EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

FS Plugin

Configuration Guide

Domibus File System Plugin

Version [2.8]

Status [Final]

Date: 21/08/2023

Document Approver(s):

Approver Name	Role
Bogdan DUMITRIU	Project Manager

Document Reviewers:

Reviewer Name	Role
Cosmin BACIU	Technical Lead
Caroline AEBY	eDelivery Support
Catalin-Emanuel ENACHE	eDelivery Technical Office

Summary of Changes:

Version	Date	Created by	Short Description of Changes
0.01	16/10/2017	Chaouki BERRAH	Initial Document
0.02	25/10/2017	Chaouki BERRAH	Domain Setup added
0.03	27/10/2017	Chaouki BERRAH	Internal Review
0.04	13/11/2017	Chaouki BERRAH	Updated with Cosmin Comments
1.0	21/11/2017	Chaouki BERRAH	Final
1.1	20/02/2018	Chaouki BERRAH	Metadata.xml examples added
1.2	09/03/2018	Chaouki BERRAH	Metadata availability configuration Added (Jira 2908)
1.3	20/03/2018	CEF Support	Reuse notice added
1.4	21/06/2018	CEF Support	Reference to nexus replaced by reference to CEF Digital site.
1.5	16/08/2018	CEF Support	Default PMode changes in Domibus 4.0 version.
1.6	20/09/2018	Ioana DRAGUSA	Update with Multitenancy
1.7	26/09/2018	Caroline AEBY	End of standby service
1.8	07/02/2019	Sebastian-Ion TINCU Chaouki BERRAH	fsplugin.messages.payload.id property added (Update for Domibus 4.0.2)
1.9	20/05/2019	Caroline AEBY	2018 => 2019 + approver's role updated
2.0	16/07/2019	Caroline AEBY	Mistakes corrected
2.1	28/08/2019	Chaouki BERRAH	To the purpose => for the purpose.
2.2	15/05/2020	François GAUTIER	Add corner case for payload name not used
2.3	15/09/2020	Chaouki BERRAH	Profile configuration.

2.4	05/10/2020	Chaouki BERRAH	Updates including fsplugin.messages.sent.purge.expired=
2.5	30/11/2020	Cosmin BACIU Caroline AEBY	Domibus 4.2 document review
2.6	18/01/2021	Chaouki BERRAH	Configuration of different recipients
2.7	27/04/2022	Caroline AEBY	No more CEF references + links update
2.8	21/08/2023	Caroline AEBY	Cef_edelivery replaced by edelivery

Table of Contents

1. INTRODUCTION 6

2. CONVENTIONS, REFERENCES AND PREREQUISITES 7

2.1. Conventions 7
2.1.1. Example: Sample Configuration file 7
2.2. References 9
2.3. Prerequisites 9

3. FILE SYSTEM CONFIGURATION PROCEDURE 10

3.1. File System Plugin download and installation 10
3.2. File System Plugin location configuration 10
3.2.1. Local 11
3.2.2. SMB/CIFS 11
3.2.3. SFTP 11
3.2.4. FTP 12
3.3. FS Plugin Properties Configuration 12
3.4. Message Filter configuration 13
3.5. Metadata.xml configuration 14
3.6. PMode configuration 15

4. SENDING PROCEDURE OPTIONS 17

4.1. Scenario 1: WS-plugin is used at the sender's side 17
4.1.1. Using PayloadName 17
4.1.2. PayloadName not used option 19
4.2. Scenario 2: FS-plugin is used at the sender's side 20

5. FINE TUNING 22

5.1. Delete/Archive Message 22
5.1.1. Delete 22
5.1.2. Archive 22
5.1.3. Purge 22
5.2. Specific Domain Setup 22
5.2.1. fs-plugin.properties Domain Configuration changes 22
5.2.2. Metadata.xml Domain Configuration changes 23
5.2.3. PMode Domain Configuration changes 24
5.2.4. Multitenancy 25
5.2.5. Different recipients 26

6. FILE SYSTEM PERMISSIONS 27

6.1. Local File Systems: 27
6.2. Remote File Systems: 27

7. CONTACT INFORMATION 28

1. INTRODUCTION

This Guide is intended for Administrators in charge of installing, managing and troubleshooting an eDelivery Access Point.

The purpose of this guide is to provide detailed information on how to configure and deploy the File System Plugin available in Domibus 3.3.2 and later versions.

