



Software Engineering

Open eGov MessageHandler v3.4

Technical Documentation

GLUE Software Engineering AG explicitly draws attention to possible changes of this document due to technical or functional progress without further notice and without Justification.

The copyright of this document is owned by **GLUE Software Engineering AG**. The document reflects the current stage of development at the time of writing or changing.

Copyright © 22.02.2024 by

GLUE Software Engineering AG

Zieglerstrasse 34

CH – 3007 Bern

Document Administration	
Author(s)	Kasimir Blaser, Igor Metz, Marc Zweiacker, Beat Weisskopf, Roland Keller
Doc.-Tool	Microsoft Word 2010
Storage	G:\kunden\Open_eGov\MessageHandler\handbuch\OE MessageHandler V3.4.4_v1.23.docx
Distribution	Public

Version	1.23	Release Date	22.02.2024
Function			
Date			
Signature			

Version History			
Version	Description	Date	Initials
0.1	Draft based on the German documentation (version 1.7) as well as the eSchKG Red Book by Marc Zweiacker	24.07.2012	KB
0.2	Revised and extended	15.08.2012	IM
0.3	Changes for the new config.xml structure	06.11.2012	KB
0.4	Minor corrections	15.11.2012	IM
0.9	Major modifications after feedback of M. Zweiacker	18.12.2012	IM
0.10	Change config.xsd. New element "localRecipients"	08.01.2013	KB
0.11	Major modifications after feedback of M. Zweiacker	18.01.2013	IM
0.12	Minor fixes. "createPerMessageProtocols", "recipientId"	21.01.2013	KB
1.0	Changes to local recipient documentation; First public release	27.01.2013	IM
1.1	Chap. 3.1: note for 64-bit installations	09.02.2013	IM



1.2	Chap. 3.1: note for Java version 7	25.02.2013	KB/IM
1.3	Adapted to v 3.1. release	11.03.2013	KB/IM
1.4	Chap. 4.9.1 recipientIdResolver.groovy replaced; Release notes for 3.1.1	13.03.2013	IM
1.5	Clarification in chap. 5.2.1; Release notes for 3.1.2	24.05.2013	IM
1.6	Release notes for 3.1.3	07.08.2013	KB
1.7	Release notes for 3.1.4	22.08.2013	IM
1.8	Release notes for 3.1.5 added	08.11.2013	IM
1.9	Release notes for 3.1.6 added	27.01.2014	KB
1.10	Release notes for 3.2.0, chap. 2.5.5 added, needs Java7 (and JCE), installation packages split for Windows / Linux	26.08.2015	BW
1.11	Release notes for 3.3.0 and 3.3.1 added; note added to chap. 2.5.5; several minor corrections	24.02.2016	IM
1.12	Extended chap 3.1; highlighted the importance of Unlimited Strength Jurisdiction Policy Files	01.03.2016	IM
1.13	Added instructions about Windows Installer package	12.04.2016	AG
1.14	Troubleshooting information for installation/deinstallation problems	29.04.2016	AG
1.15	Update for v 3.3.2	19.12.2017	IM
1.16	Update for v.3.3.3	29.12.2017	IM
1.17	Update for v 3.3.4; Added instructions about Windows Installer package	01.03.2018	IM
1.18	Update for v.3.3.5	25.10.2018	BW
1.19	Update for v 3.4.0; reference to Amazon Corretto 8 for installation; some content of the FAQ integrated	12.02.2020	IME
1.20	Update for v 3.4.1	07.10.2022	IME
1.21	Update for v 3.4.2	01.06.2023	RKE,IME
1.22	Update for v 3.4.3	26.07.2023	IME
1.23	Update for v 3.4.4	22.02.2024	RKE

Glossary	
MH	MessageHandler
DB	Database
participantId	The sedex address from a participant. It depends on the context if this is the <code>senderId</code> or the <code>recipientId</code> .
senderId	The sedex address from the sender. E.g. 7-4-1
recipientId	The sedex address from the recipient. E.g. 7-4-1
Native Mode nativeApp	Both names have the same meaning. A <code>Native Mode</code> is represented in the <code>config.xml</code> as <code>nativeApp</code> .
Transparent Mode transparentApp	Both names have the same meaning. A <code>Transparent Mode</code> is represented in the <code>config.xml</code> as <code>transparentApp</code> .
Mixed Mode	Is represented in the <code>config.xml</code> at least with one <code>nativeApp</code> and one <code>transparentApp</code>



Table of contents

1	Introduction	8
1.1	Who Should Read This Document	8
1.2	MessageHandler version and licensing	8
2	Overview.....	8
2.1	Basic principles	8
2.2	Understanding sedex.....	8
2.2.1	Message types in sedex	8
2.2.2	Envelope File	8
2.3	Modes of operation	9
2.4	Transparent Mode	10
2.5	Native Mode.....	10
2.5.1	Overview	10
2.5.2	Addressing participants.....	12
2.5.3	Preparation of Transmission	12
2.5.4	Signing.....	13
2.5.5	Decryption.....	15
2.6	Local Recipient.....	15
2.7	Protocol.....	17
2.7.1	Per Message Protocols.....	17
2.7.2	Global Protocol	18
3	Installation	19
3.1	Packages	19
3.2	Prerequisites.....	19
3.3	Installation using MSI on a Microsoft Windows operating system	19
3.3.1	Variants	19
3.3.2	Windows Service	19
3.3.3	Removal.....	20
3.3.4	Troubleshooting	20
3.4	Manual installation	21
3.4.1	Prerequisites.....	21
3.4.2	Installing MessageHandler	21
3.4.3	Preparatory Steps	22
3.4.4	Installing the MessageHandler Service.....	23
3.4.5	Configuring the Service Wrapper.....	23
4	Configuration	24
4.1	Overview	24
4.2	Toplevel structure of config.xml	25
4.3	Sedex Adapter definitions in config.xml.....	26

4.4	MessageHandler definitions in config.xml	27
4.4.1	Working directory	28
4.4.2	Checker	28
4.4.3	Web service Interface	29
4.4.4	Status Information Store	29
4.4.5	Local Recipients	30
4.4.6	Protocol	31
4.5	nativeApp definitions in config.xml	32
4.5.1	NativeApp PDF Signing	33
4.5.2	Decrypting an incoming zip file	34
4.6	TransparentApp definitions in config.xml	35
4.7	Configure Logging	36
4.8	Configure the Global Protocol	36
4.9	Addressing	36
4.9.1	Resolve the recipientId when sending a message	36
4.9.2	Summary of scripting resolvers	39
5	Webservice Interface	40
5.1	Ping	40
5.1.1	Introduction	40
5.1.2	Request	40
5.1.3	Response	40
5.1.4	Examples	41
5.2	Monitor	43
5.2.1	Introduction	43
5.2.2	Request	43
5.2.3	Response	43
5.2.4	Examples	44
5.3	Trigger	46
5.3.1	Introduction	46
5.3.2	Request	46
5.3.3	Response	46
5.3.4	Examples	47
6	Change Log	48
6.1	What is new in version 3.4.4	48
6.2	What is new in version 3.4.3	48
6.3	What is new in version 3.4.2	48
6.4	What is new in version 3.4.1	48
6.5	What is new in version 3.4.0	48
6.6	What's new in version 3.3.5	49
6.7	What's new in version 3.3.4	49



6.8	What's new in version 3.3.3	49
6.9	What's new in version 3.3.2	49
6.10	What's new in version 3.3.1	50
6.11	What's new in version 3.3.0	50
6.12	What's new in version 3.2.0	50
6.13	What's new in version 3.1.6	50
6.14	What's new in version 3.1.5	50
6.15	What's new in version 3.1.4	50
6.16	What's new in version 3.1.3	50
6.17	What's new in version 3.1.2	50
6.18	What's new in version 3.1.1	51
6.19	What's new in version 3.1	51
6.20	What's new in version 3.0.1	51
6.21	What's new in version 3.0	51
7	References	51

1 Introduction

1.1 Who Should Read This Document

This document addresses programmers and integration engineers who use MessageHandler to connect their applications to a community network relying on sedex, like eSchKG. This document is a technical guide to installing, configuring and managing the MessageHandler V 3.x software package.

This document addresses the installation and operation of MessageHandler. It does not explain the installation and operation of the sedex client. For instruction on how to install the sedex client see the official sedex Manual ("sedex Handbuch") [1].

The administrative tasks related to the sedex client, like getting a sedex-ID and the associated digital certificates, are specific to each community network, and are thus not discussed in the document.

1.2 MessageHandler version and licensing

This manual is adapted to the specification of MessageHandler, version 3.x and above. MessageHandler is distributed freely under the GNU public license (GPL) [2].

2 Overview

2.1 Basic principles

MessageHandler (MH) mediates between the sedex transport infrastructure and one or more applications (see Figure 1). Messages received through the sedex transport infrastructure are dispatched by MH to the appropriate application; messages created by an application destined to a remote receiver are handed over to the sedex transport infrastructure for delivery.

2.2 Understanding sedex

In order to understand what MH can do for you, some basic understanding of the sedex client's functionality is required. This section aims at addressing the some important concepts as far as the interaction between the Sedex client and MH is concerned. Note that it is not a replacement for the sedex user manual.

2.2.1 Message types in sedex

Message types allow for a separation of communities. The message types are managed and assigned to the participants of a community by the Federal Office of Statistics.

Each `outbox` folder can be assigned a specific message type specified in the configuration options of MessageHandler. Alternatively to setting the message type option of an outbox folder in the configuration file, a script doing the job can be referenced.

On the recipient's end, any number of `inbox` folders may be configured, each assigned one or more message types. MessageHandler checks the `inbox` of the sedex Client for documents of a known message type and moves them forward to the corresponding `inbox` folder of MessageHandler, while disregarding documents and envelopes of unknown types.

2.2.2 Envelope File

In order to transmit a message, the sedex Client expects to find a so-called envelope, a file containing the message type, message ID (`messageId`), and the sedex address from the recipient (`recipientId`) and sender (`senderId`).

