ in the DSAC package. If it is used as it is, it logs in the „logs“ folder of the package. This setting can also be overwritten.

## 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 connector listens at the default folder set, which is the „message/outgoing“ folder within the DSAC 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:

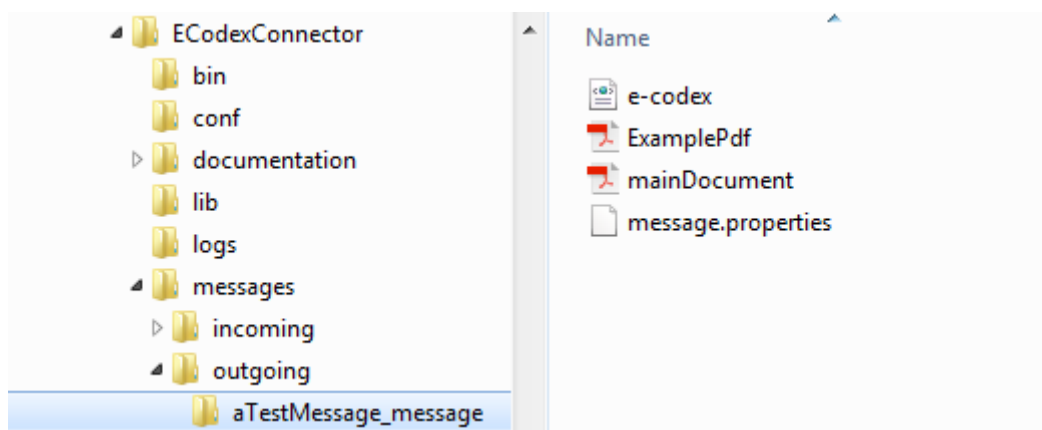


Figure: example files in outgoing folder

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“ over the gateway AT-GW to the „TargetCourtId“ at the DE-GW gateway using EPO/Form\_A.

The connector 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 irrelevant. 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:

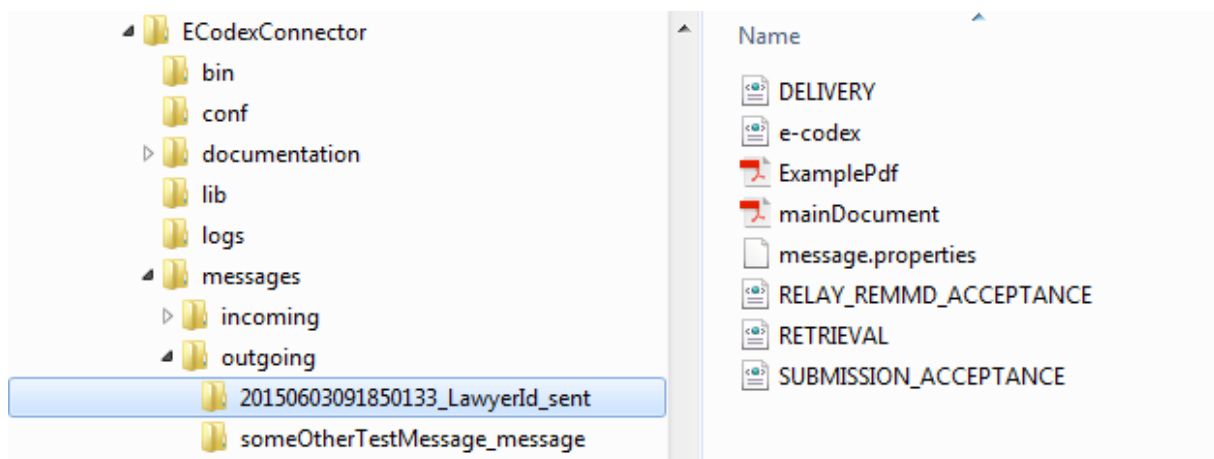


Figure: evidences in the folder of the sent message

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!