The instructions that are provided are valid for all the platforms supported by Domibus, including WebLogic, Tomcat and Wildfly, with MySQL or Oracle backend.

2. CONVENTIONS, REFERENCES AND PREREQUISITES

2.1. Conventions

The Commands and Configuration files listed in this document usually contain a mix of reserved words (commands, instructions and system related special words) and user defined words (chosen by the user) as well as comments and preferred values for certain variables. The conventions used in this document, to distinguish between them, are the followings:

- To keep this document release agnostic as much as possible, the strings "x-y-z" or "x.y.z" are intended to refer to the version of Domibus discussed in this version of the document, in the present case "Domibus FS Plugin".
- **Bold** is used for "reserved" words and commands.
- *Normal italic* together with a short description of the argument is used for user-defined names (chosen by you to designate items like users, passwords, database etc.). It normally contains at least 2 words separated by "_".
- ***Bold and Italic*** is used for variables names that can be modified by the user.
- Comments are sometimes added to describe the purpose of the commands, usually enclosed in brackets ().

By default, non-OS specific paths will be described using Linux patterns.

2.1.1. Example: Sample Configuration file

```
# E.g.: 1
#fsplugin.domains.DOMAIN1.order=1

# Regular expression used to match the domain for the reception of messages. This regular expression will be evaluated
# against the Service and Action values from the incoming message separated by #.
# E.g.: DOMAIN1SampleService#.*
#fsplugin.domains.DOMAIN1.messages.expression=
fsplugin.domains.SOMEDOMAIN.messages.expression=SomeDomainService#SomeDomainAction

# The location of the folder that the plugin will use to manage the messages to be sent and received in case no domain
# expression matches. This location must be accessible to the Domibus instance. The domain locations must be independent
# from each other and should not overlap.
# E.g.: /home/domibus/fs_plugin_data/DOMAIN1
#fsplugin.domains.DOMAIN1.messages.location=
fsplugin.domains.SOMEDOMAIN.messages.location=/PATH/SOMEDOMAIN

# The user used to access the domain location specified by the property fsplugin.domains.<domain_id>.messages.location.
```

SOMEDOMAIN, PATH, SomeDomainAction are all user defined names which are chosen by the user.

2.2. References

Ref.	Document	Content outline
[REF1]	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus	Documents Locations

2.3. Prerequisites

Domibus 3.3 or a later version needs to be already installed in a folder called ***edelivery***. Please see the Domibus Administration Guide (see [REF1]) for details.

The FS Plugin can either be installed on the Sending Access Point or on the receiving Access Point, or on both.

3. File System Configuration Procedure

The FS (File System) Plugin configuration procedure includes:

- Downloading and installing the FS Plugin
- Specifying the FS Plugin location and setting its permissions
- Configuring the message filter
- Configuring the metadata.xml file
- Configuring the PMode

Each of these steps is described in details below.

3.1. File System Plugin download and installation

- Download the **domibus-distribution-X.Y.Z-default-fs-plugin.zip** from the Digital website see §2.3
- Extract the zip file downloaded previously
- Copy the domibus-default-fs-plugin-X.Y.Z.jar file to ***edelivery*** /conf/domibus/plugins/lib folder (where ***edelivery*** is the Domibus installation location)
- Copy the **fs-plugin.xml** and the **fs-plugin.properties** files specific to the Application Server used (**tomcat, weblogic or wildfly**) to ***edelivery***/conf/domibus/plugins/config

3.2. File System Plugin location configuration

The FS Plugin supports multiple file system types via Apache VFS. There are 4 file systems currently supported:

- Local
- SMB/CIFS
- SFTP
- FTP

These four file systems are described below.

3.2.1. Local

A local file system is simply a directory on the local physical system. The URI format is:

[file://]absolute-path

Where ***absolute-path*** is a valid absolute directory name on the local platform. UNC names are supported under Windows.

This type of file system does not support authentication hence the Domibus user needs read/write access to this directory. See § 6 - File System Permissions for more details.

Examples:

file:///home/someuser/somedir
file:///C:/Documents and Settings
/home/someuser/somedir
c:\program files\some dir
c:/program files/some dir

3.2.2. SMB/CIFS

A SMB/CIFS file system is a remote directory shared via Samba or Windows Share, with the following URI format:

smb://hostname[:port]/sharename[/relative-path]

Notice that a share name is mandatory.

This type of file system supports authentication via user and password domain properties. See section 3.1 of the Domibus Interface Control Document File System Plugin (see [REF1]) for more details.

Examples:

smb://somehost/shareA
smb://somehost/shareB/nesteddir

3.2.3. SFTP

An SFTP file system is a remote directory shared via SFTP. Uses an URI of the following format:

sftp://hostname[:port][/relative-path]

The path is relative to whatever path the SFTP server has configured as base directory, usually the user's home directory.

This type of file system also supports authentication via user and password domain properties. See section 3.1 of the Domibus Interface Control Document File System Plugin (see [REF1]) for more details.

Example:

```
sftp://somehost/pub/downloads/
```

3.2.4. FTP

An FTP file system is a remote directory shared via FTP. It accepts URIs of the following format:

```
ftp://hostname[:port][/relative-path]
```

The path is relative to whatever path the FTP server has configured as base directory, usually the user's home directory.

This type of file system also supports authentication via user and password domain properties. See section 3.1 of the Domibus Interface Control Document File System Plugin [see ([REF1])] for more details.

Example:

```
ftp://somehost/pub/downloads/
```

Due to incompatibilities between the current version of VFS and certain FTP servers on Linux (e.g.: vsftpd), using a relative path prevents some of the plugin's functionality from working correctly. For that reason, in those cases an absolute path must be specified, e.g.:

```
ftp://somelinuxhost/home/someuser/pub/downloads/
```

3.3. FS Plugin Properties Configuration

1. Choose a name for the File System Plugin folder, where various types of messages will be stored (e.g: **filestore** refers to this folder in our documentation). Create the folder and make sure it can be read and written by the user running Domibus.

2. Edit the *edelivery_path* /conf/domibus/plugins/conf/fs-plugin.properties.

3. Set **fsplugin.messages.location**=***PATH***/**filestore** where ***PATH*** is the full path to the **filestore** folder.
   Example: ***fsplugin.messages.location=/home/domibus/fs_plugin_data/MAIN.***

4. *Optional: Set the* **fsplugin.messages.payload.id** property (Default is ***cid:message***)

5. Restart Domibus.

The following folders should **automatically** be created under the **filestore** folder, after the successful start of Domibus:

- **IN**: folder where the messages are received.
- **SENT**: folder where the messages sent are stored.
- **FAILED**: folder where the messages that failed to be sent are stored.

- **OUT**: folder from where the messages are sent.

Remark:

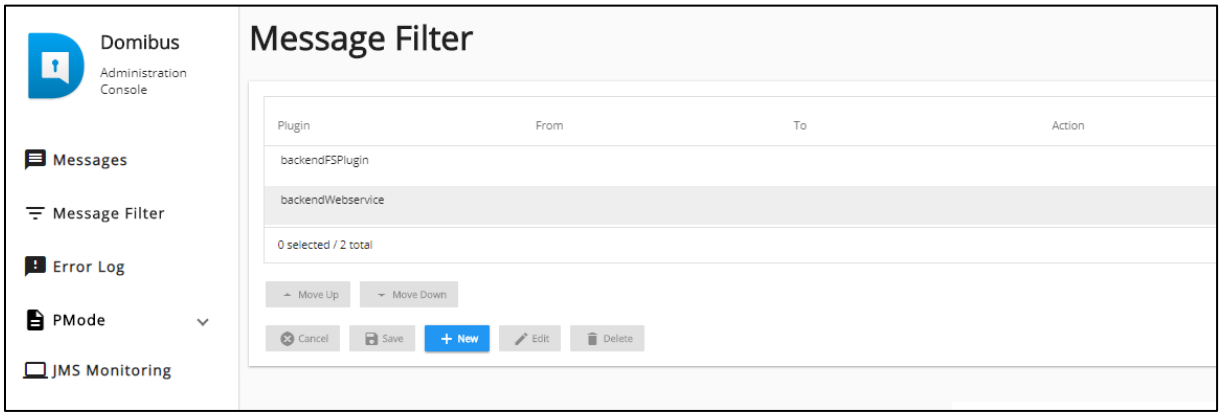
It is possible to configure multiple locations for incoming and outgoing files to be sent with the Domibus file plugin. The procedure of configuring different domains is described in §5.2 - Specific Domain Setup.

Properties defined in the property file *edelivery_path* /conf/domibus/plugins/conf/**fs-plugin.properties** can be used to configure the FS-Plugin. The list of FS plugin properties can be found in the FS Plugin ICD document (see [REF1]).

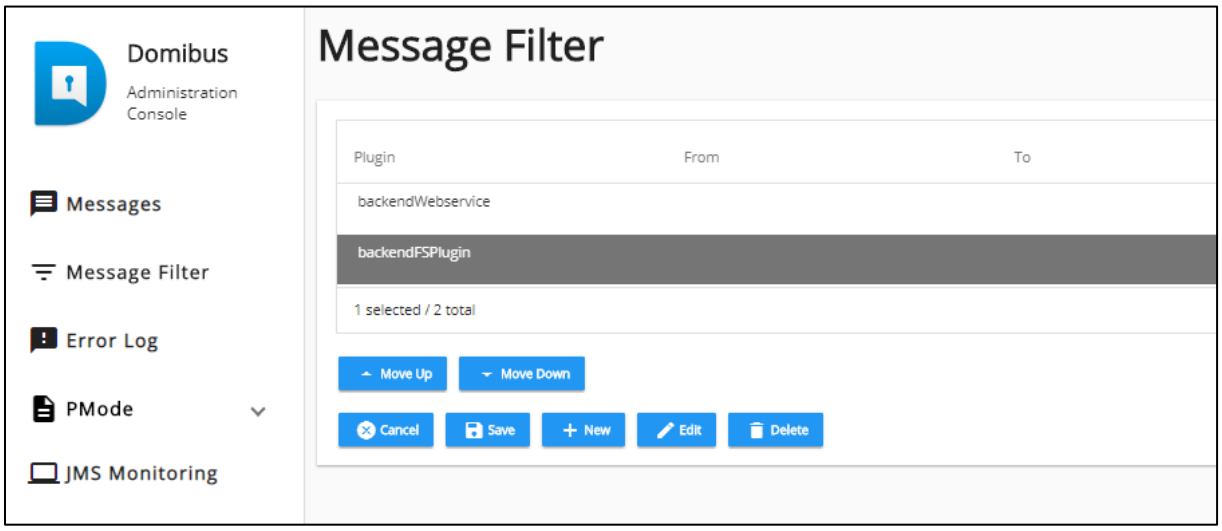
3.4. Message Filter configuration

Log on the Administration Console. Make sure that the FS Plugin is the first in the **order** of Plugins that are processed using the following procedure in the Admin Console:

1. Choose **Messages filter**:



2. Select the **backendFSPlugin**, then click on **Move Up** to move it up to the top of the list:



3. Save the changes.

3.5. Metadata.xml configuration

The metadata.xml contains the details of the sender and receiver party IDs as well as other information and affects all future message exchanges immediately after the file is placed in the OUT(tray) folder of the Sending Access Point.

A sample **metadata.xml** file is provided, which needs to be modified to match the PMode specifications of the sending and receiving Access points used.

```
<?xml version="1.0" encoding="UTF-8" ?>
<UserMessage xmlns="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/">
   <PartyInfo>
      <From>
         <PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
blue</PartyId>
         <Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</Role>
      </From>
      <!--Optional:-->
      <To>
         <PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-red</PartyId>
         <Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder</Role>
      </To>
   </PartyInfo>
   <CollaborationInfo>
      <!--You may enter the following 4 items in any order-->
      <!--Optional:-->
      <!-- <AgreementRef type="">A1</AgreementRef> -->
      <Service type="tc1">bdx:noprocess</Service>
      <Action>TC1Leg1</Action>
   </CollaborationInfo>
   <MessageProperties>
      <!--1 or more repetitions:-->
      <!--originalSender and finalRecipient are mandatory-->
      <Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</Property>
      <Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C4</Property>
   </MessageProperties>
   <PayloadInfo>
      <PartInfo href="cid:message">
         <PartProperties>
            <Property name="MimeType">text/xml</Property>
         </PartProperties>
      </PartInfo>
   </PayloadInfo>

</UserMessage>
```

Sample Metadata.xml file

Copy the metadata.xml file to the sender's filestore/OUT folder.

Remark:

For each message received a metadata.xml file is also generated by FS-Plugin and stored in the ***/IN/FINAL_RECIPIENT/MESSAGE_ID/…*** *folder*

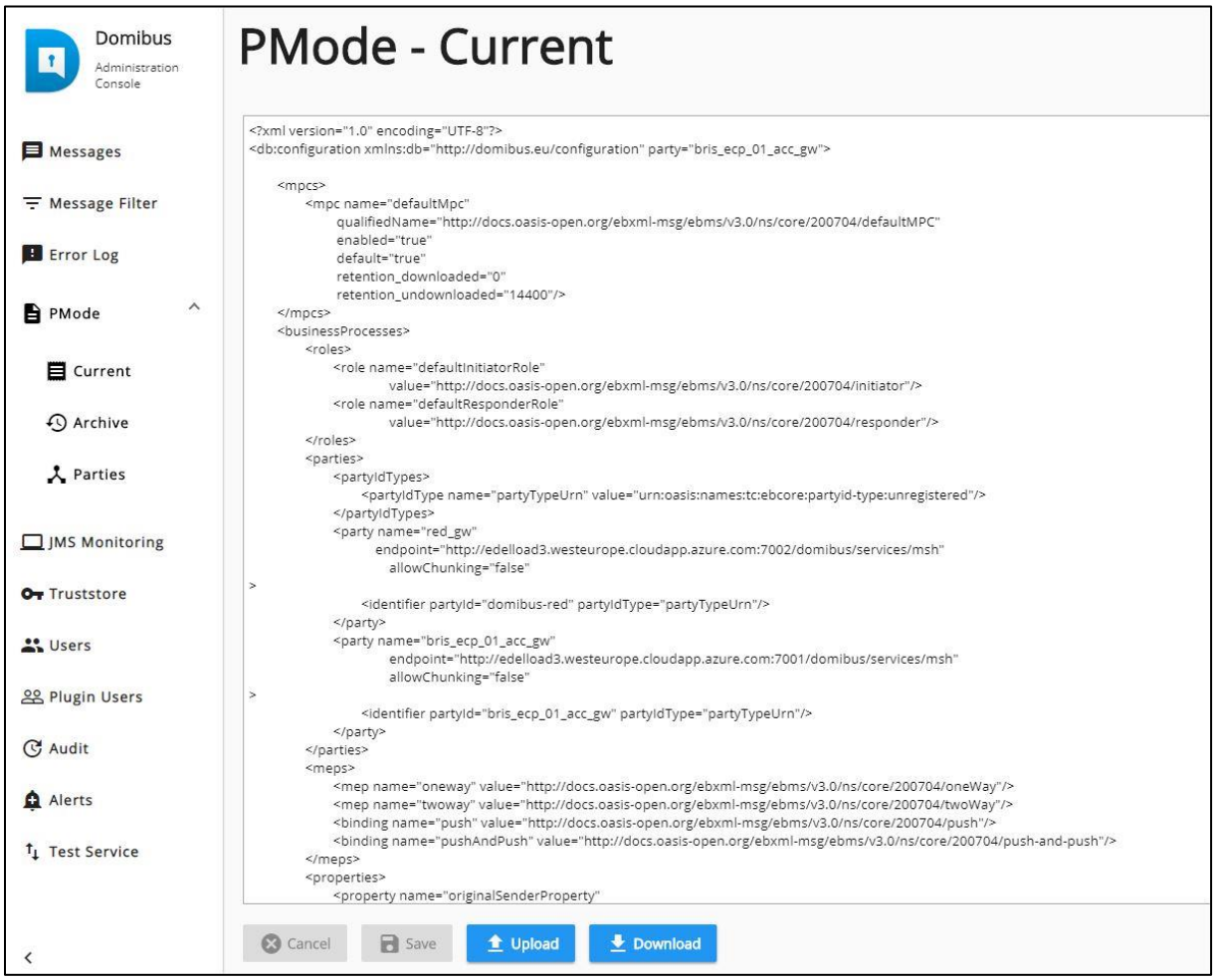
3.6. PMode configuration

Using the sample PMode file provided with the Domibus software:

- Edit the PMode file and remove the 2 instances of **payloadProfile="MessageProfile"** below:

```
<?xml version="1.0" encoding="UTF-8"?>
……..
……..
                <legConfigurations>
                        <legConfiguration name="pushTestcase1tc1Action"
                                        service="testService1"
                                        action="tc1Action"
                                        defaultMpc="defaultMpc"
                                        reliability="AS4Reliability"
                                        security="eDeliveryAS4Policy"
                                        receptionAwareness="receptionAwareness"
                                        propertySet="eDeliveryPropertySet"
                                        payloadProfile="MessageProfile"
                                        errorHandling="demoErrorHandling"
                                        compressPayloads="true"/>
                        <legConfiguration name="testServiceCase"
                                                service="testService"
                                                action="testAction"
                                                defaultMpc="defaultMpc"
                                                reliability="AS4Reliability"
                                                security="eDeliveryAS4Policy"
                                                receptionAwareness="receptionAwareness"
                                                propertySet="eDeliveryPropertySet"
                                                payloadProfile="MessageProfile"
                                                errorHandling="demoErrorHandling"
                                                compressPayloads="true"/>
                </legConfigurations>
<process name="tc1Process"
……..
……..
……..
```

- Upload the **PMode** to the Access point using the PMode Option in the admin console:



4. SENDING PROCEDURE OPTIONS

In this section we will demonstrate, using examples, how messages Payloads are sent from one Access Point to another.