envl_N.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<envelope xmlns="http://www.ech.ch/xmlns/eCH-0090/1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ech.ch/xmlns/eCH-0090/1 eCH-0090-1-0.xsd"
version="1.0">
  <messageId>a4afd149-8e4e-4bd0-acd3-ab28152f433c</messageId>
  <messageType>10003</messageType>
  <messageClass>0</messageClass>
  <senderId>T7-4-2</senderId>
  <recipientId>T7-4-1</recipientId>
  <eventDate>2012-10-16T09:42:01.510+02:00</eventDate>
  <messageDate>2012-10-16T09:42:01.446+02:00</messageDate>
</envelope>
```

The naming convention for the document file and the envelope file is as follows:

document file: data_N.xxx
envelope file: envl_N.xml

where N is a unique string that usually equals the message-ID in the envelope. In `Native Mode` the MessageHandler generates a sedex envelope and uses a random UUID (using the Java class `java.util.UUID`, as the message ID).

2.3 Modes of operation

The MessageHandler supports different modes of operation:

Mode	Description
Native Mode	This mode is the "classic" mode. One sedex Client and one MessageHandler instance serve n applications. Each of the n applications uses the MessageHandler interface. This mode is used for communication in the eSchKG and Commercial Register community networks.
Transparent Mode	One sedex Client and one MessageHandler instance serve n applications. All of the n applications use the sedex interface. The MessageHandler is a dispatcher of sedex messages. This mode is used for communication outside of the eSchKG and Commercial Register community networks, e.g. Registerharmonisierung/ Harmonisation des registres.
Mixed Mode	One sedex Client and one MessageHandler instance serve n applications. Some of the applications use the MessageHandler interface and the other uses the sedex interface.

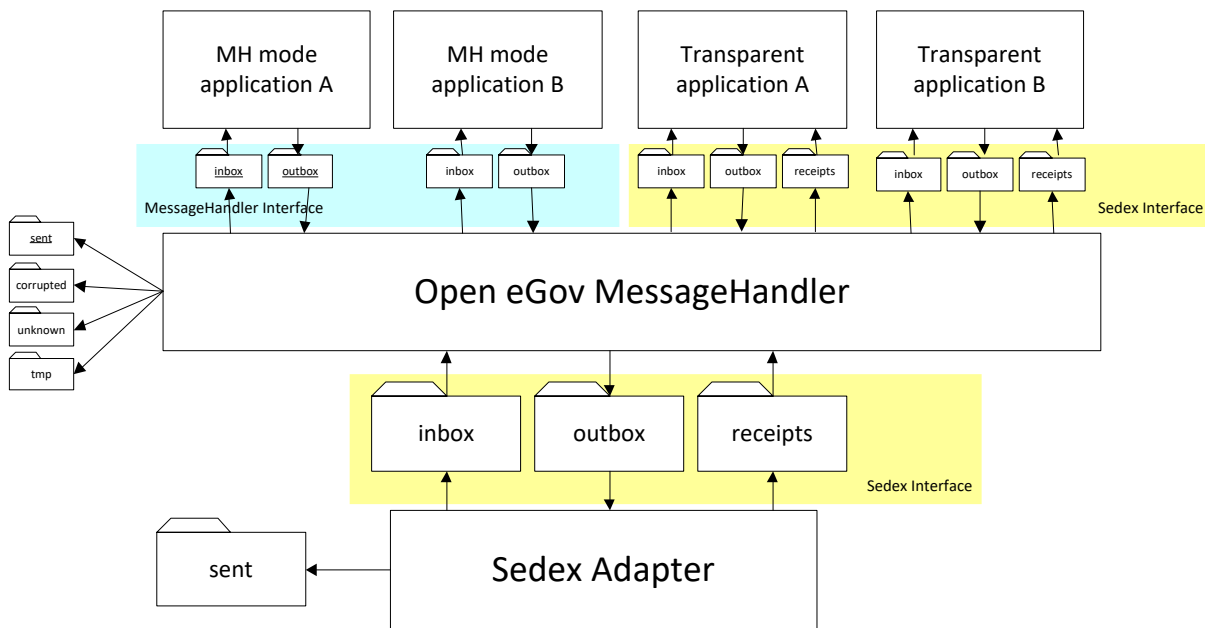


Figure 1 - MessageHandler in mixed mode serving four applications

2.4 Transparent Mode

Applications which are able to cope with the sedex messages themselves can use the MessageHandler in **Transparent Mode**. In this mode the MessageHandler acts as a dispatcher, which connects the `in-` and `outboxes` of each application transparently to the `in-` and `outbox` of a single sedex client.

When sending messages, an application just has to create sedex envelopes and data files in the usual way in an `outbox` owned by the application. The MessageHandler will check this application specific directory periodically, and move the envelopes and data files located there to the `outbox` directory of the sedex Client.

On the receiver-side the MessageHandler can be configured to dispatch messages by inspecting the sedex ID of the receiver or/and the message type of an incoming message. Based on the rules given in the MH configuration, received messages (e.g. envelope and data files) will be moved from the `inbox` of the sedex client to appropriate directory of an application.

The MessageHandler checks the sedex Clients receipt directory periodically. If there are sedex receipts available for a message sent in `Transparent Mode`, the receipt file will be moved to the appropriate directory of an application.

Please note, that in `Transparent Mode` – as opposed to `Native Mode` – data files are neither packed to ZIP archives nor unpacked from ZIP archives.

2.5 Native Mode

2.5.1 Overview

MessageHandler's communication model comprises three hierarchical layers and two interfacing levels between them: `sedex-to-MessageHandler` and `MessageHandler-to-Application`.

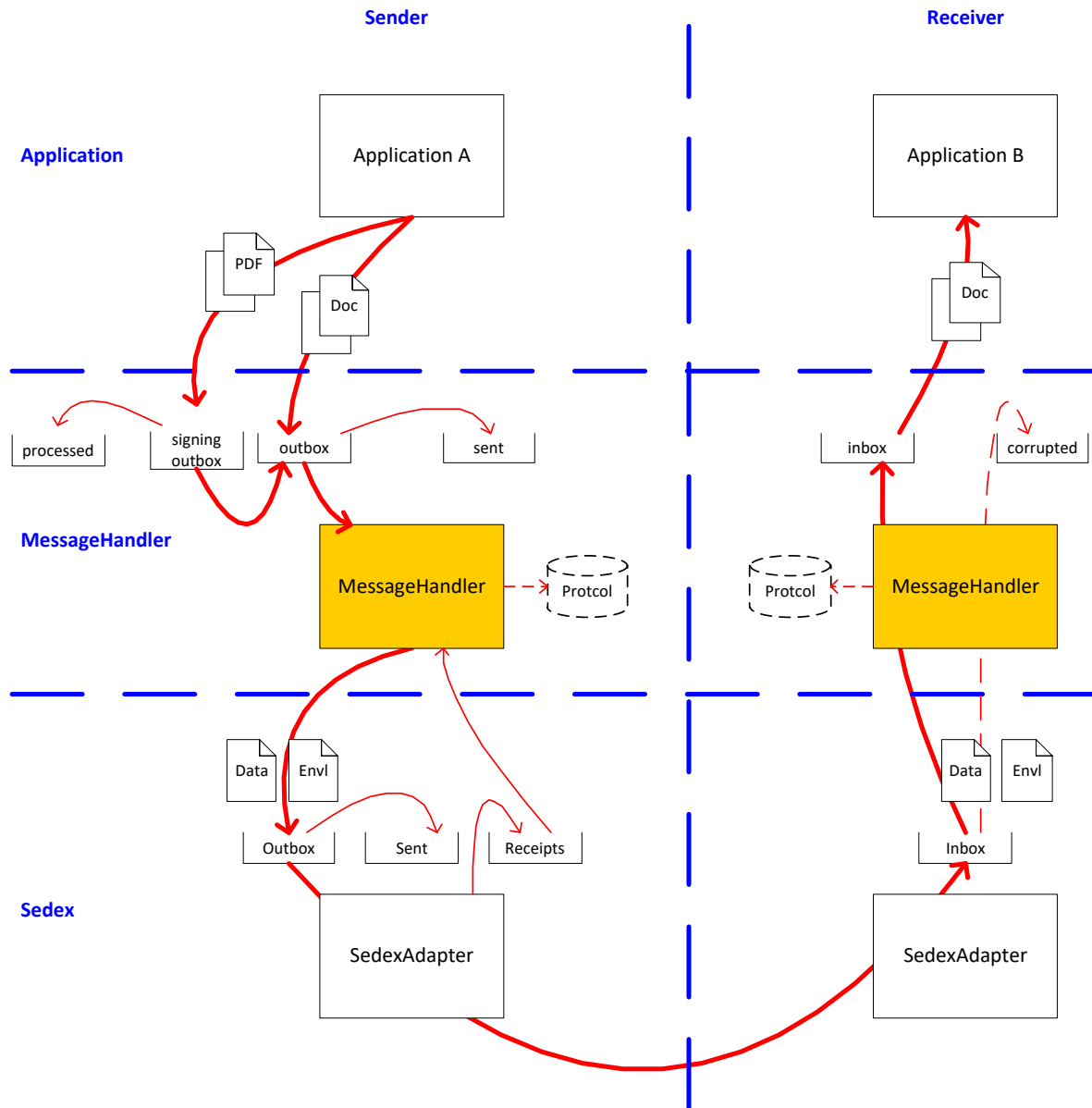


Figure 2 – Native Mode concept

MessageHandler mediates between the application and the sedex transport infrastructure. A community using sedex, including eSchKG, can be operated without the MessageHandler involved. However, using MessageHandler eases application integration dramatically.

Application A sends documents by writing them to one or more local application `outbox` folders from which they are fetched by MessageHandler on a regular basis. MessageHandler determines the recipient of the document(s) using the sedex member identifier, and the message type(s) involved. It then puts all documents with a common message type that are addressed to a specific recipient into one ZIP file, and produces the corresponding sedex envelope. MessageHandler then copies the ZIP file and the envelope into its local sedex `outbox` folder. After sending, the files are moved to the `sent` folder along with the protocol file. The name of it is the name of the document extended by ".prot", e.g. Doc1.xml.prot.

The following is an example record in the protocol file:

```
messageId=ae4864b4-07ae-4e20-b606-4ec6b4d56560
recipientId=2-JU-4
```

```
sent=2008-07-09T16:37:04
delivered=
```

When MessageHandler fails to process a document for whatever reason, the document remains in the `outbox` folder and a new send is attempted sometime later.

The sedex Client periodically checks the `outbox` and sends all documents and envelopes to the appropriate sedex recipients.

The receiving sedex Client will check for authorization to receive the documents and put them into the local `inbox`. MessageHandler checks the local sedex Client `inbox` for new envelopes of the appropriate message type on a regular basis. It then unzips the corresponding ZIP files and puts all documents into the `inbox` folder that matches the message type. Note that sedex Client may be configured to support many different message types – MessageHandler dispatches the documents accordingly into distinct folders, ready to be fetched by the applications. In case there is a problem unpacking the ZIP file, MessageHandler moves the file and its envelope to the `corrupted` folder.

On the sender's end, MessageHandler periodically checks the transmission status by consulting the `sent` and `receipts` folder of the sedex Client and writes a protocol record for each document to a log. In addition, another file, either protocol or error, is written to MessageHandler's `sent` folder.

The name of the error file, if there is one, is the name of the document extended by ".err", e.g. Doc1.xml.err.

Example record in error file:

```
messageId=ae4864b4-07ae-4e20-b606-4ec6b4d56560
recipientId=2-JU-4
sent=2008-07-09T16:37:04.138+02:00
errorCode=301
description=Unknown recipient id 2-JU-4
```

2.5.2 Addressing participants

Entities connected to the sedex network, the sedex clients, are identified by the sedex ID.

When sending a document, MessageHandler can be configured to perform a call to a script in order to resolve the sedex ID of the recipient. This allows for high flexibility in the way addressing is done. Applications may encode the addressee of a document either by a document naming convention, a path naming convention, as part of the document itself, or by means of a mapping table in a database.

2.5.3 Preparation of Transmission

Each time MessageHandler is called, it combines all documents addressed to a specific recipient having a specific message type into one common ZIP file for transmission. In the figure below, the documents in the `outbox` folder are packed into three ZIP files. One is addressed to recipient A using message type R, the two others are addressed to recipient B using message type R and S, respectively.

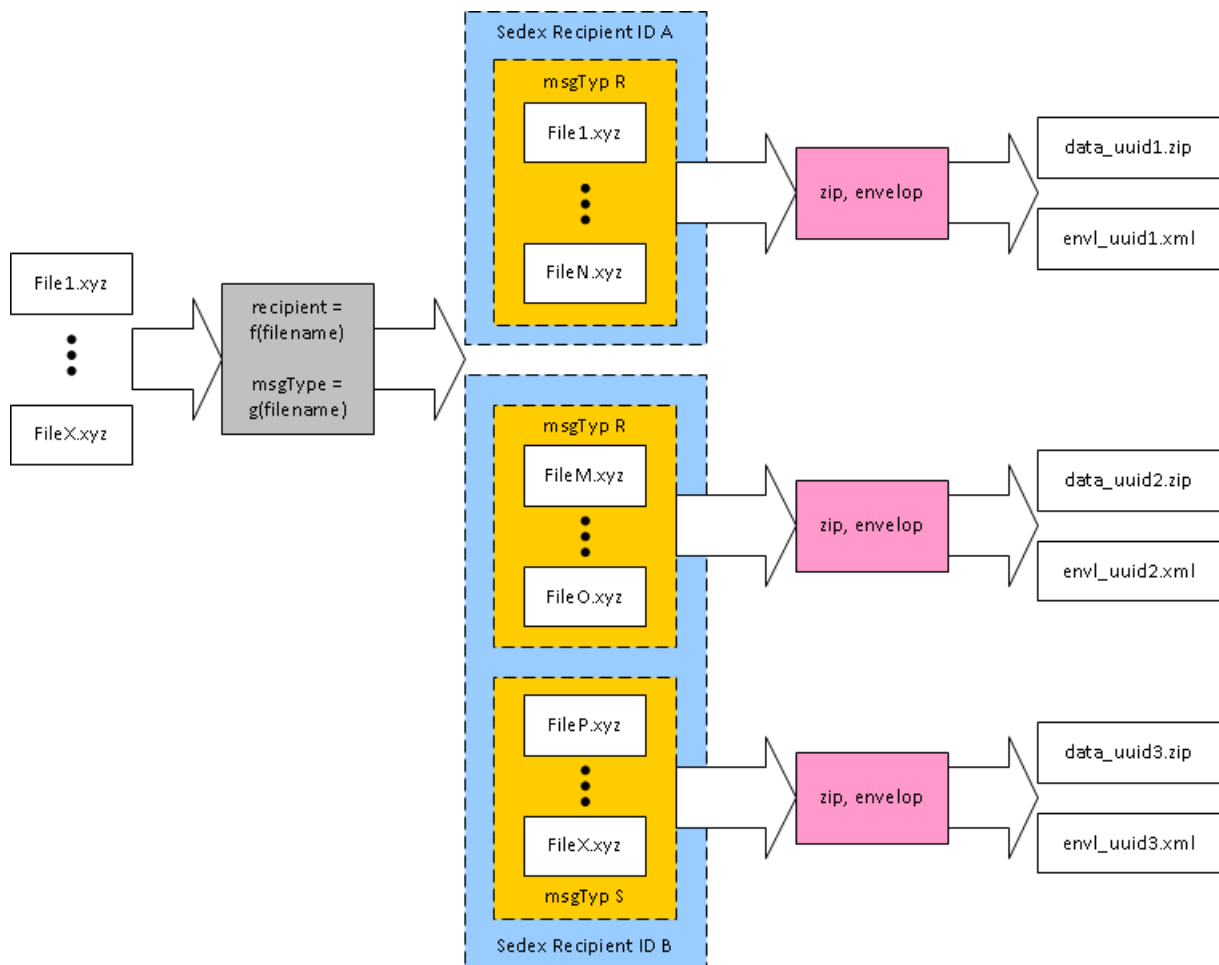


Figure 3 - Preparation of transmission

2.5.4 Signing

Since version 3.0 MessageHandler supports digital signing of out-going PDF files.

Note:

Signing of PDF files is only supported for Native Mode (`nativeApps`)!

Your Java installation **must** have the Unlimited Strength Jurisdiction Policy Files in order to use this feature!!

Out-going PDF files can be digitally signed using a X.509 certificate and a private key stored in a PKCS#12 file [7].

For each `outbox` directory there can be 0...n signing `outboxes` defined. Each signing `outbox` can be configured using its own signature property and PKCS#12 [7] file. If the MessageHandler finds a PDF file inside one of the `signing outbox` directories, the PDF file will be signed.

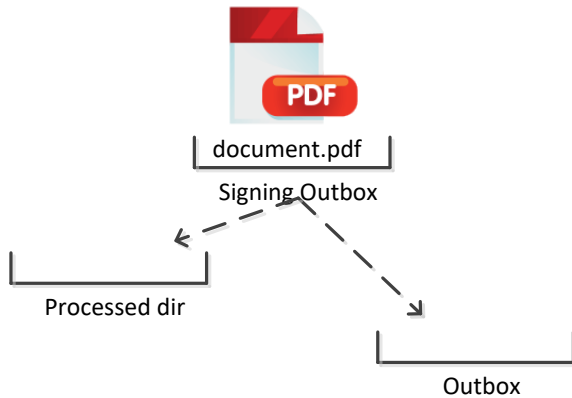
Before processing an `outbox`, the MessageHandler will first look in the associated `signing outboxes` for PDF files to be signed. If a PDF file is found (e.g. `document.pdf`), the file will be signed and the signed file will be placed under a new name (the suffix "-sig" will be added to the files base name, e.g. `document-sig.pdf`) in the `outbox` directory.

The original PDF file will then be either moved to the `processed` (see Figure 4) or deleted, if the (optional) `processed` directory is not configured (see Figure 5).

Note: If the PDF file can't be signed for some reason, the original PDF file will be move into the `corrupted` directory.

After processing all `signing` outboxes the MessageHandler will start sending all files in the `outbox` directory (i.e. including the signed PDF files).

Before signing



After signing

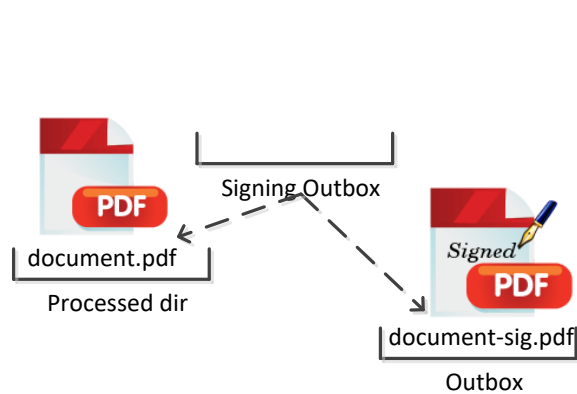
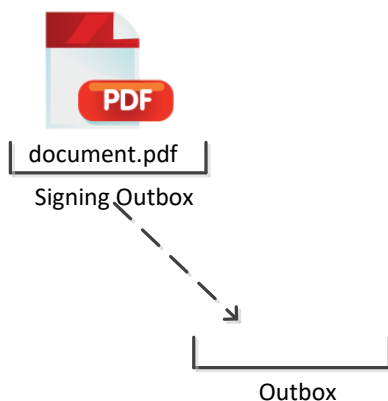


Figure 4 - Signing process – processed directory configured

Before signing



After signing

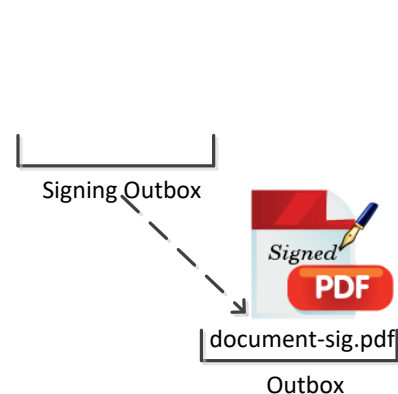


Figure 5 - Signing process – processed directory *not* configured

Below is an excerpt of a MH configuration, which defines:

- A Native Mode eSchKG outbox (located in the directory `<workingDir>/outbox`).
- One single `signing` outbox (located in the directory `<workingDir>/signing-outbox`) for the above outbox.
- The BatchSigner signing profile is located in the file `/mh-config/signature.properties`. The signing profile specifies, where the signature is to be placed, which timestamp provider is to be used, etc.
- The original PDF files are placed into directory `<workingDir>/signingOutboxProcessed` after signing.
- The PKCS#12 file with the certificate to be used for signing is `/mh-config/certificate.p12`.
- The password for the PKCS#12 file.



config.xml snippet