## 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

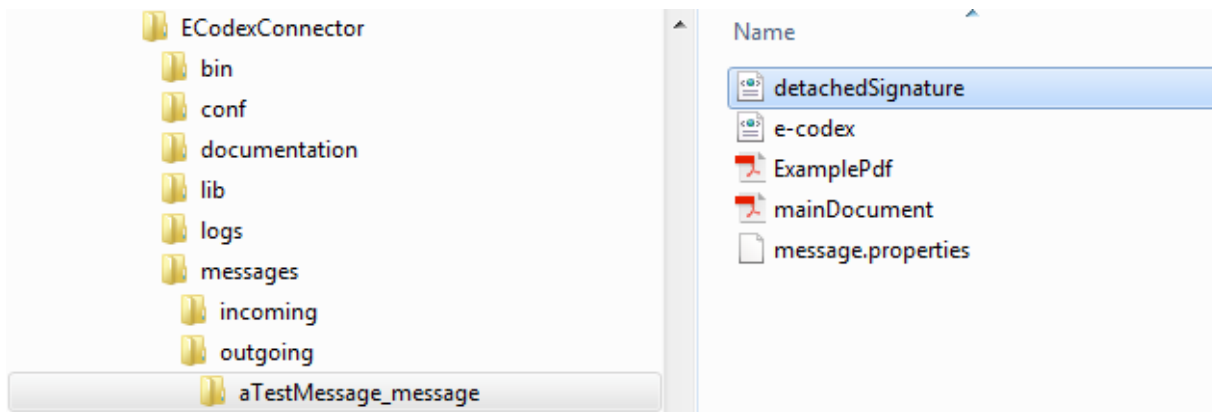


Figure: detached signature in message folder

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.

## 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

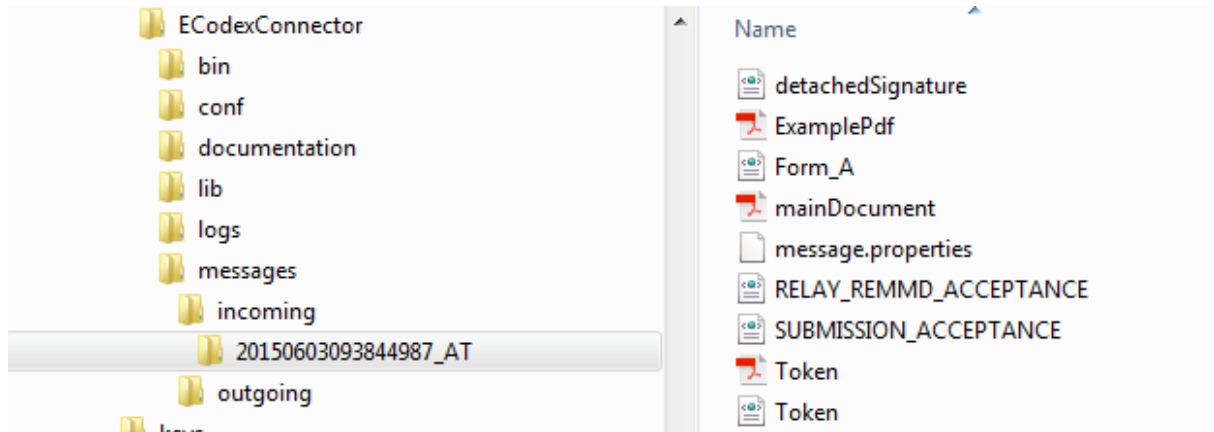


Figure: message received folder

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.

## 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 messag 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

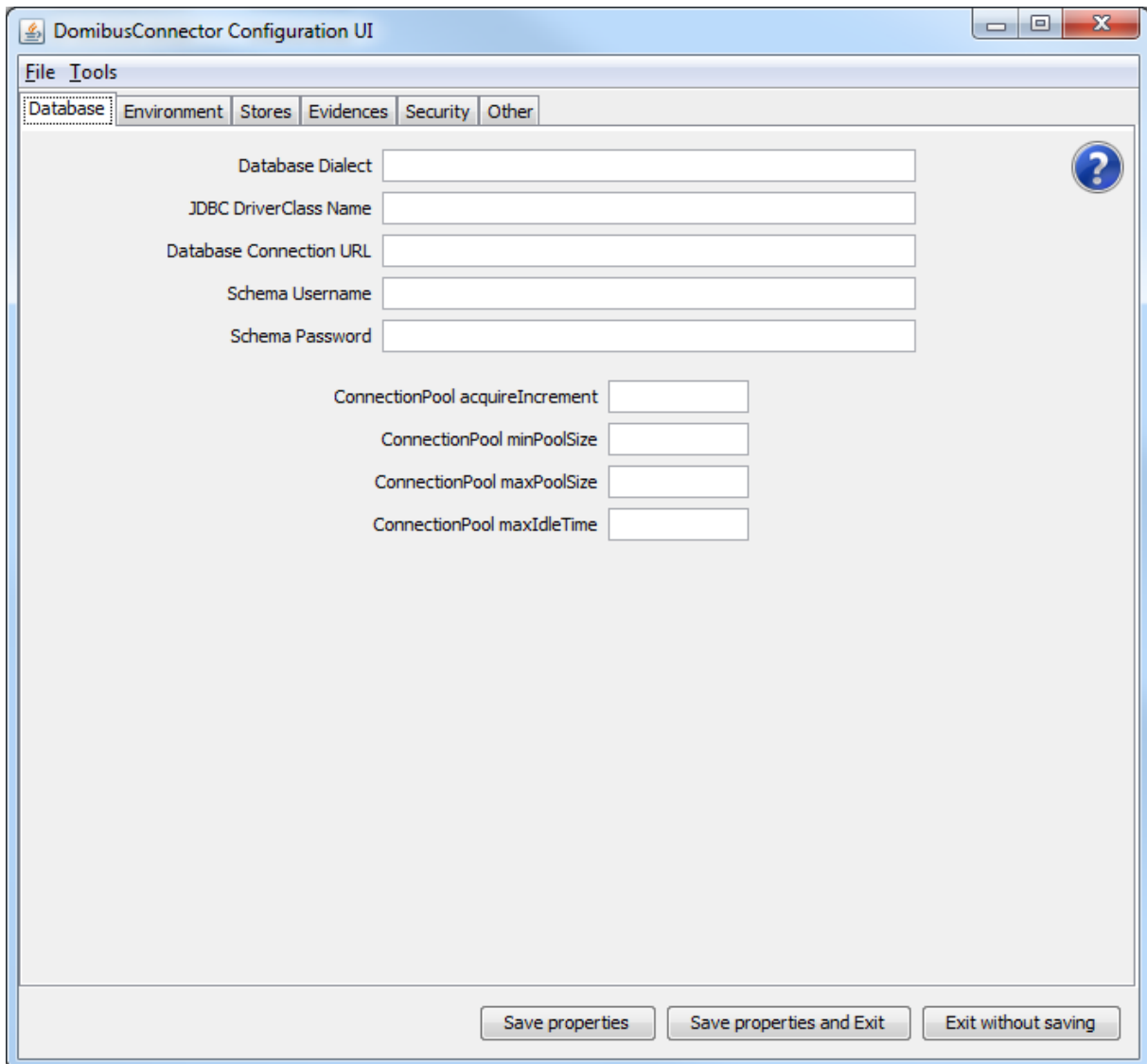
## The domibusConnectorConfigurator

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.

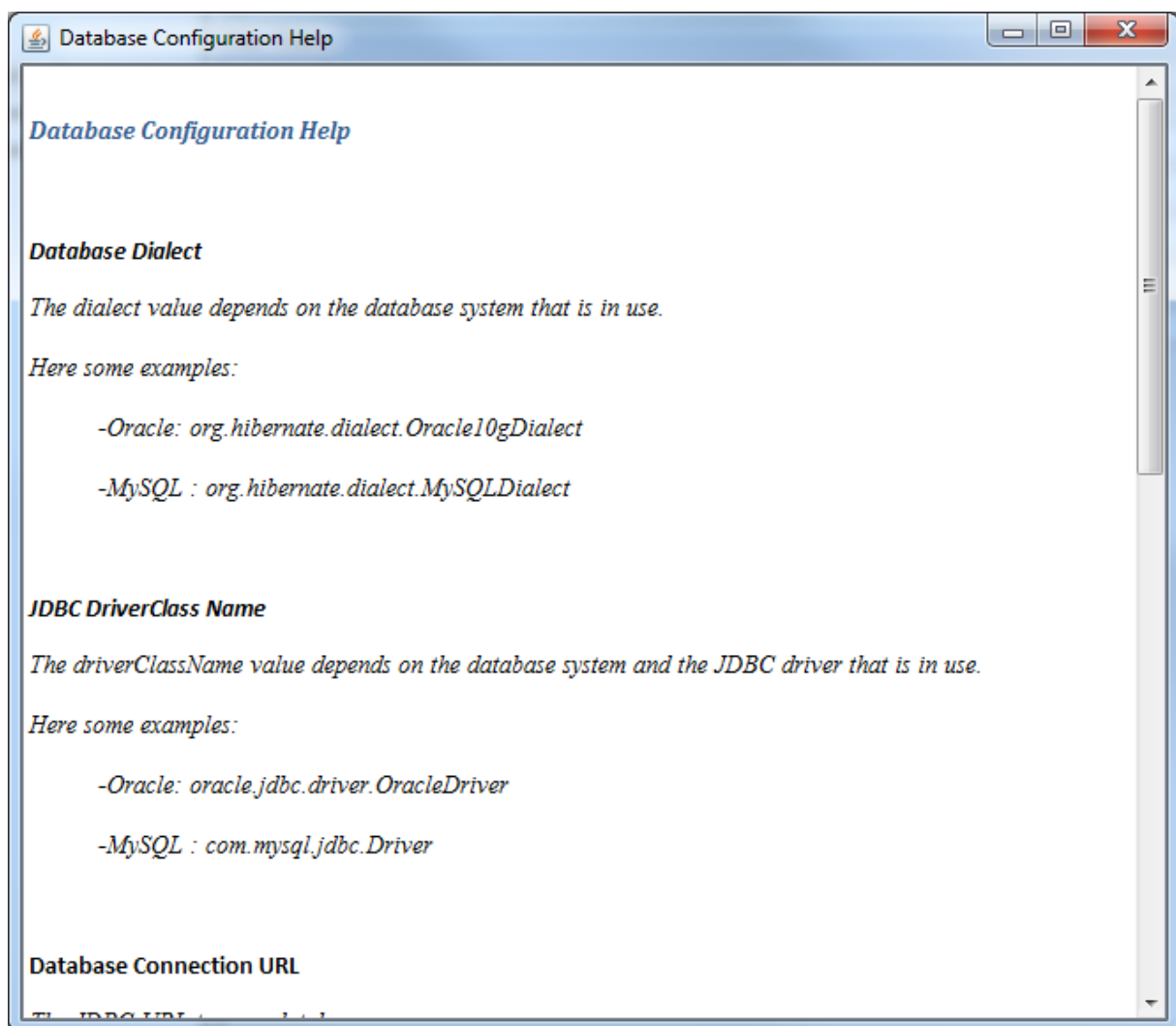
If the domibusConnectorConfigurator must be started manually (if, for example, properties are missing or need to be changed) this can be done by simply executing the „domibusConnectorConfigurator.jar“ file that is in the installation path of the domibusStandaloneConnector.



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 possibilities of how to save/exit the domibusConnectorConfigurator.

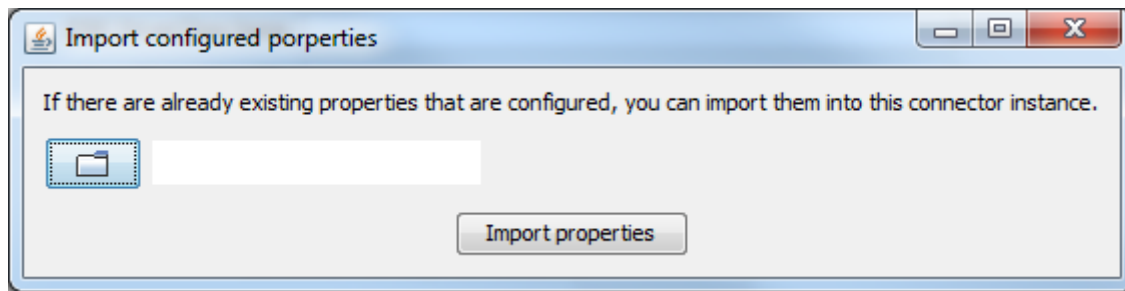
In the main section, where the tabs with the grouped settings are, a question mark symbol can be found in every tab. When clicking on the symbol a help window opens that contains a supporting text for every field contained in the tab.