If FS-plugin is used for the sending as well as the receiving:

1. Place the **metadata.xml** file under the sender's **filestore/OUT** folder.
2. Any subsequent file(s) placed in the same **filestore/OUT** folder will automatically be sent to the receiver's Access Point and placed under the receiver's **filestore/IN** folder.

If WS-plugin is used to send a message to an FS-plugin access point, then:

1. The metadata.xml file is not used at the sender side, instead you can use SoapUI or another client backend application.
2. Sent payloads will be also placed under the receiver's **filestore/IN** folder.

Remark:
The next sections describe the structure of the receiver's filestore /IN folder which contains received messages.

4.1. Scenario 1: WS-plugin is used at the sender's side

If WS-plugin is used at the sender's side, we will have two configuration choices:

4.1.1. Using PayloadName

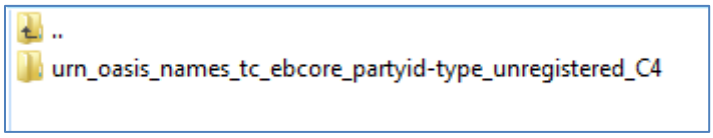
1. If the **PayloadName Property_name** is used as shown below, where SoapUI used as Backend, the received payload name will be the value given to the PayloadName parameter (**testing1.xml**):

```
...
          <ns:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C4</ns:Property>
        </ns:MessageProperties>
        <ns:PayloadInfo>
          <ns:PartInfo href="cid:message">
            <ns:PartProperties>
              <ns:Property name="MimeType">text/xml</ns:Property>
              <ns:Property name="PayloadName">testing1.xml</ns:Property>
            </ns:PartProperties>
          </ns:PartInfo>
        </ns:PayloadInfo>
      </ns:UserMessage>
...
```

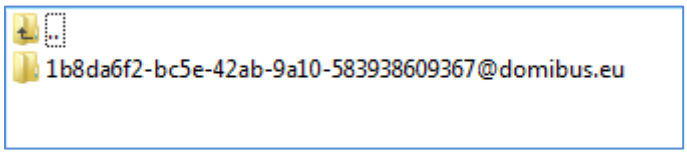
2. The SoapUI response is shown here:

```
<messageID>1b8da6f2-bc5e-42ab-9a10-583938609367@domibus.eu</messageID>
```

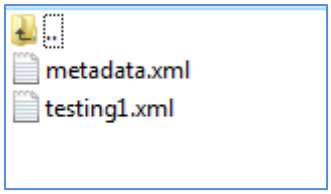
3. At the Receiver's IN folder, a first subfolder is created and named using the **finalRecipient** parameter value from the Sender's metadata.xml file (e.g: ***urn_oasis_names_tc_ebcore_partyid-type_unregistered_C4***). Note that the colon ":" in the folder name has been replaced by the underscore (_) character:



4. Within the **finalRecipient** folder, is then created a second subfolder named using the **MessageID** (e.g: 1b8da6f2-bc5e-42ab-9a10-583938609367@domibus.eu in this case)



5. Finally, the payload (**testing1.xml**) and the metadata.xml file are stored in the second folder, the **MessageID** folder (1b8da6f2-bc5e-42ab-9a10-583938609367@domibus.eu) as shown here:



4.1.2. *PayloadName not used option*

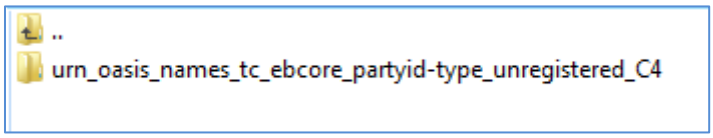
1. The received payload name will be the value given after href="cid:**xxxxx**" as shown (e.g:**message.xml**) or a random string if href is absent (e.g.: 476cb7ab-df21-456f-bb6e-619e1d6fb1ea.xml):

```
…<ns:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C4</ns:Property>
        </ns:MessageProperties>
        <ns:PayloadInfo>
          <ns:PartInfo href="cid:message">
            <ns:PartProperties>
              <ns:Property name="MimeType">text/xml</ns:Property>
            </ns:PartProperties>
          </ns:PartInfo>
        </ns:PayloadInfo>
      </ns:UserMessage>….
```

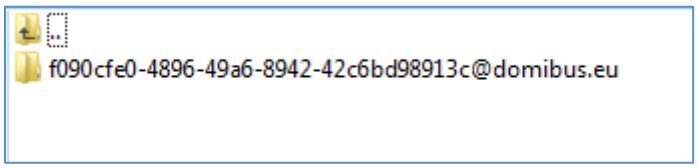
2. The SoapUI response is shown here:

```
<messageID>f090cfe0-4896-49a6-8942-42c6bd98913c@domibus.eu</messageID>
```

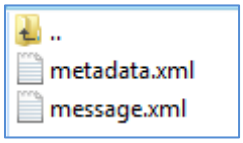
3. At the Receiver's IN folder, a first subfolder is created and named using the **finalRecipient** parameter value (e.g: ***urn_oasis_names_tc_ebcore_partyid-type_unregistered_C4***). Note that the colon ":" in the folder name has been replaced by the underscore (_) character:



4. Within the **finalRecipient** folder a second subfolder is created, with a name using the current **MessageID** value (e.g: f090cfe0-4896-49a6-8942-42c6bd98913c@domibus.eu in this case)



5. Finally, the payload (message.txt) and the metadata.xml file are stored in the second **MessageID** folder (f090cfe0-4896-49a6-8942-42c6bd98913c@domibus.eu) as shown here:

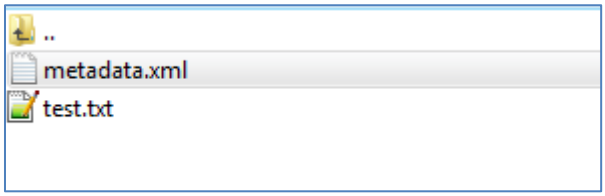


4.2. Scenario 2: FS-plugin is used at the sender's side

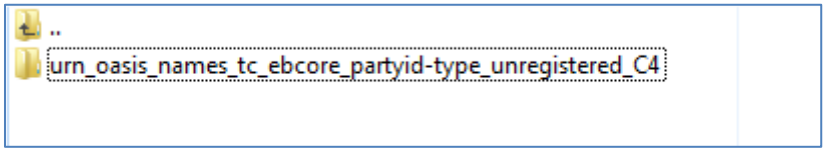
1. With the **finalRecipient** parameter, in metadata.xml, set to "urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4":

```
<MessageProperties>

    <Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</Property>

    <Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4</Property>

  </MessageProperties>

</UserMessage>
```

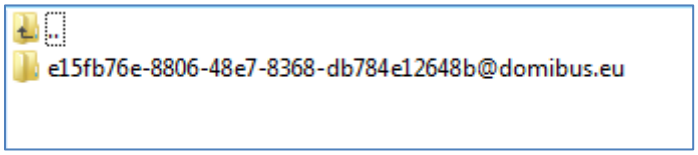
2. And a payload (e.g: test.txt) is placed in the Sender's OUT folder, to be sent, as shown here:



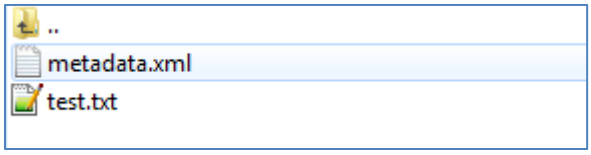
3. At the Receiver's OUT folder, a first subfolder named using the **finalRecipient** parameter value is created (e.g: ***urn_oasis_names_tc_ebcore_partyid-type_unregistered_C4***). Note that the colon ":" in the folder name has been replaced by the underscore (_) character:



4. Within the **finalRecipient** folder is then created a second subfolder named using the current **MessageID** value (e.g: e15fb76e-8806-48e7-8368-db784e12648b@domibus.eu in this case)



5. Finally, the payload (test.txt) and the metadata.xml file copied from the Sender's OUT folder are stored in the second folder, the **MessageID** folder (e15fb76e-8806-48e7-8368-db784e12648b@domibus.eu) as shown here:



Remark:

*Using the **finalRecipient** value from the Sender's metadata.xml, the name of Receiver's finalRecipient folder will be modified is such a way that:*

- *All characters that are not alphanumeric characters[A-Za-z0-9], not dots (.) and not dashes (-) will be replaced by the underscore character (_)*
- *e.g: urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4 will be changed to: urn_oasis_names_tc_ebcore_partyid-type_unregistered_C4*

5. FINE TUNING

5.1. Delete/Archive Message

The **fs-plugin.properties** can be setup to either delete sent or failed messages or archive them in the sent or failed folders under filestore.