```
<outbox dirPath="outbox" msgType="10301">
  <recipientIdResolver
    filePath="/mh_config/scripts/recipientIdResolver.groovy" method="resolve"
  />
  <signingOutbox dirPath="signing-outbox"
    signingProfilePath="/mh_config/signature.properties">
    <certificate filePath="/mh_config/certificate.p12"
      password="secret"/>
  </signingOutbox>
</outbox>
```

2.5.5 Decryption

Since version 3.2.0 of MessageHandler the decryption of incoming zip files is supported. To be able to decrypt a file a PKCS#12 key store must be configured with the appropriate certificate and the corresponding private key.

The decryption of zip files is configured on the inbox element as follows:

config.xml snippet

```
<nativeApp participantId="T7-4-1">
  <inbox dirPath="application/inbox" msgTypes="10001">
    <decrypt algorithm="AES/CTR/NOPADDING"
      keystore="/mh_config/keystore.p12"
      password="123456" certificateAlias="companysigncert"/>
  </inbox>
</nativeApp>
```

The incoming zip file is a plain zip file that contains the encrypted zip file and the encrypted shared secret (a key). For example:

```
$ unzip -l data_b345c072-6b17-4a3b-914f-01e2b7aff2bf.zip
Archive:  data_b345c072-6b17-4a3b-914f-01e2b7aff2bf.zip
 Length   Date       Time    Name
-----
          0   03-23-2015  09:47   b26dade5-091b-4c98-bcfb-234cfb19a229.zip.enc
          0   03-23-2015  09:44   file.key.enc
-----
          0                               2 files
```

The shared secret key is encrypted with the public key associated with the certificate. Therefore the private key associated with the certificate is used to decrypt the shared secret key (asymmetric encryption).

The plain key is then used to decrypt the encrypted zip file (symmetric encryption).

Note:

MessageHandler provides no functionality to create such zip files; the zip file must be created using a separate application.

Your Java installation **must** have the Unlimited Strength Jurisdiction Policy Files in order to use this feature!!

2.6 Local Recipient

If MessageHandler is used to dispatch messages for multiple application (transparent and/or native) on the same machine, so-called local recipients can be used to deliver messages from one local

application to another local application without sending the messages through the sedex network. (The details of the necessary configuration is described in chapter 4.4.5.)

In the following example we assume that a transparent application (with sedex ID T7-4-2) sends a message another transparent application (with sedex ID T7-4-3) on the same machine.

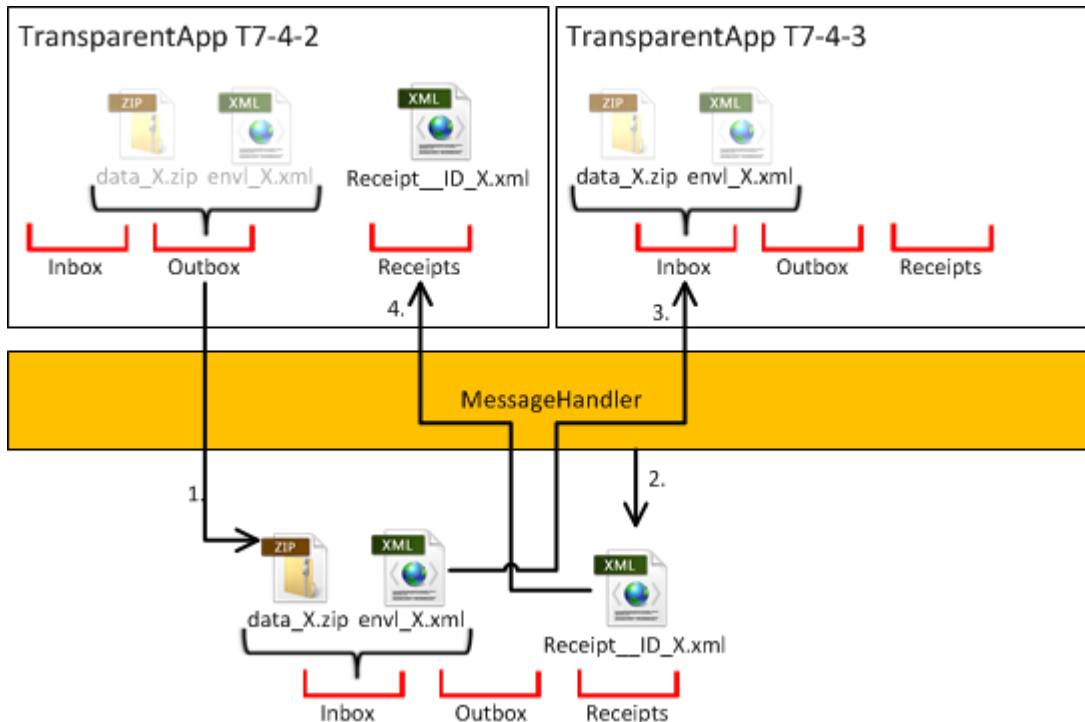


Figure 6 – Transparent Local Recipients

- 1.) MessageHandler recognizes that the message is destined for a local recipient. So the message (envelope and payload) will be placed directly in the sedex inbox directory.
- 2.) Since this is a local delivery, MessageHandler will generate a receipt file. (The receipt file is normally generated by the sedex client, if a message has been successfully delivered to the recipient.)
- 3.) Now the normal MessageHandler delivery process starts: the MessageHandler delivers the message to the inbox of the receiving application.
- 4.) And MessageHandler delivers the corresponding receipt to the sending application.

In the following example we assume that a native application (with sedex ID T7-4-4) sends a message another native application (with sedex ID T7-4-5) on the same machine.

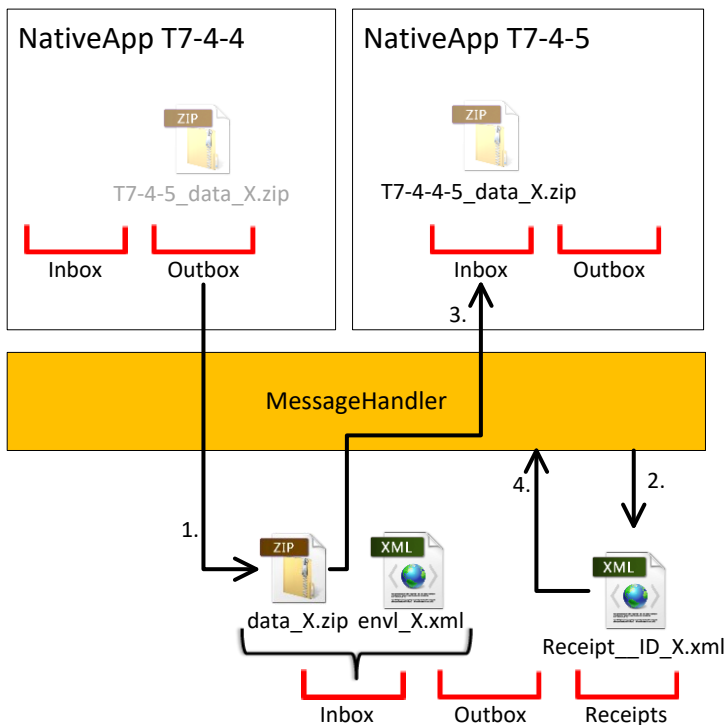


Figure 7 – Native Local Recipients

- 1.) MessageHandler recognizes that the message is destined for a local recipient. The message and the sedex envelope created by Messagehandler will be placed directly in the sedex inbox directory.
- 2.) Since this is a local delivery, MessageHandler will generate a receipt file. (The receipt file is normally generated by the sedex client, if a message has been successfully delivered to the recipient.)
- 3.) Now the normal MessageHandler delivery process starts: the MessageHandler delivers the message to the inbox of the receiving application. (The envelope is “consumed” the the MessageHandler.)
- 4.) The MessageHandler consumes the receipt file in order to update its status database.

2.7 Protocol

2.7.1 Per Message Protocols

For messages sent in `Native Mode`, the MessageHandler can be configured (see chap. 4.4.4) to keep a protocol per message it sends. For each message sent a corresponding file with the added suffix `.prot` will be generated and stored together with the message itself in the `sent` folder, e.g.

```
mr_20120722_063019_7.xml
mr_20120722_063019_7.xml.prot
```

The content of the protocol file

Content of the protocol file mr_20120722_063019_7.xml.prot

```
messageId=af223e27-8c29-4400-8837-3e92b4bd2d07
recipientId=3-CH-18
sent=2012-07-22T06:30:30.050+02:00
delivered=2012-07-22T06:36:38.000+02:00
```

2.7.2 Global Protocol

In `Native Mode` as well as in `Transparent Mode`, `MessageHandler` writes a global protocol file. This file summarizes all state transitions of the files sent by the `Messagehandler`.

The following extract shows a part of the global protocol file (note: `<baseDir>` is the corresponding base directory configured in the MH configuration file):

msg-handler.prot snippet

```
filename=/<baseDir>/outbox1/123_Dokument1.txt
messageId=410c4671-e29d-488b-82a2-ed9f72455385
recipientId=T7-4-1
preparing=2008-07-11T11:43:24.500+02:00

filename=/<baseDir>/outbox1/123_Dokument1.txt
messageId=410c4671-e29d-488b-82a2-ed9f72455385
recipientId=T7-4-1
forwarded=2008-07-11T11:43:24.512+02:00

filename=/<baseDir>/outbox1/123_Dokument1.txt
messageId=410c4671-e29d-488b-82a2-ed9f72455385
recipientId=T7-4-1
sent=2008-07-11T11:44:07.095+02:00
```

The global protocol file is configured in the `log4j.properties` file (see chap. 4.8).

3 Installation

This document addresses the installation and operation of MessageHandler. It does not explain the installation and operation of the sedex client. For instruction on how to install the sedex client see the official sedex Manual ("sedex Handbuch") [1].

Note: It is recommended to install and test the sedex client prior to installing MessageHandler.

3.1 Packages

MessageHandler may be downloaded as packages in following formats:

- Windows 32bit (also works on 64bit systems):
 - Zipped binary
 - MSI installer
- Linux 64bit:
 - Zipped binary
- Source code:
 - Zipped maven project

3.2 Prerequisites


In order to operate MessageHandler, the following prerequisites must be met:

- Registration with sedex is done and valid digital certificates have been issued to you by the Federal Office of Statistics;
- The sedex client is correctly installed and tested;
- The MessageHandler account/user (i.e. the system account, under which the MH is being run) and the application account/user using it have been granted appropriate access rights to launch services and scripts and to store and remove documents in the respective directories.

3.3 Installation using MSI on a Microsoft Windows operating system

To simplify the installation on a Microsoft Windows operating system, a Windows installer package (MSI) is provided. It offers a wizard-based installation procedure gathering information needed to install and configure MessageHandler as well as starting the MessageHandler service.

The installer goes through the exact same steps as the installation by hand described in chapter 3.4 *Manual installation* but in a more convenient manner. The result will be equal and subsequent configuration changes may be done using explanations in this chapter.


 the MSI installer package does not support the installation of more than one instance of MessageHandler in the same machine!

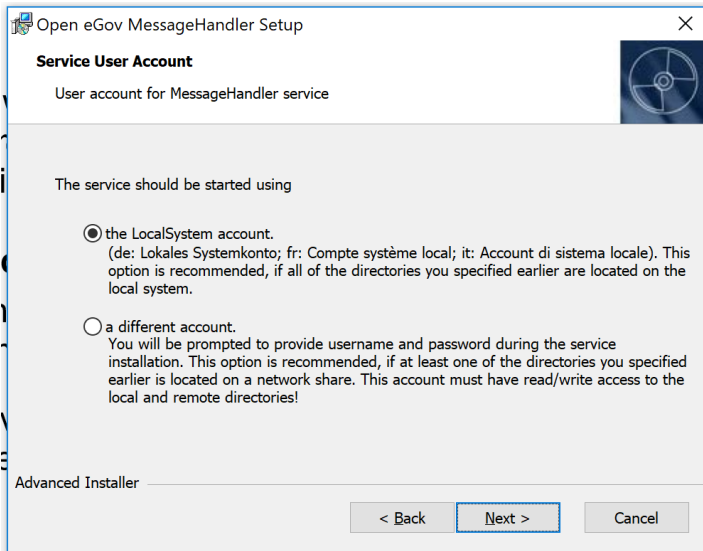
3.3.1 Variants

The installation process will ask you to decide whether you wish to install MessageHandler for an application of the type *creditor* or *collecting office*. If you select the *collecting office* variant, the installed configuration will allow your outgoing messages to be digitally signed.

3.3.2 Windows Service

MessageHandler will run as a windows service. Its name is "Open eGov MessageHandler" and it is configured to start automatically.

 MessageHandler service will fail to start, if the user account used for the MessageHandler service has no access to the directories configured in the installation process. You will be prompted to provide the correct user account during the installation process:



3.3.3 Removal

To uninstall the application re-run the installer or remove it using the *Programs and Features* control panel of Windows.

The uninstallation process does not remove the working directory even if it's within the installation directory, so you don't lose any important files. As soon as you are sure you don't need them anymore, you may delete them manually.

The interface directories of sedex and the eSchKG/e-LP/e-LEF application selected while installing MessageHandler don't get deleted of course.

3.3.4 Troubleshooting

3.3.4.1 Installation fails

If the installation fails because the service could not be started the following message boxes appears.

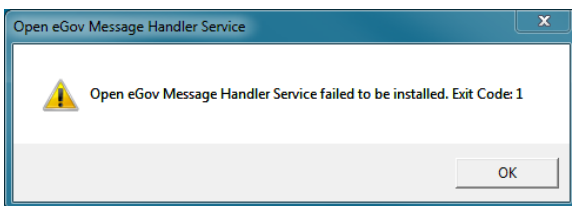


Figure 8: Installation of MessageHandler Service failed

This case indicates that there might be a problem with one of the various configuration files. Ensure that

- the target installation directory for MessageHandler is empty. Delete or rename the existing one or choose another installation directory.
- MessageHandler refuses to install, if you specify directories in the installation wizard, which do not exist! Inspect the installation logfile and search for error messages like

```
MSI (c) (7C:54) [14:32:12:501]: Note: 1: 1314 2: /Program
Files/Sedexclient/interface
MSI (c) (7C:54) [14:32:15:251]: Product: Open eGov MessageHandler -- Error
1314. The specified path '/Program Files/Sedexclient/interface' is
unavailable.
```



3.3.4.2 Removal fails

By running the MSI package on a machine with MessageHandler installed removes the software. As of various reasons the Windows Service may be unable to be deleted. This case is expressed by a similar message box as a failed installation of the service. The removal of the software will not fail even if the service couldn't be stopped or removed.

This problem may have two outcomes. Either the services as well as all locked files will be removed after an ordinary reboot of the computer. If the service is still listed in the services table (Start → Control Panel → search for "Administrative Tools" → Services: Open eGov Message Handler) open a command shell (Start → type "cmd" then press the enter) and run the following command:

```
sc delete MessageHandler
```

This stops and removes the Open eGov Message Handler Service. Afterwards you can delete the remaining files in the installation directory of Open eGov Message Handler when necessary.

If you still can't remove the package, try using the tool Revo Uninstaller¹.

3.4 Manual installation

If you are not running a Microsoft Windows operating or prefer to install MessageHandler by hand, follow the instructions below.

3.4.1 Prerequisites

To operate MessageHandler a Java 8 Update ≥ 232 runtime must be installed.

MessageHandler is developed and tested using the Amazon Corretto 8² variant of the OpenJDK. We recommend using this distribution. You are free to use any of the other OpenJDK distributions like AdoptOpenJDK³ or Azul Zulu⁴.

- On Windows:
only the 32bit version of Java is supported out-of-the box!
To allow the installation of MH as a service (on Windows), the 32-bit version of the Java Service Wrapper by Tanuki Software is bundled in the MH distribution. If you intend to use the 64-bit version of Java on your 64-bit operating system, you must replace the bundled Java Service Wrapper (Windows: wrapper.exe) with the 64-bit version.
Note: The Windows 64-bit version of the Java Service Wrapper is a commercial product (costs approximately 380 €).
- On Linux:
only the 64bit version of Java is supported out-of-the box!
To allow the installation of MH as a daemon (on Linux/Unix) the 64-bit version of the Java Service Wrapper by Tanuki Software is bundled in the MH distribution.

3.4.2 Installing MessageHandler

For installing MessageHandler, download the latest MessageHandler distribution archive [8] for your platform (Windows / Linux). Note that the 32-Bit Windows distribution can be used without problems on 64-Bit Windows (see previous chapter).

¹ https://www.revouninstaller.com/revo_uninstaller_free_download.html

<https://aws.amazon.com/de/corretto/>

³ <https://adoptopenjdk.net/>

⁴ <https://www.azul.com/downloads/zulu-community/>

Extract the distribution into a directory of your choice (<installation-dir>). The following files and directories will be created:

```

<installation-dir>/bin          /InstallMessage-handler-NT.bat
                               /UninstallMessage-handler-NT.bat
                               /message-handler.bat
                               /wrapper.exe
                               /run.bat
<installation-dir>/conf       /config.xml
                               /config.xsd
                               /log4j.properties
                               /recipientIdResolver.groovy
                               /wrapper.conf
<installation-dir>/example    /...
<installation-dir>/lib        /...
<installation-dir>           HANDBUCH_INFO
                               LICENCE
                               README
                               RELEASE-NOTES
  
```

Directory	Content
bin	This directory contains scripts for the MessageHandlers installation and de-installation. There are also start and stop scripts for the service which can run under Windows and Unix systems.
lib	This directory contains all required libraries for the MessageHandler.
conf	This directory contains all necessary configuration files. The most important files are: config.xml and log4j.properties. It contains also some example groovy scripts.
example	Example directory layout referenced by the config.xml installed in manual installation.

3.4.3 Preparatory Steps

Prior to starting MessageHandler, a working directory, `workingDir`, must be created. The working directory is the directory where MH keeps files which it needs for its internal workings ("the guts"). In that directory create the following sub-directories:

- `<workingDir>/sent`
- `<workingDir>/corrupted`
- `<workingDir>/tmp/receiving`
- `<workingDir>/tmp/preparing`
- `<workingDir>/unknown`

Important note: These directories need to be named **exactly** like this and they must be subdirectories of `<workingDir>/!`

Next, create a base directory (`baseDir`). Directories and files underneath the base directory can be referenced using a relative path in the MH configuration file.

Important note: The name of the base directory and the directories underneath may be arbitrarily chosen (directories don't even have to be located underneath `<baseDir>`) but must exactly match those declared inside the `nativeApp` or `transparentApp` definitions in `config.xml`.



Next create a directory `log`, where the log files of the MessageHandler will be placed.

3.4.4 Installing the MessageHandler Service

Change the working directory to the distribution installation `<installation-dir>`. Install the services as follows:

- Windows (as an NT service): `bin/InstallMessage-handler-NT.bat`
- Linux (as a Daemon): `bin/message-handler start`

To uninstall:

- Windows (as an NT service): `bin/UninstallMessage-handler-NT.bat`
- Linux (as a Daemon): `bin/message-handler stop`

Note: You must grant the appropriate execution rights to the files in the `/bin` directory.

Note: You can start MessageHandler without a running a sedex instance. So, you'll be able to test the MessageHandler in a safe environment. For example, put a file in an `outbox` directory and watch if the file will be packed in a `data_XYZ.zip` and `envl_XYZ.xml` file and placed in the `sedex outbox` directory.

3.4.5 Configuring the Service Wrapper

MessageHandler is a Java program that uses a virtual machine controlled by a service wrapper. The execution parameters of the wrapper are specified in the wrapper's configuration file `wrapper.conf` in the `/conf` subdirectory of your installation.

By default, the first JRE on the system path is used, make sure this is the one you intended to use!

See [4] for a detailed description of `wrapper.conf`.

4 Configuration

Note

The online supplement to this document located at <http://www.openegov.ch/messagehandler> shows concrete configuration examples.

4.1 Overview

Configuration of MessageHandler is done via the file `config.xml`, the global configuration file. The configuration options include:

- directory and path definitions;
- definition of one or more `outbox` folders;
- definition of optional multiple `signing outbox folder per outbox folder`;
- definition of one or more `inbox` folders;
- activity schedule for the sender, checker and receiver processes;
- resolution of destination addresses and definition of message types;
- remote management over HTTP;
- configuration of the internal database for logging and status information storage;
- location of the sedex clients directories

The structure of the configuration file `config.xml` is specified in the corresponding XML Schema file `config.xsd` (located in `<installation-dir>/conf`). At startup MessageHandler validates the `config.xml` against the XML Schema file `config.xsd`. MessageHandler will not start unless the validation succeeds.

A typical MessageHandler configuration is structured like follows

Typical config.xml	
Config heading	<pre><?xml version="1.0" encoding="UTF-8"?> <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://msghandler.suis.admin.ch/xmlns/config config.xsd" xmlns="http://msghandler.suis.admin.ch/xmlns/config" version="3.0"></pre>
Sedex Client definitions	<pre><sedexAdapter> <participantId>T7-4-1</participantId> <inboxDir>/mh_examples/case3/sedex/inbox</inboxDir> <outboxDir>/mh_examples/case3/sedex/outbox</outboxDir> <receiptDir>/mh_examples/case3/sedex/receipts</receiptDir> <sentDir>/mh_examples/case3/sedex/sent</sentDir> </sedexAdapter></pre>
MessageHandler definitions	<pre><messageHandler> <!-- In diesem Pfad müssen die MH Basis-Verzeichnisse sein: corrupted, tmp, unkown, etc. --> <workingDir dirPath="/mh_examples/case3/mh/working-dir"/> <!-- Wo starten relative Pfade --> <baseDir dirPath="/mh_examples/case3"/> <!-- wie oft wird die sedex inbox geprft --> <sedexInboxDirCheck cron="0/30 * * * * ?" /> <!-- wie oft wird das sedex receipts directoty geprft --> <sedexReceiptDirCheck cron="0/30 * * * * ?" /> <!-- jede *outbox kann den Wert Überschreiben --> <defaultOutboxCheck cron="0/30 * * * * ?" /> <webserviceInterface host="localhost" port="18080"/> <statusDatabase dirPath="/mh_examples/case3/mh/working-dir/db" dataHoldTimeInDays="2" resend="true"/> </messageHandler></pre>



nativeApp definitions	<pre> <nativeApp participantId="T7-4-1" > <outbox dirPath="applicationA/outbox" msgType="10301"> <recipientIdResolver filePath="/mh_examples/case3/mh/install-dir/conf/recipientIdResolver.groovy" method="resolve" /> </outbox> <inbox dirPath="applicationA/inbox" msgTypes="10301"/> </nativeApp> </pre>
transparentApp definitions	<pre> <transparentApp participantId="T7-4-1" > <outbox dirPath="applicationB/outbox" /> <inbox dirPath="applicationB/inbox" msgTypes="112"/> <receipts dirPath="applicationB/receipts" msgTypes="112"/> </transparentApp> </config> </pre>

4.2 Toplevel structure of config.xml

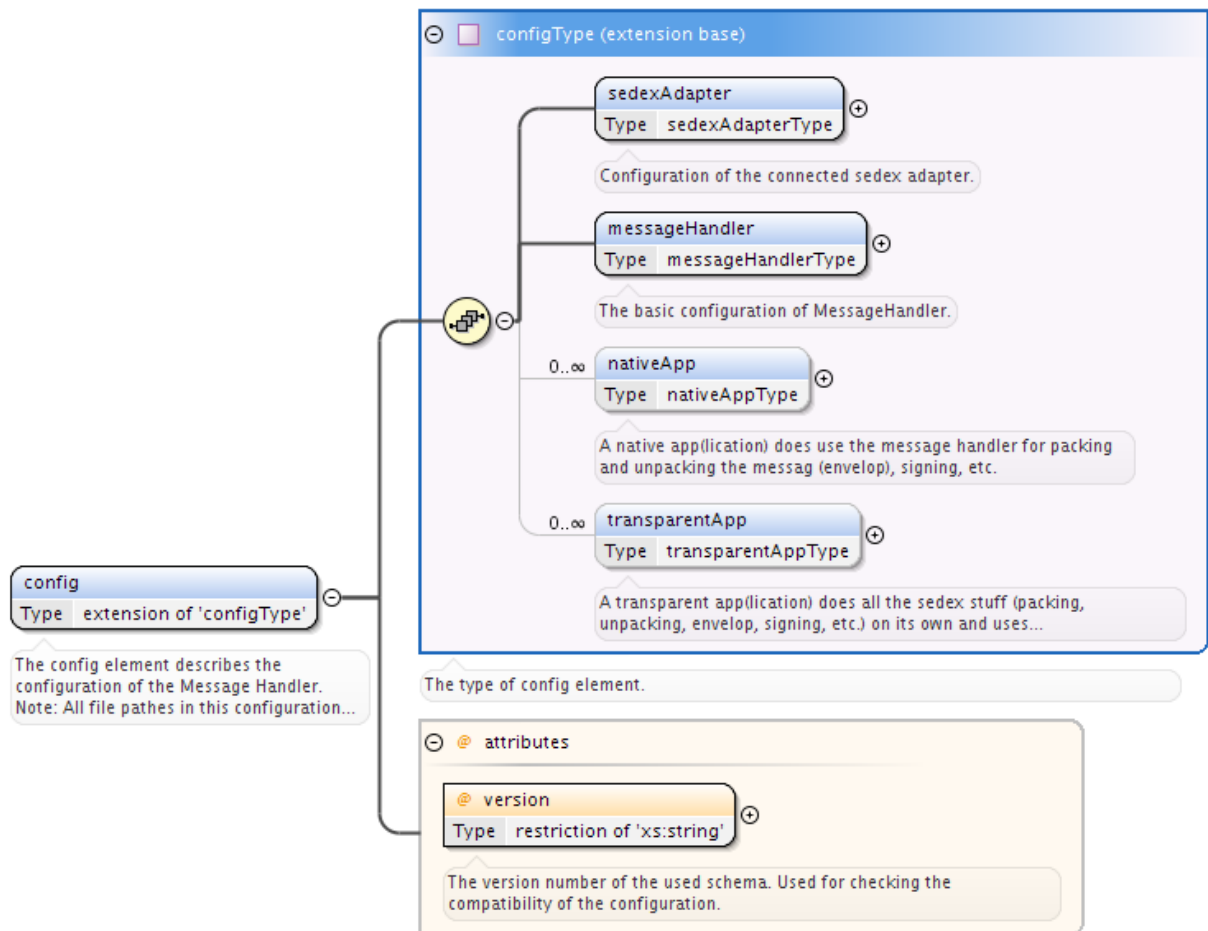


Figure 9 – Toplevel Structure of config.xml

The toplevel structure of config.xml requires at least a `sedexAdapter` and a `messageHandler` tag. There can be optional 0..n `nativeApp` and `transparentApp` tags.

4.3 Sedex Adapter definitions in config.xml

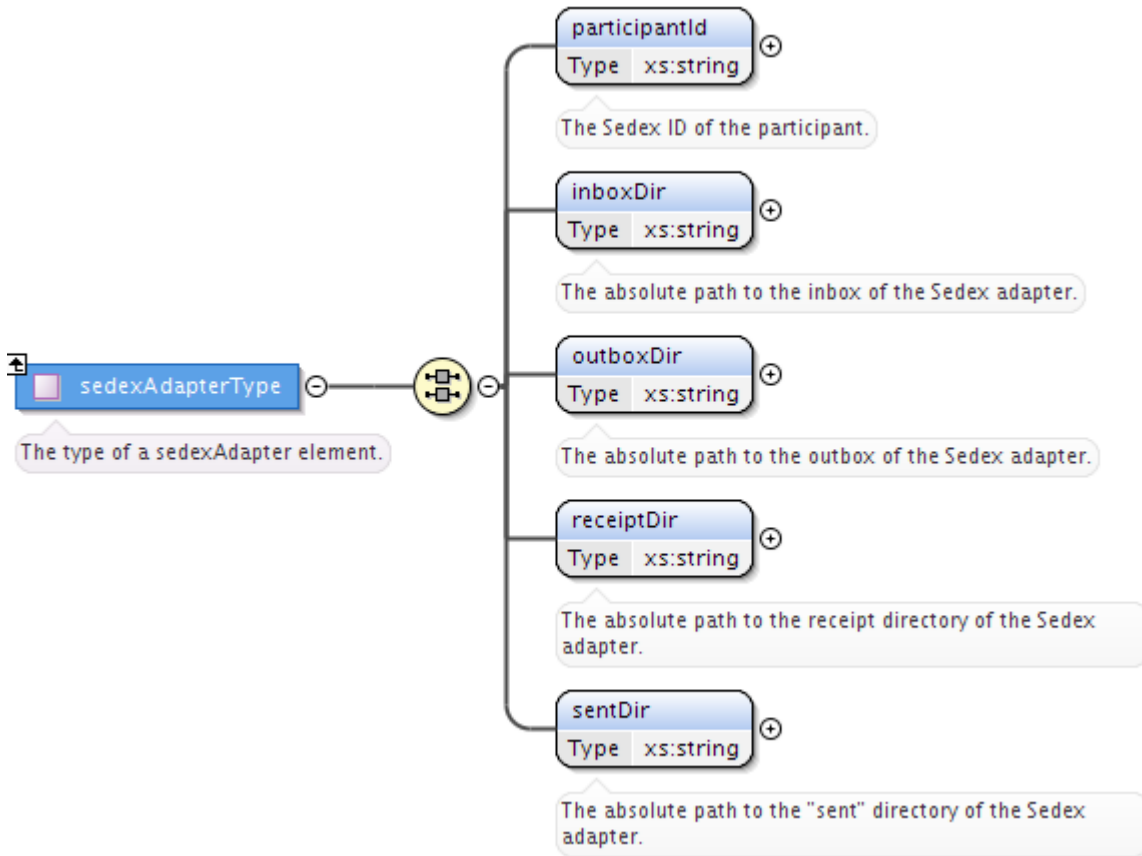


Figure 10 - 4.3 Sedex Adapter definitions in config.xml

The `sedexAdapter` tag specifies the `inbox` and `outbox` directories that are used to move messages back and forth between the sedex Client and MessageHandler.

In the example below we assume, that the sedex ID of the client is T7-4-1, and that all relevant sedex directories are located underneath the directory `c:\sedexAdapter_T7-4-1`:

```

config.xml snippet
<sedexAdapter>
  <participantId>T7-4-1</participantId>
  <inboxDir>c:/sedexAdapter_T7-4-1/inbox</inboxDir>
  <outboxDir>c:/sedexAdapter_T7-4-1/outbox</outboxDir>
  <receiptDir>c:/sedexAdapter_T7-4-1/receipts</receiptDir>
  <sentDir>c:/sedexAdapter_T7-4-1/sent</sentDir>
</sedexAdapter>
  
```

Note: You can copy these elements from your sedex configuration file. This part from the configuration is equivalent to the sedex configuration.

4.4 MessageHandler definitions in config.xml

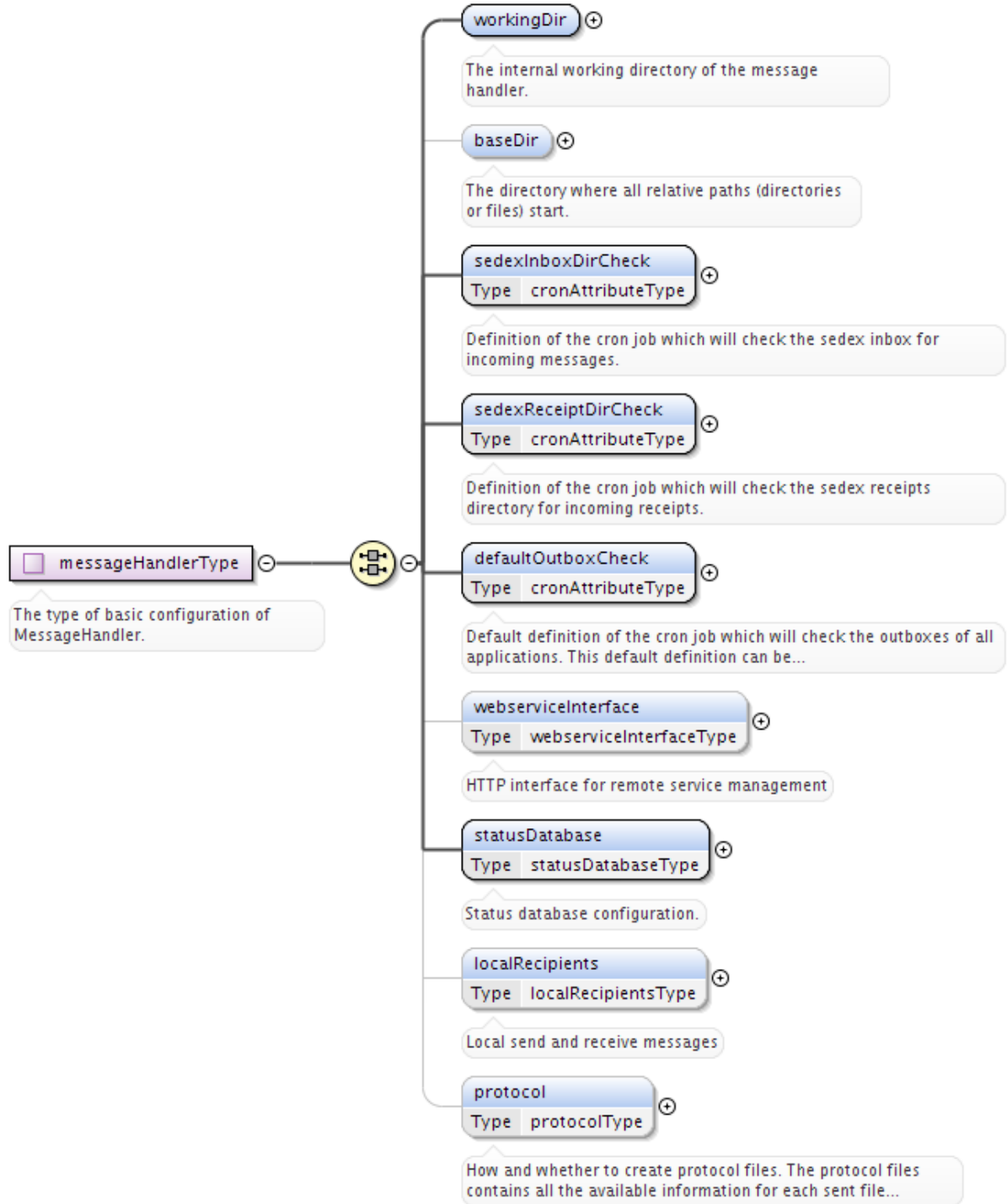


Figure 11 MessageHandler definitions in config.xml

The `messageHandler` tag contains global settings for the MessageHandler. Some of these settings can be overwritten in a specialized context. For example the `cron` attributes.

XML Tag	Description
<code>workingDir</code>	See chapter: 4.4.1 Working directory

XML Tag	Description
baseDir	The <code>baseDir</code> defines a base directory path. This gives the possibility to use relative paths in the <code>nativeApp</code> and <code>transparentApp</code> section. The relative paths will be combined with the <code>baseDir</code> . It's a good idea to place your <code>nativeApps</code> and <code>transparentApps</code> inboxes and outboxes in the <code>baseDir</code> .
sedexInboxDirCheck	Checks the sedex inbox directory. Check time is defined with cron job syntax. See chapter: 4.4.2 Checker for an example.
sedexReceiptDirCheck	Checks the sedex receipt directory. Check time is defined with cron job syntax. See chapter: 4.4.2 Checker for an example.
defaultOutboxCheck	Checks the configured <code>nativeApp</code> and <code>transparentApp</code> outbox directories. Check time is defined with cron job syntax. See chapter: 4.4.2 Checker for an example.
webserviceInterface	The web service interface publishes information about the state of the MessageHandler and its sent messages. See chapter: 4.4.3 Web service Interface.
statusDatabase	See chapter: 4.4.4 Status Information Store.
localRecipients	Used to send messages to recipients which are on the same machine. See chapter 4.4.5 Local Recipients.
protocol	See chapter: 0 Protocol.

4.4.1 Working directory

config.xml snippet

```
<workingDir dirPath="/mh_examples/case3/mh/working-dir"/>
```

MH requires a given directory structure underneath its `workingDir` directory. The following table shows the directories of the required structure and their purpose:

Directory	Purpose
sent	The sent messages will be placed in this directory.
corrupted	This directory contains <ul style="list-style-type: none"> a) corrupted incoming messages. A corrupt message is a file which couldn't be unpacked for some reason. b) PDF files, which could not be signed
unknown	This directory contains messages which couldn't be dispatched to an application.
tmp/preparing	Messages which will be sent will be prepared in this directory.
tmp/receiving	In this directory received messages will be unpacked

4.4.2 Checker

config.xml snippet

```
<sedexInboxDirCheck cron="0/30 * * * * ?" />
<sedexReceiptDirCheck cron="0/30 * * * * ?" />
<defaultOutboxCheck cron="0/30 * * * * ?" />
```



In the config.xml snippet above all checkers will run each 30 seconds.

Note: See [5] for more information on how to define cron-jobs.

4.4.3 Web service Interface

In order to use the MessageHandler HTTP web service interface, you need to configure a host and a TCP-port as follows:

config.xml snippet

```
<webserviceInterface host="localhost" port="18080"/>
```

The `host` attribute defines the host through which the web service is to be accessed. Which the definition above, the MessageHandler web service can only be accessed locally (i.e. from the machine where the MessageHandler is running).

See chapter 5 for the supported operations.

Note:

- When using the interface from a remote computer, a fully qualified hostname or IP address shall be defined in the `host` attribute.
- A free TCP/IP port on the machine has to be chosen.
- The TCP/IP port numbers below 1024 are special in that normal users are not allowed to run servers on them.

4.4.4 Status Information Store

MessageHandler keeps track of all messages that it has sent in a local (embedded) database. The location and the behavior of the database are specified using the `statusDatabase` tag.

config.xml snippet

```
<statusDatabase dirPath="/mh_examples/casel/mh/working-dir/db" dataHoldTimeInDays="2" resend="false"/>
```

Attribute	Required	Description
dirPath	Yes	The path where to store the database files. An absolute path is required.
dataHoldTimeInDays	Yes	Number of days until all status information about a message sent is deleted. This includes: <ul style="list-style-type: none"> - entries concerning the message in the status database. - Sedex receipts kept in the directory referenced by the config value <code>/config/sedexAdapter/receiptDir</code> <p>Note: setting <code>dataHoldTimeInDays</code> to 0 is not recommended, as this will remove the datasets the same day. For installation with high traffic a value between 2 and 10 for <code>dataHoldTimeInDays</code> is recommended.</p>
resend	Yes	If <code>resend</code> is false, the MessageHandler will refuse to send messages with equal file names to the same destination more than once. The decision is based on the living datasets in the database. After <code>dataHoldTimeInDays</code> <code>resend</code> will be allowed. <p>Note: this applies only to <code>Native Mode</code>.</p>

In the example above, the database will be stored underneath the directory `/mh_examples/casel/mh/working-dir/db`. Records in the database are deleted after 30 days. With `resend` set to `false`, you will be sure not to resend the same file (or a document that has the same file name) within 30 days.

A database dump, `dump.csv`, is produced with every run of the service and stored in the database directory. The CSV format is:

```
<ParticipantId>, <File>, <MessageId>, <Sent>, <Received>, <StatusCode>
```

StatusCode	Description
1	Sending in progress
2	Sent
3	Forwarded to the sedex client
4	Delivered
5	Expired
8	Error

Example:

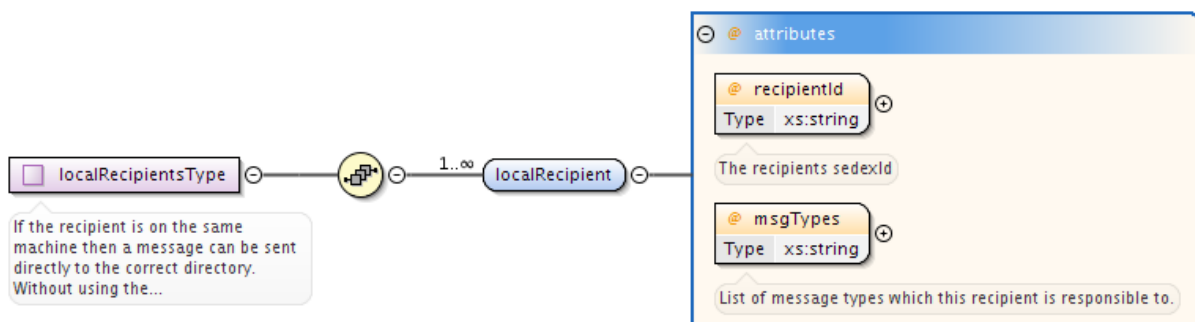
```
T7-4-1, myDoc.txt, 410c4671-e29d-488b-82a2-ed9f72455385, 11.07.08 11:43,
11.07.08 11:45, 4
```

Note: It is not recommended to use the file `dump.csv` to track the status of messages, as this file will be removed in a future version of MessageHandler! The HTTP Monitor Interface (See 5.2 Monitor) provides access to the status information stored in the internal DB.

4.4.5 Local Recipients

So-called local recipients can be used in order to deliver messages from one local application to another local application without sending the messages through the sedex network. Transparent as well as native application can be defined as local recipients.

MessageHandler will place a message sent from an application to a local recipient in the sedex inbox (instead of the outbox). On the next delivery cycle such a message will be delivered to the local recipient. This mechanism works for native as well as transparent applications, and replaces the “targetDirectoryResolver” mechanism introduced in MessageHandler 2.2.x



config.xml snippet

```
<localRecipients>
  <localRecipient recipientId="T7-4-1" msgTypes="10301"/>
  <localRecipient recipientId="T7-4-2" msgTypes="10301 112"/>
</localRecipients>
```



With the configuration above MessageHandler will place an outgoing message with msgType "10301" for the recipient "T7-4-1" into the inbox directory of an application (native or transparent), which is configured to receive messages of type "10301" for the recipient "T7-4-1". If no such recipient has been configured, an ERROR log entry will be produced.

4.4.6 Protocol

config.xml snippet

```
<protocol createPerMessageProtocols="false"/>
```

This is an optional tag. Default value for the attribute `createPerMessageProtocols` is `false`. If set to `true`, a per message protocol will be written (see chap. 2.7.1). For each message sent with a `nativeApp` a corresponding `.prot` file will be generated and stored in the `sent` folder.

Note: the global protocol (see chap. 2.7.2) is **not** controlled using this attribute. See chap. 4.8 for enabling/disabling the global protocol.

4.5 nativeApp definitions in config.xml

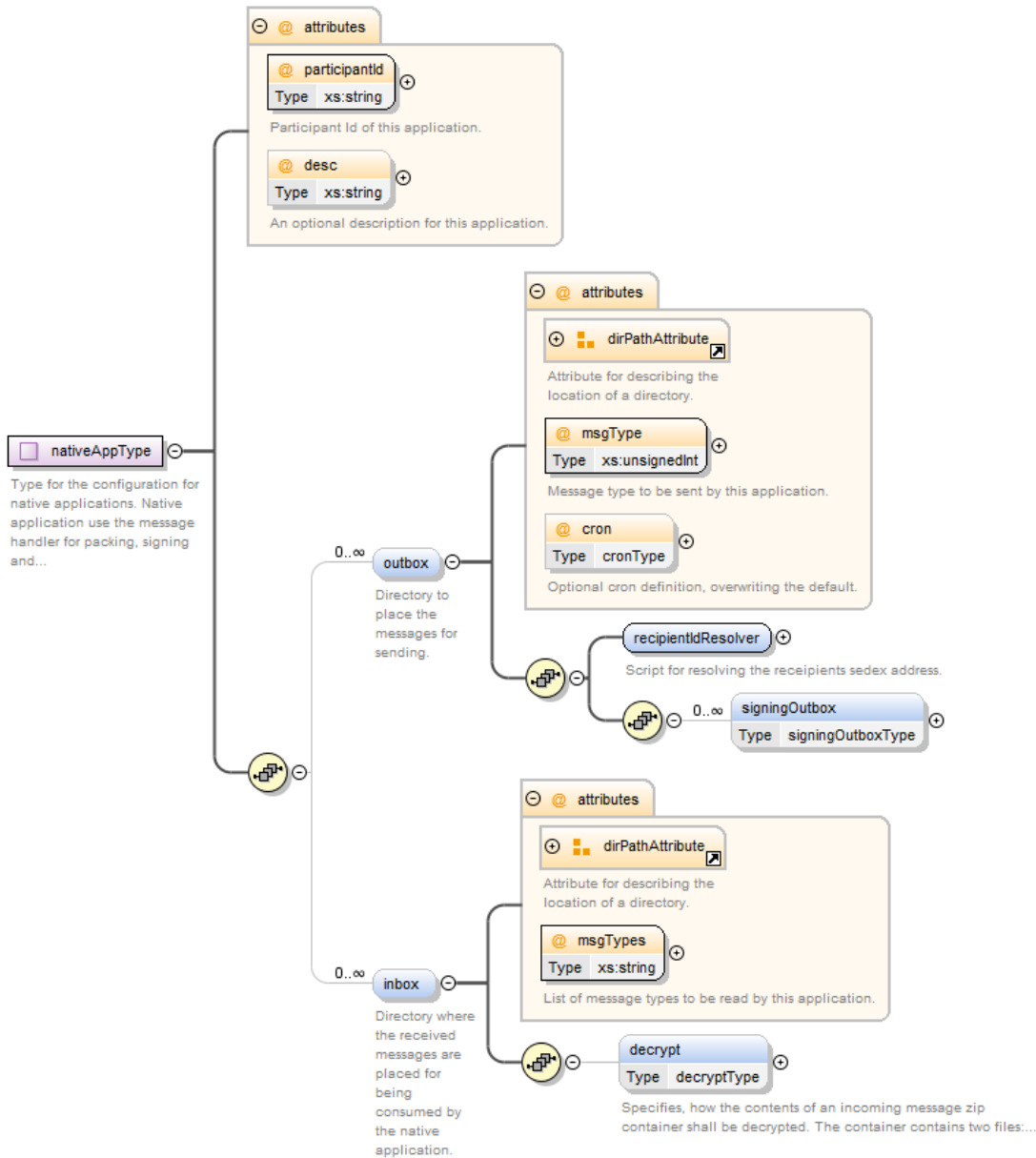


Figure 12 4.5 nativeApp definitions in config.xml

A nativeApp can have 0...n outboxes and 0...n inboxes. A nativeApp is able to work with exact one participantId.

An outbox requires exact one msgType. An inbox is able to receive multiple msgTypes.

Note: It's not allowed that two or more outboxes have the same msgType. And it's also not allowed that two or more inboxes have the same msgTypes. In the context of the same nativeApp.

Each nativeApp requires a recipientIdResolver script. This script is responsible to determine the receiver (recipientId). For details see: 4.9 Addressing.

It's possible to set an other cron value for the outbox. If this optional attribute is set it will override the global cron value defaultOutboxCheck for this nativeApp.

config.xml snippet


```
<nativeApp participantId="T7-4-1" >
  <outbox dirPath="application/outbox" msgType="10301">
    <recipientIdResolver
      filePath="/resolvers/recipientIdResolver.groovy" method="resolve" />
    </outbox>
    <inbox dirPath="application/inbox" msgTypes="10301"/>
  </nativeApp>
```

4.5.1 NativeApp PDF Signing

As described in chapter “2.5.4 Signing” a `NativeApp` is able to sign PDF files before sending them to a recipient. MessageHandler supports signing using certificates and private keys stored in PKCS#12 [7] files.

Prerequisite:

- A PKCS#12 file and the appropriate password
- Signature.properties file.
- The Unlimited Strength Jurisdiction Policy Files are installed in your Java installation

Note: The package `open-egov-msghandler-3.x.y-bin.[tar.gz, zip]` contains demonstration files inside the directory `conf/signing` for signing.

The PKCS#12 file and the appropriate password used for signing can be specified as follows:

config.xml snippet

```
<nativeApp participantId="T7-4-1" >
  <outbox dirPath="application/outbox" msgType="10301">
    <recipientIdResolver filePath="conf/recipientIdResolver.groovy"
      method="resolve" />
    <signingOutbox dirPath="application/signing-outbox"
      signingProfilePath="conf/signature.properties">
      <certificate filePath="conf/demo-certificate.p12"
        password="123456"/>
    </signingOutbox>
  </outbox>
  <inbox dirPath="application/inbox" msgTypes="10301"/>
</nativeApp>
```

If the certificate provided with the sedex client is to be used for signing, the corresponding configuration file of the sedex client can be referenced instead. Since the sedex client manages the renewal of certificates on its own, MH will check the the configuration file for modifications. When the file was modified, it will be reloaded by MH and the renewed certificate will be used for signing.

config.xml snippet

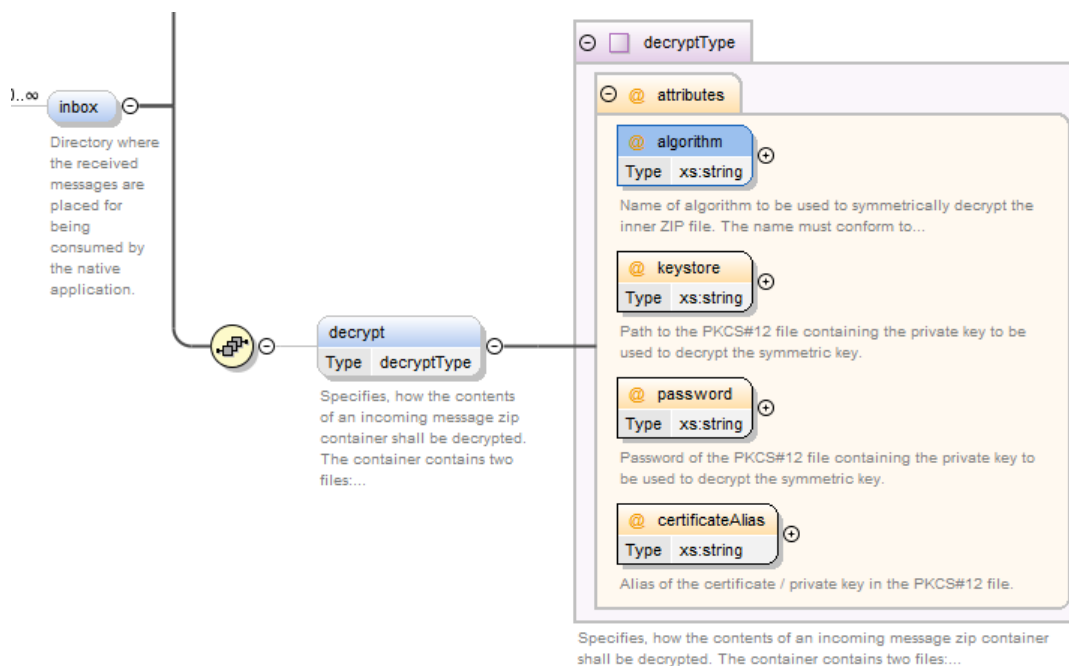
```

<nativeApp participantId="T7-4-1" >
  <outbox dirPath="application/outbox" msgType="10301">
    <recipientIdResolver filePath="conf/recipientIdResolver.groovy"
      method="resolve" />
    <signingOutbox dirPath="application/signing-outbox"
      signingProfilePath="conf/signature.properties">
      <certificateConfigFile
        filePath="../installDir/conf/certificateConfiguration.xml"/>
    </signingOutbox>
  </outbox>
  <inbox dirPath="application/inbox" msgTypes="10301"/>
</nativeApp>
    
```

4.5.2 Decrypting an incoming zip file

Starting with MessageHandler 3.2.0 there is support for decrypting incoming zip files, please also see chapter 2.5.5.

The following fragment shows the configuration:



The keystore is a PKCS#12 file which must contain a certificate with associated public key and private key. The alias of the certificate entry inside the PKCS#12 file must match.

Note that the algorithm has to match the one used while symmetric encryption was applied. The valid values are:

<http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Cipher>.

Note that for decryption with strong cryptography the JCE must be installed.

A sample configuration:

config.xml snippet

```

<nativeApp participantId="T7-4-1">
  <inbox dirPath="application/inbox" msgTypes="10001">
    <decrypt algorithm="AES/CTR/NOPADDING"
      keystore="/mh_config/keystore.p12"
    >
  </inbox>
</nativeApp>
    
```

```

        password="123456" certificateAlias="companysigncert"/>
    </inbox>
</nativeApp>
    
```

4.6 TransparentApp definitions in config.xml

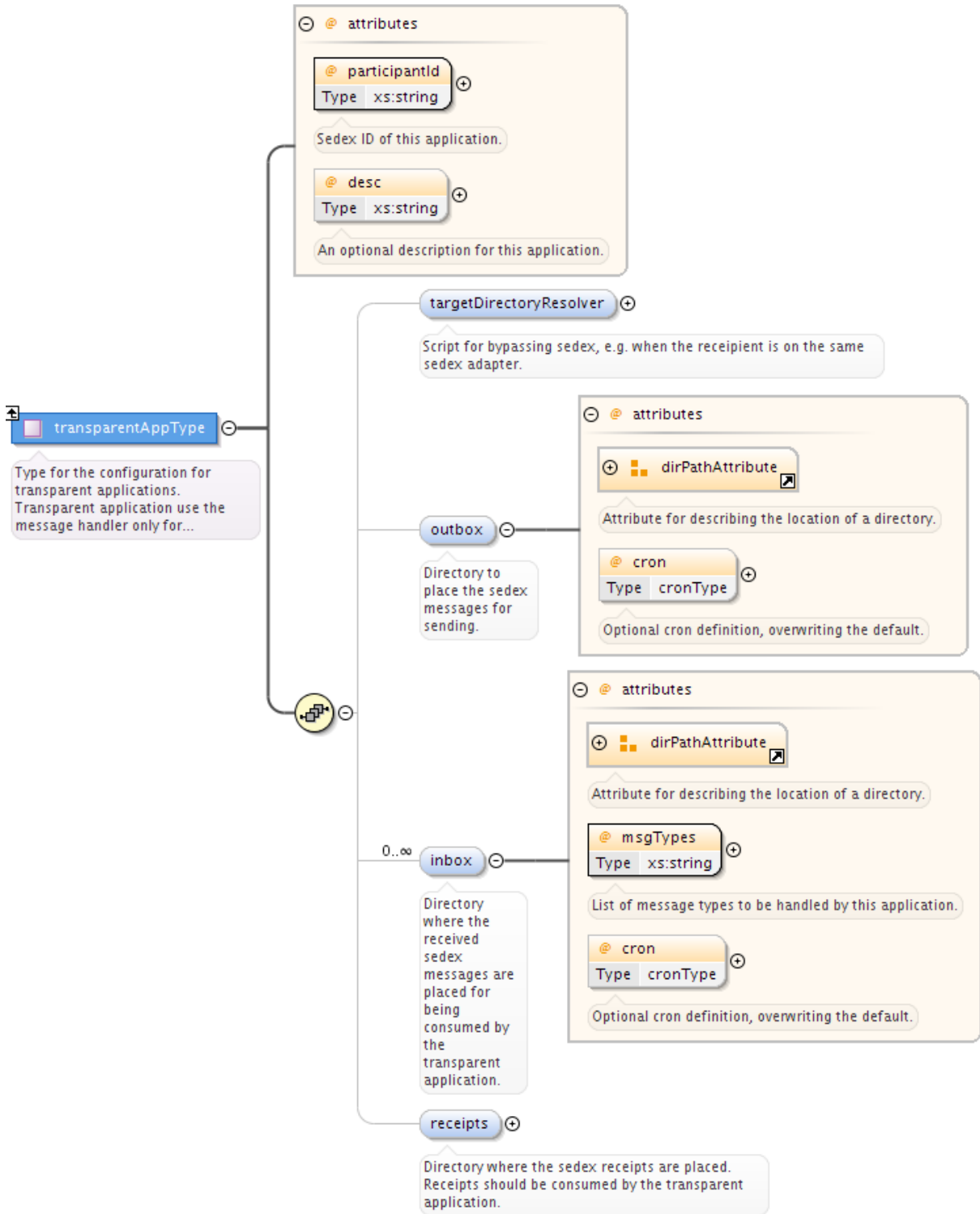


Figure 13 TransparentApp definitions in config.xml

A `transparentApp` can have 0..1 outboxes and 0..n inboxes. A `transparentApp` is able to work with exact one `participantId` (the receiving part).

An outbox does not require `msgType` attribute. Because the application itself is responsible to generate the correct data and envelope file. An inbox is able to receive multiple `msgTypes`.

Note: It's not allowed that two or more inboxes have the same `msgTypes`. In the context of the same `transparentApp`.

It is possible to set other `cron` value for the outbox. If this optional attribute is set, it will override the global `cron` value `defaultOutboxCheck`.

4.7 Configure Logging

Edit the file `<installation-dir>/conf/log4j.properties`. In the following two lines place the name of the log directory (created during preparation):

log4j.properties snippet

```
log4j.appender.LOG.File=<log>/message-handler.log
. . .
log4j.appender.PROTOCOL.File=<log>/message-handler.prot
```

For further details about the configurations options for the file `<installation-dir>/conf/log4j.properties` see [3].

Next edit the file `<installation-dir>/conf/wrapper.conf`. Locate the two line below and place the name of the log directory (created during preparation) there (by replacing the placeholder `<log>` below).

wrapper.conf snippet

```
# Log file to use for wrapper output logging.
wrapper logfile=<log>/wrapper.log
```

4.8 Configure the Global Protocol

If you want to have the MessageHandler to keep a global protocol file (see chap. 2.7.2) you have to have the following lines in `<installation-dir>/conf/log4j.properties`.

log4j.properties snippet

```
# protocol Logger
log4j.logger.GlobalLog=INFO
log4j.additivity.GlobalLog=false
```

4.9 Addressing

MessageHandler provides powerful means for

- Determining (resolve) the `recipientId` of the recipient when sending a message. This is done with the `recipientIdResolver`. This resolver is required for all `nativeApps`.

4.9.1 Resolve the recipientId when sending a message

Note: The resolution of the recipients sedex address (`recipientId`) is only supported with `nativeApps`.

Resolution of the recipients sedex address (`recipientId`) is **always needed with nativeApp** when sending a message. Since an application sends a message by placing a file in an outbox directory (see chap. 2.5.1), we need a way to resolve the `recipientId` by inspecting the file name or may be even its content. This is the `recipientIdResolver`.

Example: The eSchKG network uses the following document naming convention for addressing participants:

`<recipientId>_<original filename>`

The `recipientId` of the eSchKG Testbed is 7-4-1. If you wanted to send a file `testfile.xml` to it, the name of the document must be changed to `7-4-1_testfile.xml`

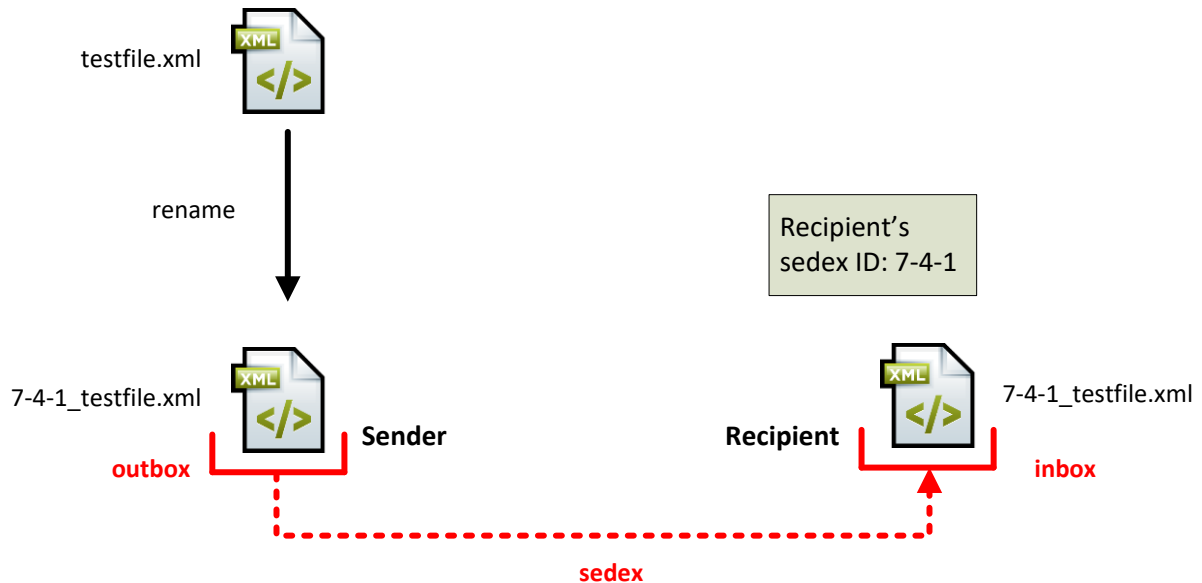


Figure 14 – resolve the recipientId

A Groovy script which resolves the `recipientId` of the recipient when sending an eSchKG/e-LP/e-LEF message could be (see the file `recipientIdResolver.groovy` in the MH distribution):

recipientIdResolver.groovy

```
/**
 * This resolver works for eSchKG messages. The sedex ID will be
 * extracted
 * from the filename.
 *
 * @param filename the name of the file to be sent including path
 * @return the resolved Sedex-ID or an empty string
 */
def String resolve(String filename) {
    Matcher matcher = null;
    if (System.getProperty("os.name").startsWith("Windows") ) {
        matcher = (filename =~ /^.*\\(\S+?)_.*$/ )
    } else {
        // hopefully some sort of Unix
        matcher = (filename =~ /^.*\/(\S+?)_.*$/ )
    }
    if (matcher.matches()) {
        return matcher.group(1)
    }

    // If all else fails: return empty string
    return ''
}
```




4.9.2 Summary of scripting resolvers

A script can only be implemented in Groovy. The scripts must implement a function (method) which takes one parameter. The implemented function (method) will return a string. If the function cannot determine the value expected, the function has to return an empty string (or null).

Note: Modifications on the script itself will become active without a MH restart.

XML tag name	Parameter Type	Return value	Description
<code>recipientId Resolver</code>	String (filename)	String (sedex ID)	The parameter is the filename inclusive the absolute path. It will return the sedex ID. This resolver resolves the <code>recipientId</code> . This resolver is required for each <code>nativeApp</code> .

The Groovy script libraries are provided with the installation in the `/lib` directory.

5 Webservice Interface

The MessageHandler supports the following three HTTP interfaces to get information about the MessageHandler and the state of messages

- Ping:
Information about the liveness of the MessageHandler.
- Monitor:
Allows monitoring of files processed by the MessageHandler.
- Trigger:
Allows triggering (starting) various actions.

If nothing other is defined all request URI and the parameters are case sensitive!

Each HTTP request will return an HTTP Status Code:

Status Code	Description
200	Ok. The request was successfully processed. The result format and MIME content type is interface specific.
400	Bad Request. This happens if the client uses a wrong parameter or passes invalid data to the server. The MIME content type of the response is 'text/plain'.
405	Method not allowed. Will be returned if the client uses an HTTP POST or PUT instead of a GET request. The MIME content type of the response is "text/html"
500	Internal Server Error. Will be returned if the MessageHandlers web server has an internal problem. The MIME content type of the response is "text/html"

5.1 Ping

5.1.1 Introduction

With the Ping request the user has the possibility to get application and system environment information such as: Application alive, memory usage and additional information.

The Ping interface supports an optional parameter called "type".

5.1.2 Request

HTTP Get: SERVER:PORT/message-handler/ping

Parameter	Description
type	Supported type values: {minimal,heapSpace,permSpace,html}. type=html is assumed, if the parameter is omitted.

5.1.3 Response

The result depends on the parameter.

5.1.3.1 No Parameter or type=html

The result content-type is "text/html". This a human readable output with an html table with keys and values.



5.1.3.2 type=minimal

The result content-type is "text/plain". The returned text is "ok", if the MessageHandler is alive

5.1.3.3 type=heapSpace

The result content-type is "text/plain". This returns the current memory usage of the heap that is used for object allocation. The format is: X:Y:Z.

Value	Description
X	The amount of used memory in bytes.
Y	The maximum amount of memory in bytes that can be used for memory management.
Z	Percent [%] of used memory

5.1.3.4 type=permSpace

The result content-type is "text/plain". This returns the current memory usage of non-heap memory that is used by the Java virtual machine. The format is: X:Y:Z.

Value	Description
X	The amount of used memory in bytes.
Y	The maximum amount of memory in bytes that can be used for memory management.
Z	Percent [%] of used memory

5.1.4 Examples

Request without parameter:

```
GET /message-handler/ping HTTP/1.0
```

Request with type=html:

```
GET /message-handler/ping?type=html HTTP/1.0
```

Response to request using no parameters or type=html:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=iso-8859-1
Content-Length: 3857
Server: Jetty(6.1.7)
```

```
<html><body><h1>Alive!</h1><table><tr><td><b>getHeap</b></td></tr><tr><td>committed</td><td>62.0MB</td></tr><tr><td>init</td><td>64.6MB</td>
...
</body></html>
```

Request using type=minimal:

```
GET /message-handler/ping?type=minimal
```

Response to request using type=minimal:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 3
Server: Jetty(6.1.7)
```

```
ok
```

Request using type=heapSpace

```
GET /message-handler/ping?type=heapSpace
```

Response to request using type= heapSpace:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 20
Server: Jetty(6.1.7)
```

```
5330856:920911872:1
```

Request using type=permSpace

```
GET /message-handler/ping?type=permSpace
```

Response to request using type= permSpace:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 21
Server: Jetty(6.1.7)
```

```
17008920:224395264:8
```

Request with invalid type:

```
GET /message-handler/ping?type=INVALID
```

Response to request with invalid type:

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 77
Server: Jetty(6.1.7)
```

Invalid parameter. Valid parameters:
 type={minimal, heapSpace, permSpace, html}

5.2 Monitor

5.2.1 Introduction

⚠ The Monitor HTTP Interface is only usable for messages sent in native mode!

With the Monitor HTTP Get Interface it is possible to retrieve the state of one or more files which the MessageHandler is processing or has processed.

The Monitor Interface supports multiple parameters which can be combined to filter the number of results. When multiple parameters will be used, they will be combined with an AND operation. The response is a JSON⁵ data structure.

5.2.2 Request

HTTP Get: SERVER:PORT/message-handler/monitor?{filename, sedexId, event, from, until, messageId}

At least one parameter is required. The parameters can be combined with each other. Multiple occurrences of the same parameter is not supported. If more than one parameter is set, they will be processed as an AND operation.

The following parameters are supported:

Parameter	Type/Format	Description
filename	String	Search by a filename. The filename is not case-sensitive. Substring will match. Example: Search with "world" would match with: "HelloWorld.pdf"
sedexId	String	This search for a participant (receivers) sedex ID. Exact match and not case-sensitive.
state	String	Search for a messages in a certain state. (refer to chap. 4.4.45.2.3). Valid values are (not case-sensitive): SENDING, SENT, FORWARDED, DELIVERED, EXPIRED and ERROR.
from	String (ISO 8601 Date/Time format)	The "from" date will return all values which are newer than the given date. For example "2007-04-05T14:30".
until	String (ISO 8601 Date/Time format)	The "until" date will return all values which are older than the given date. For example "2007-04-05T14:30".
messageId	String	The sedex message Id as generated by MH (UUID version 4) for messages send in Native Mode:

5.2.3 Response

The MIME content type is 'application/json'. The response body is an array of (0 ..n) JSON objects. The attributes of the JSON object are the following:

⁵ <http://www.json.org/>

Attribute	Data Type	Description
recipientId	String	Sedex Id of the receiver.
filename	String	The filename.
messageId	String	The message ID generated by MH (for messages sent in <code>Native Mode</code>) or the message ID in the sedex envelope (for messages send in <code>Transparent Mode</code>).
sentDate	String (ISO 8601 Date/Time format)	The date when the file was sent. This attribute will be omitted, if the message is in the <code>SENDING</code> state.
receivedDate	String (ISO 8601 Date/Time format)	The date when the file was received by the participant. This attribute is only present, if the message is in the <code>DELIVERED</code> state.
state	String	The state of this message. Details see below.
mode	String	The source of this message. Details see below.

The possible values for the **state** attribute are:

Name	Description
SENDING	The message is being sent, the files are taken from the outbox and put into a ZIP file.
SENT	The message is successfully sent, the ZIP file is transferred to the outbox of the sedex client, but not yet delivered or expired (there is no confirmation receipt).
FORWARDED	The message is taken and sent by the sedex client, but there is no confirmation receipt yet.
DELIVERED	The message has been received by the recipient. There is a receipt in the receipts directory of the sedex client.
EXPIRED	The message was not delivered and is expired (i.e. the recipient has failed to collect his messages in time). There is a receipt in the receipts directory of the sedex client.
ERROR	An error is reported by the sedex client; there is a receipt in the receipts directory.

The possible values for the **mode** attribute are:

Name	Description
MH	The message has been sent in <code>Native Mode</code> .
TRANSP	The message has been sent in <code>Transparent Mode</code> .

5.2.4 Examples

Note: For readability reasons the JSON part of the result is formatted.

Request with a single result:

```
GET /message-handler/monitor?filename=demo1 HTTP/1.0
```

Response with a single result:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=ISO-8859-1
```



```
Content-Length: 187
Server: Jetty(6.1.7)
```

```
[ { "filename" : "demo1.pdf",
  "messageId" : "2b349f2b-aa16-4fc2-b9ad-48f4158daaf1",
  "mode" : "MH",
  "recipientId" : "T7-4-1",
  "sentDate" : "2012-07-31T15:34:18.819+02:00",
  "state" : "FORWARDED"
} ]
```

Request with two results:

```
GET /message-handler/monitor?filename=Demo HTTP/1.0
```

Response with two results:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=ISO-8859-1
Content-Length: 376
Server: Jetty(6.1.7)
```

```
[ { "filename" : "demo1.pdf",
  "messageId" : "2b349f2b-aa16-4fc2-b9ad-48f4158daaf1",
  "mode" : "MH",
  "recipientId" : "T7-4-1",
  "state" : "SENDING"
},
{ "filename" : "demo2-sig.pdf",
  "messageId" : "a8c62118-2d7f-4c28-9f4a-4fc1523a6bd2",
  "mode" : "MH",
  "recipientId" : "T7-4-1",
  "receivedDate" : "2012-07-31T15:42:06.521+02:00",
  "sentDate" : "2012-07-31T15:34:28.835+02:00",
  "state" : "RECEIVED"
}
]
```

Request with no results:

```
GET /message-handler/monitor?filename=notExist.pdf HTTP/1.0
```

Response with no result:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=ISO-8859-1
Content-Length: 3
Server: Jetty(6.1.7)
```

```
[ ]
```

Request with invalid date:

```
GET /message-handler/monitor?from=2012-30-07 HTTP/1.0
```

Response to request with invalid date:

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=iso-8859-1
```

Content-Length: 94
Server: Jetty(6.1.7)

Invalid request: Unable to parse parameter 'from'. Details: Invalid value 30 for Month field.

Request with valid date:

GET /message-handler/monitor?from=2012-07-31 HTTP/1.0

Response to request with valid date:

HTTP/1.1 200 OK
Content-Type: application/json; charset=ISO-8859-1
Content-Length: 376
Server: Jetty(6.1.7)

```
[ { "filename" : "demo1.pdf",  
  "messageId" : "2b349f2b-aa16-4fc2-b9ad-48f4158daaf1",  
  "mode" : "MH",  
  "recipientId" : "T7-4-1",  
  "sentDate" : "2012-07-31T15:34:18.819+02:00",  
  "state" : "FORWARDED"  
},  
{ "filename" : "demo2-sig.pdf",  
  "messageId" : "a8c62118-2d7f-4c28-9f4a-4fc1523a6bd2",  
  "mode" : "MH",  
  "recipientId" : "T7-4-1",  
  "sentDate" : "2012-07-31T15:34:28.835+02:00",  
  "state" : "FORWARDED"  
}  
]
```

5.3 Trigger

5.3.1 Introduction

With the trigger interface can be used to immediately start jobs (which are configured in the config.xml). Following jobs can be started:

- Send: Starts a send job
- Receive: Starts the receive job
- Poll: Starts the checker job

5.3.2 Request

HTTP Get: SERVER:PORT/message-handler/trigger

Parameter	Description
action	Supported actions are: {receive,poll,send}
name	To be provided, if action=send. Specifies the name of the Outbox for which sending should be started.

5.3.3 Response

The MIME content type of the response is always text/plain. If the HTTP status code 200 is returned the trigger action was successfully started. The response text contains a reference to which action has been started.



5.3.4 Examples

Request to trigger the receive action:

```
GET /message-handler/trigger?action=receive HTTP/1.0
```

Response to request to trigger the receive action:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 36
Server: Jetty(6.1.7)
```

```
receive action successfully started
```

Request to trigger the poll action:

```
GET /message-handler/trigger?action=poll HTTP/1.0
```

Response to request to trigger the poll action:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 26
Server: Jetty(6.1.7)
```

```
poll successfully started
```

Request to trigger the send action:

```
GET /message-handler/trigger?action=send&name=outbox3 HTTP/1.0
```

Response to request to trigger the send action:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 28
Server: Jetty(6.1.7)
```

```
send successfully triggered
```

Request to trigger the send action on an outbox directory which not exist:

```
GET /message-handler/trigger?action=send&name=NOTEXIST HTTP/1.0
```

Response to request to trigger the send action on an outbox directory which not exist:

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=iso-8859-1
Content-Length: 46
Server: Jetty(6.1.7)
```

```
No outbox directory found with name: NOTEXIST
```

6 Change Log

6.1 What is new in version 3.4.4

- Improvement
 - [MSGHANDLER-107] – Added better error message when a file is locked.
 - [MSGHANDLER-106] – Upgrade to Corretto 8.402.08.1 for the MSI Installer distribution

6.2 What is new in version 3.4.3

- Bugfixes
 - [MSGHANDLER-104] – addition of LTV information is now disabled when signed PDF documents. Reason: the certificates issued by the Swiss Government PKI, which are used by the collection offices, refer to a CRL, which is too large to be integrated in the signature.
This bug prevented collection offices to sign PDF files.
- Improvement
 - [MSGHANDLER-106] -- Upgrade to Corretto 8.382.05.1 for the MSI Installer distribution

6.3 What is new in version 3.4.2

- Bugfixes
 - [MSGHANDLER-98] – Corrupted PDF files which cannot be signed are now moved to the Corrupted folder
- Improvement
 - [MSGHANDLER-97] – Logging has been improved to show more detailed information about the problem when a file cannot be moved.
 - [MSGHANDLER-101] – Access check added so that files are no longer sent multiple times in case of an access problem.
 - [MSGHANDLER-103] – Update libraries to fix vulnerabilities
 - Update to SUIB-Batchsigner 1.8.2
 - Update to Groovy 2.4.21
 - Update to File-Encryptor 1.0.3
 - Update Jetty to v 9.4.51.v20230217
 - Migration to Log4j 2.20
 - Commons-beanutils was removed
 - Commons-collections was removed
 - Commons-logging was removed
 - PDF Jempbox was removed

6.4 What is new in version 3.4.1

- Bugfixes
 - [MSGHANDLER-82] get RESToption /message-handler/ping now displays the current version of MH
- Improvement
 - [MSGHANDLER-100] Upgrade to Corretto 8.342.07.3 **for Windows Installer**

6.5 What is new in version 3.4.0

- Change
 - [MSGHANDLER-81], [MSGHANDLER-84] – Amazon Corretto 8.242 packaged with MSI Installer
- New Feature
 - [MSGHANDLER-78] – upgraded several libraries
 - [MSGHANDLER-79] – Docker image for MessageHandler



- [MSGHANDLER-80] – code added to prevent XXE (External Entity Processing) vulnerabilities
- Improvement
 - [MSGHANDLER-77] – upgrade the embedded database HSQLDB from version 1.8.0.10 to version 2.4.1. This will hopefully solve the out of memory problems on Windows when compacting the database.
 - [MSGHANDLER-87] – start wrapper in the Docker Container as process with pid=1
 - [MSGHANDLER-88] – changes in build process for MSI installer

6.6 What's new in version 3.3.5

- Bugfixes
 - [MSGHANDLER-74] now works with the changed Sedex XML configuration (“cannot validate against config.xsd”-issue)
 - [MSGHANDLER-75] updating dependencies

6.7 What's new in version 3.3.4

The sole change from 3.3.3 to 3.3.4 is a bugfix in the Windows MSI installer package. The MSI installer package of v 3.3.3 erroneously installed a wrong version of the file recipientIdResolver.groovy in its configuration directory.

Note: The ZIP package for Windows and the distribution for Linux are not affected. If you used any of these packages, you are not required to update.

6.8 What's new in version 3.3.3

- Bugfixes
 - [MSGHANDLER-69] – MH has problems to start up, when its status DB is too big. The reason for this issue is, that MH tries to clean up its status DB upon start up. When the DB is big, this process can take several minutes. In this case the Tanuki service wrapper loses patience and terminates the start up process. Countermeasure in v 3.3.3:
 - The clean-up of the DB is done in a separate thread in order not to block the start-up process.
 - The size of the DB log has been reduced to 1 MB (was 200 MB)
 - The DB file is compacted after cleaning up
 - [MSGHANDLER-70] - Wrong version number was printed out to the log file (again!)
- Improvement
 - [MSGHANDLER-68] - Upgrade to Tanuki Service Wrapper 3.5.34
 - [MSGHANDLER-71] – PDF files, which could not be signed are moved into the corrupted directory

6.9 What's new in version 3.3.2

- Bugfixes
 - [MSGHANDLER-58] – MSI Installer did not work under Windows 10 Pro v 1607 and cause problems under some Windows Server installations. We changed the installation process to address these issues.
 - [MSGHANDLER-61] – MSI Installer showed application type „Collecting Office“ although “Creditor” was chosen.
 - [MSGHANDLER-64] – Signing of PDF files terminated, when a single PDF file could not be signed.
- New Features
 - [MSGHANDLER-67] - As a service for those who install MH manually, the installation directory of MH is now “self-contained”. The config.xml references directories in the installation directory, which allows testing the correct startup of MH prior to creating the productive working and interface directories.
- Improvements

- [MSGHANDLER-65] - MH now logs information found in the Sedex envelop (Sedex ID of sender and recipient, message type and message ID) when dispatching received messages.
- [MSGHANDLER-66] - Since Java 7 is not supported anymore by Oracle, MH now requires Java 8 to run! We recommend you install at least JRE 8u152.

6.10 What's new in version 3.3.1

- [Change Request] SHA-256 is now used to signed PDF document

6.11 What's new in version 3.3.0

- Possibility to rename the target folder with a Groovy script while decompressing the encrypted ZIP files

6.12 What's new in version 3.2.0

- [Feature Request] Allow to decrypt incoming messages (zip files) for nativeApps
- Dropped Java 6 support
- Installation packages separated between Linux (64-Bit) and Windows (32-Bit)

6.13 What's new in version 3.1.6

- [Bug] Wrong version number was printed out to the log file.
- [Bug] Reading lot of files ends in performance problems.
- [Bug] Project source file encoding change to utf-8.

6.14 What's new in version 3.1.5

- [Bug] Only concerns PDF signing for nativeApps: After having signed PDF files, MH does not only move (or delete) the files he has signed, but just all the files in the signing_outbox. This problem occurs if an application places a new PDF file in the signing_outbox just after the moment where MH has begun to sign.

6.15 What's new in version 3.1.4

- [Feature Request] Variables used in the sedex configuration file certificateConfiguration.xml (e.g. <location>\${SEDEX_HOME}/zertifikate/fence-IT-AG-CH-701.0.010.742-8-10300.p12</location>) will now be replaced with the directory above the directory where certificateConfiguration.xml is located. Alternatively an environment variable with the same name can be defined to point to the directory where the sedex client is installed.

6.16 What's new in version 3.1.3

- [Bug] Windows UNC paths can now the specified with backslashes (\\server\directory) as well as with slashes (//server/directory).

6.17 What's new in version 3.1.2

- [Feature Request] MH now behaves correctly, if the filesystem gets full while receiving or sending messages.
- [Bug] Windows UNC paths can now the specified with backslashes (\\server\directory) as well as with slashes (//server/directory).
- [Feature Request] MH now behaves correctly, if a remote filesystem (e.g. NFS or CIFS shared directory) is not available.
- [Change Request] Better error logging in case of usages of variables inside certificateConfiguration.xml



6.18 What's new in version 3.1.1

- [Bug] MH can't parse productive certificateConfiguration.xml files.

6.19 What's new in version 3.1

- [Feature Request] SigningOutbox: automatic exchange of sedex certificate must be considered → <certificateConfigFile/> tag for signing outboxes.
- [Change Request] Change product name
- [Bug] MH now correctly handles multiple <recipientId> in sedex envelope file

6.20 What's new in version 3.0.1

- [Feature Request] MH recognizes at start-up, if the config file matches the correct XML schema
- [Feature Request] MH has been tested with Java SE 7
- [Change Request] Upgraded the Java Service Wrapper to latest release
- [Change Request] wrapper.conf: no dependency declared to SedexAdapter
- [Bug] wrapper.conf: config.xml is now expected in ../conf (instead of ../bin)
- [Bug] recipientIdResolver.groovy now works with Linux/Unix and Windows

6.21 What's new in version 3.0

In the MessageHandler version 3.0 are several new features and some other are removed.

The new features are:

- Complete refactoring of the configuration of MessageHandler. When upgrading from version 2.x of MessageHandler to version 3.0, the configuration file (config.xml) has to be completely rewritten.
- PDF Signing based on PKCS12 for native applications
- The protocol per file (.prot / .err) can now be en-/disabled though a configuration option.
- Replacement of "targetDirectoryResolver" script with "localRecipient" configuration. This new feature allows MessageHandler to deliver messages to local (transparent as well as native) applications.
- Extended HTTP interface which now allows for monitoring messages sent by MessageHandler.
- New documentation. Now in English only!

The removed features are:

- MySQL connector for logging (<protocol> Tag)
- Hook Script removed (<hook> Tag)
- Removed unnecessary scripts

7 References

- [1] sedex-Handbuch, Version 4.0.3, 15.11.2012,
<http://www.bfs.admin.ch/bfs/portal/de/index/news/00/00/12/01.html>
- [2] GNU License,
<http://www.gnu.org/licenses>
- [3] Log4j, The Complete Manual, Ceki Gülcü, 2004,
www.qos.ch
- [4] Java Service Wrapper, Configuration Property Overview,
<http://wrapper.tanukisoftware.org/doc/english/properties.html>
- [5] Quartz, Cron Triggers, Open Symphony,
<http://www.opensymphony.com/quartz/wikidocs/TutorialLesson6.html>
<http://www.quartz-scheduler.org/documentation/quartz-2.1.x/tutorials/crontrigger>

- [6] Open eGov BatchSigner,
<http://www.openegov.admin.ch/content/egov/de/home/produkte/signieren/batchsigner.html>
- [7] PKCS #12: Personal Information Exchange Syntax Standard,
<http://en.wikipedia.org/wiki/PKCS12>
- [8] Open eGov MessageHandler binary, sources and documentation download,
<https://www.e-service.admin.ch/wiki/display/openegovdoc/MessageHandler>
- [9] Open eGov MessageHandler examples and FAQ,
<https://www.e-service.admin.ch/wiki/display/openegovdoc/MessageHandler>
- [10] eSchKG Standard documentation,
<http://www.eschkg.ch>