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.

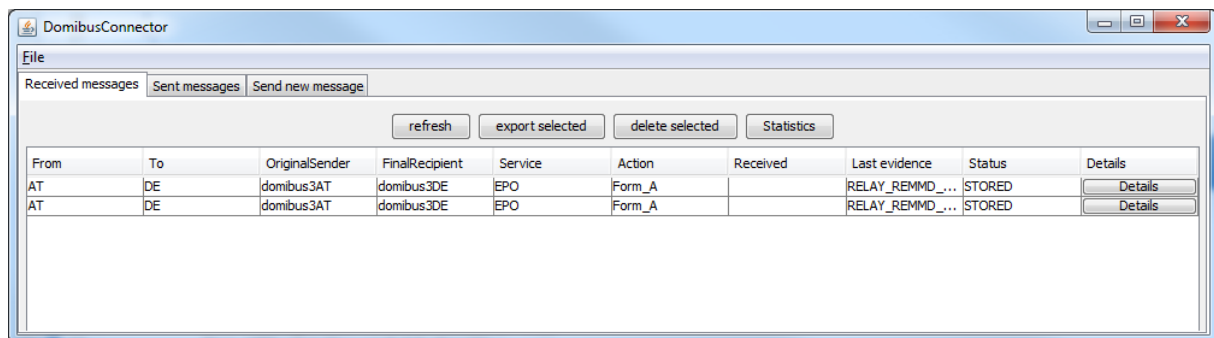
## The domibusConnectorGUI

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 give the possibility to delete the underlying message folders.
- Statistics: This only opens an information that statistical informations can only be retrieved by the domibusWebAdmin.

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

The screenshot shows a window titled "Message 20150818111845147\_AT". It contains the following fields and sections:

- 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: (empty field)
- Message directory: xmibusConnector-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
- send response message button

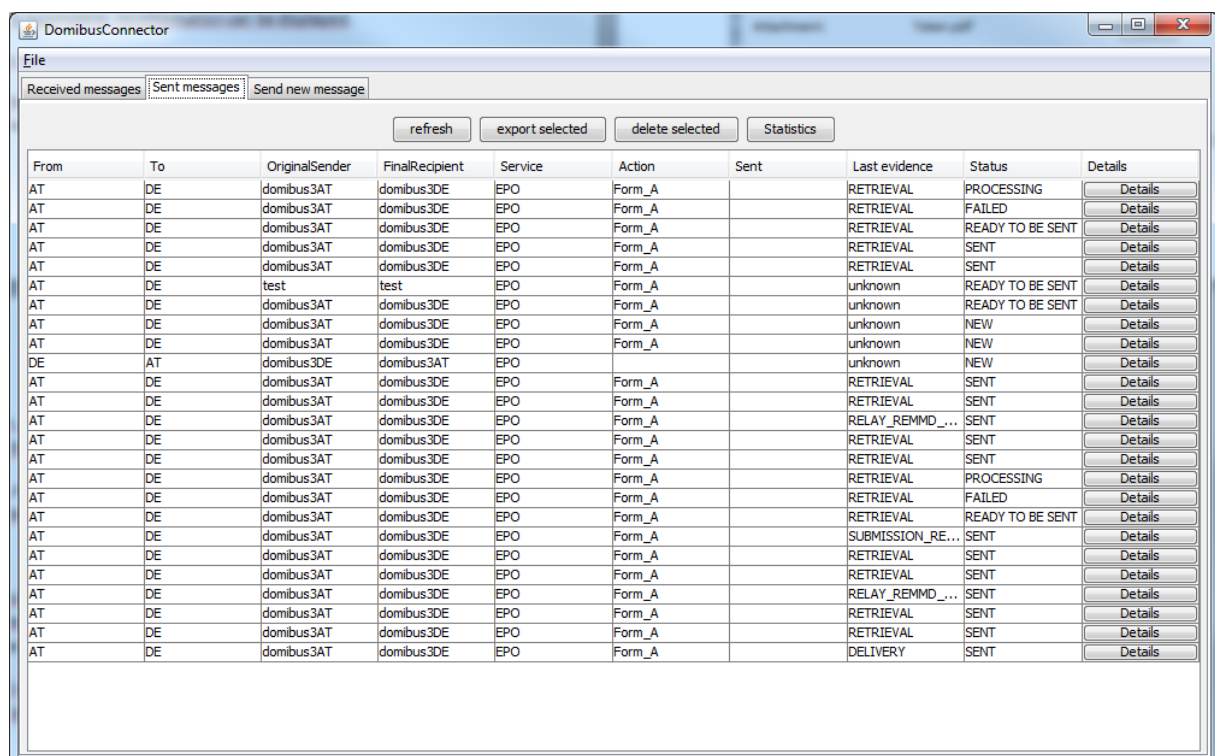
It shows more information from the „message.properties“ file, gives the option to open the message directory, lists all files contained in the message folder and opens the files if selected.

