**ECodex Standalone Connector - Guide**

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

**Preconditions:**

Supported operating systems are Windows based or unix based. The ECodexConnector-Standalone.zip package has to be downloaded and unzipped at 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 fort he connector. There are prepared scripts for MySql and Oracle included in the ESAC package documentation folder. Running the initial scripts of your db vendor on the database should do the job.
* Connector.properties: Within the ESAC 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 ist slf4j which is an abstraction level for java logging. In the particular case oft he connector log4j is used as the implementor for logging. Therefore the corresponding configuration hast o be set up to run properly. This is done by default in „conf/log4j.properties“ in the ESAC package. This can be used as is, then it logs in the „logs“ folder oft he package, or can be overridden.

**Sending a message with the Standalone Connector**

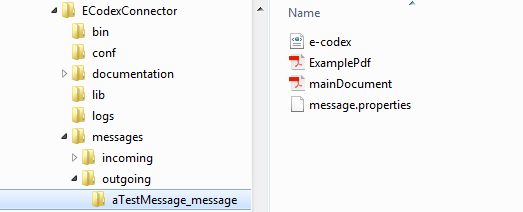
In the connector.properties the folder for outgoing messages can be configured. When no folder is configured the connector listens at the default folder set. It ist he „message/outgoing“ folder within the ESAC package. To send a message first a message folder has to be prepared. The message folder needs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

* The message.properties -> see „Structure of the message.properties“
* A PDF file, the main document transported in the message
* A XML file, containing the structured data representation oft he main document

Optionally the following files can be added:

* A file containing the detached signature with which the main document was signed.
* Additional files to be attached to the message.

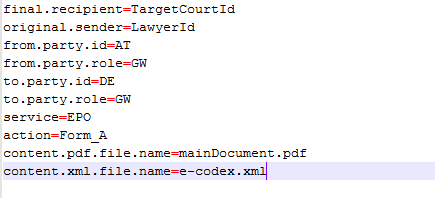
Find here a screenshot with example files:



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 informations for the message

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

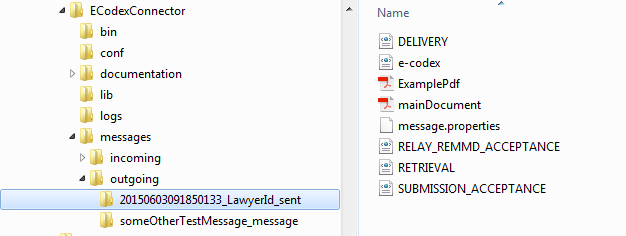


Here the message goes from „LawyerId“ over the gateway AT-GW to the „TargetCourtId“ at the DE-GW gateway using EPO/Form\_A.

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 no national message id is given in the message.properties, so 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 tot he 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 ESAC 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 will then look like:



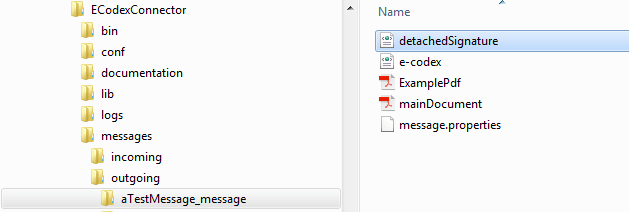
As seen in the screenshot, the evidences are all stored in the message folder. If the folder does not exist anymore, the connector re-creates it.

Your message is now sent successfully.

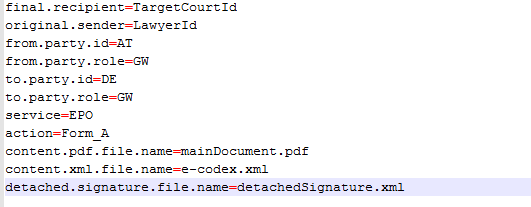
**Sending a message with detached Signature**

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



the message.properties must be like this



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

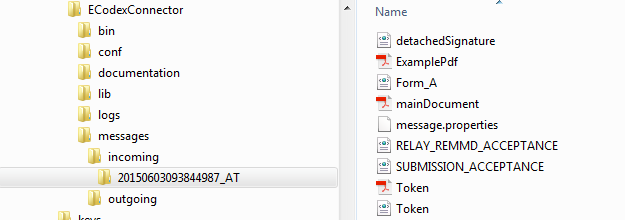
**Receiving a message with the Standalone Connector**

Let’s assume that the receiving gateway of our message above also uses an ECodexStandaloneConnector. 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 ESAC package.

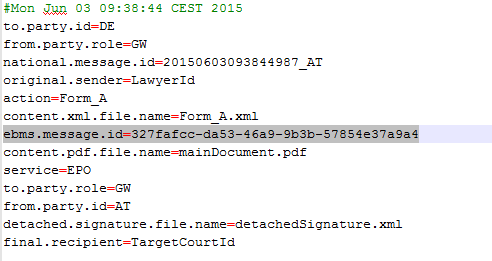
Inside this incoming messages folder the connector creates a new folder for the received message. This message will be named with a generated date/time pattern „yyyyMMddhhmmssSSSS“ plus the gateway it was received from Id. In our example this would be like „20150603093844987\_AT“.

In this folder all the message files will be stored.

For our above example this will be



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



Besides the data we already know from the sending side, this 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 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: That 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**

To cover some detailed 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:

