domibus Standalone Connector - Guide

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

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: Within the DSAC package there is an example properties configuration in the „conf“ folder. This should be adapted for your environment. Each Property given is documented inline for better understanding.
- 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:

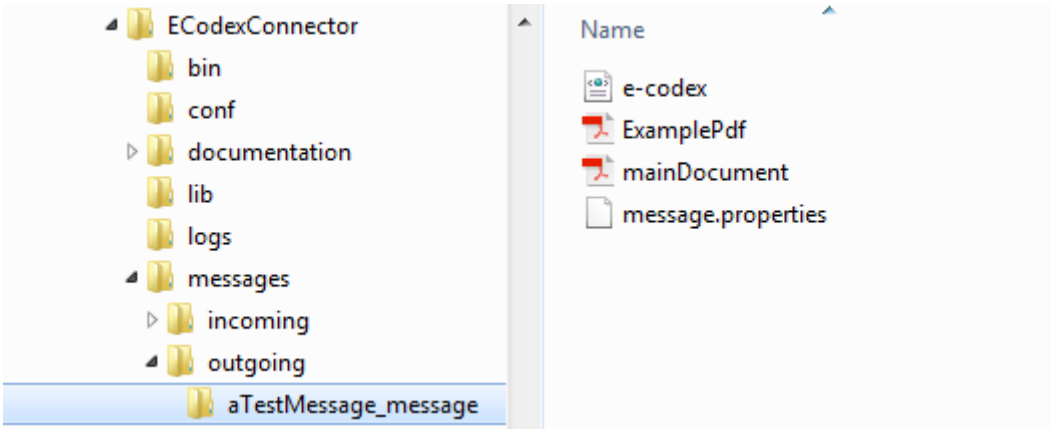

Figure: example files in outgoing folder

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:

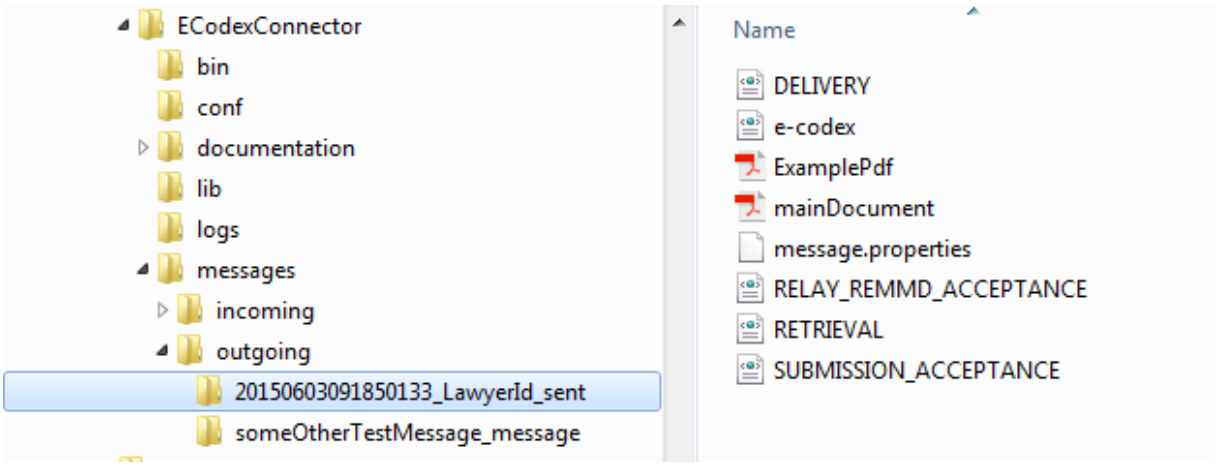

Figure: evidences in the folder of the sent message

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

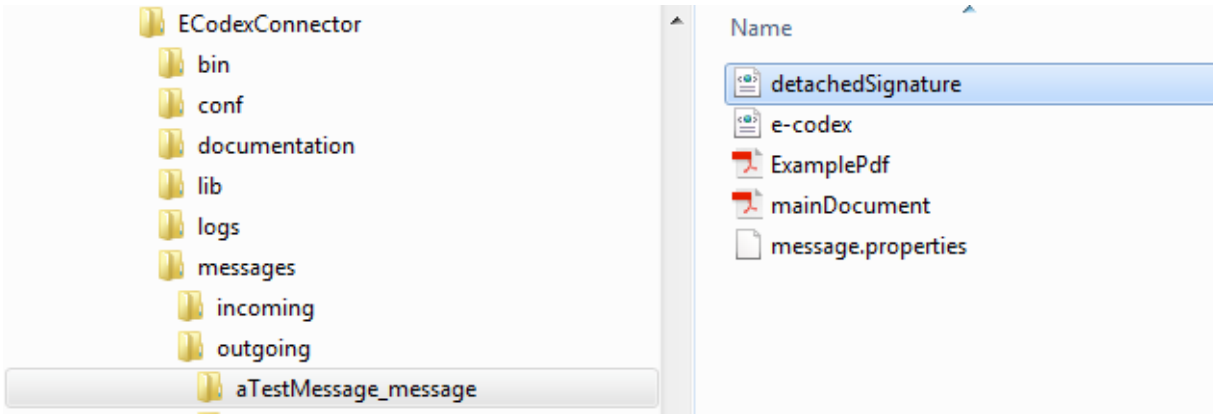

Figure: detached signature in message folder

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

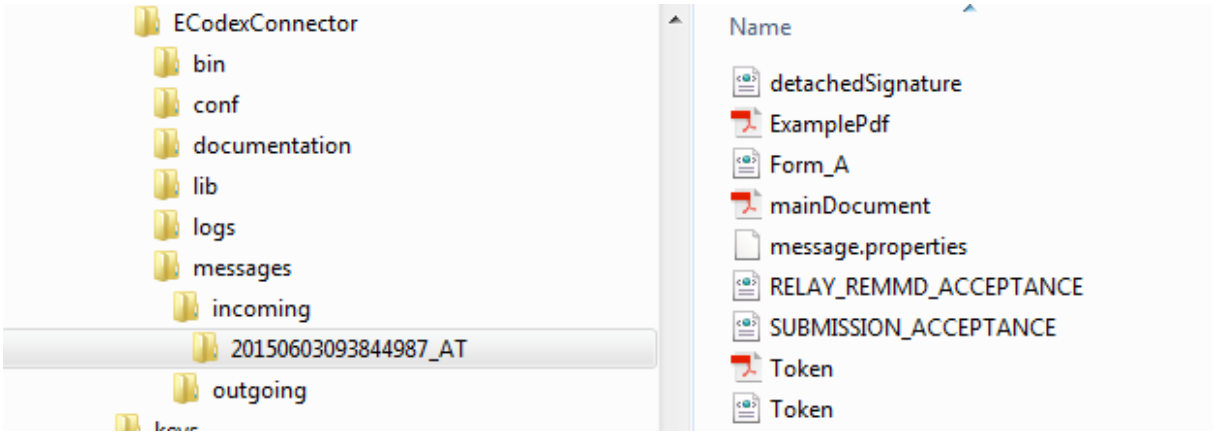

Figure: message received folder

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