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.



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	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:

Message null

National Message ID:

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 sent time:

Message directory: domibus-3-tests/domibusConnector-AT/messages/outgoing/1\_sent

Files in message directory:

Attachment: DELIVERY.xml

Form XML File: e-codex.xml

Attachment: ExamplePdf.pdf

Form PDF File: mainDocument.pdf

Attachment: message.properties

Attachment: RELAY\_REMMD\_ACCEPTANCE.xml

Attachment: RETRIEVAL.xml

Attachment: SUBMISSION\_ACCEPTANCE.xml

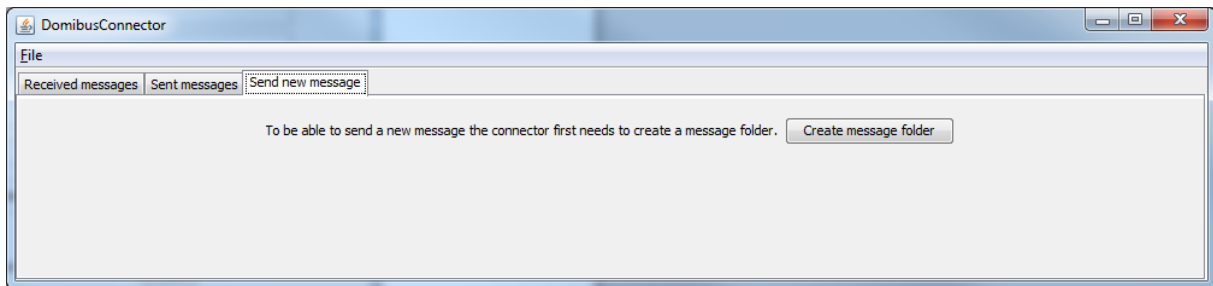
Attachment: SUBMISSION\_REJECTION.xml

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



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

The screenshot shows a window titled 'Message 20160621104411580\_AT'. It contains several input fields for message details: 'National Message ID' (pre-filled with '20160621104411580\_AT'), 'EBMS Message ID', 'From Party ID' (pre-filled with 'AT'), 'From Party Role' (pre-filled with 'GW'), 'Original Sender', 'To Party ID', 'To Party Role', 'Final Recipient', 'Service', and 'Action'. Below these is a 'Message directory' field with the path 'busConnector-AT/messages/outgoing/20160621104411580\_AT\_new' and a folder icon button. At the bottom, there are three fields for attachments: 'Form XML File', 'Form PDF File', and 'Attachment 1', each with a folder icon button. A 'send message' button is located at the very bottom.

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.