5.1.1. Delete

- ***fsplugin.messages.sent.action=delete****: message Deleted after being sent*
- ***fsplugin.messages.failed.action=delete****: message Deleted after send failure*

5.1.2. Archive

- ***fsplugin.messages.failed.action=archive****: message Archived in the FAILED folder after failing to be sent*
- ***fsplugin.messages.sent.action=archive****: message Archived in the SENT folder after being sent*

5.1.3. Purge

- ***fsplugin.messages.received.purge.expired=600:*** *will trigger a quartz job that will delete the entire message_id folder under IN/FINAL_RECIPIENT*
- ***fsplugin.messages.sent.purge.expired=*** *will keep the message in the SENT folder indefinitely (the* ***fsplugin.messages.sent.action=archive*** *must be set first)*

5.2. Specific Domain Setup

5.2.1. fs-plugin.properties Domain Configuration changes

In order to configure multiple locations for incoming and outgoing files to be sent with the Domibus file plugin, multiple domains can be setup in order to store messages in Domain specific locations. The following additional steps have to be performed for the purpose:

1. Create a ***SOMEDOMAIN*** folder, before (re)starting Domibus where ***SOMEDOMAIN*** is a user defined name. You can have as many ***SOMEDOMAIN*** folders as needed (***SOMEDOMAIN1***, ***SOMEDOMAIN2*** etc...) for your multiple locations purpose.

2. Add the following details to the **fs-plugin.properties** files (see the below highlighted sections in Yellow):

    - The new Domain expression which includes the domain name and location (e.g: **SOMEDOMAIN**), the Domain Service (e.g: **SomeDomainService**) and the Domain Action (e.g: **SomeDomaineAction**).

```
.
# E.g.: 1
#fsplugin.domains.DOMAIN1.order=1

# Regular expression used to match the domain for the reception of messages. This regular expression will be evaluated
# against the Service and Action values from the incoming message separated by #.
# E.g.: DOMAIN1SampleService#.*
#fsplugin.domains.DOMAIN1.messages.expression=
fsplugin.domains.SOMEDOMAIN.messages.expression=SomeDomainService#SomeDomainAction

# The location of the folder that the plugin will use to manage the messages to be sent and received in case no domain
# expression matches. This location must be accessible to the Domibus instance. The domain locations must be independent
# from each other and should not overlap.
# E.g.: /home/domibus/fs_plugin_data/DOMAIN1
#fsplugin.domains.DOMAIN1.messages.location=
fsplugin.domains.SOMEDOMAIN.messages.location=/PATH/SOMEDOMAIN

# The user used to access the domain location specified by the property fsplugin.domains.<domain_id>.messages.location.
# This value must be provided if the location access is secured at the file system level so that users from other
.
```

3. *Optional:* Set the **fsplugin.*domains.SOMEDOMAIN*.messages.payload.id** property (Default is ***cid:message***)

Remark

- *Create the **SOMEDOMAIN** folder before (re)starting Domibus.*
- ***IN**, **OUT**, **SENT** and **FAILED** folders will be automatically created under **SOMEDOMAIN**.*

The following properties defined in the property file *edelivery_path* /conf/domibus/plugins/conf/**fs-plugin.properties** can be used to configure the FS-Plugin for a particular domain:

5.2.2. Metadata.xml Domain Configuration changes

Insert an additional Service and Action to the **metadata.xml,** as shown in the highlighted section below (in Yellow):

```
<?xml version="1.0" encoding="UTF-8" ?>
<UserMessage xmlns="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/">
  <PartyInfo>
    <From>
      <PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-blue</PartyId>
      <Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</Role>
    </From>
```

```
      <!--Optional:-->
      <To>
         <PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-red</PartyId>
         <Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder</Role>
      </To>
   </PartyInfo>
   <CollaborationInfo>
      <!--You may enter the following 4 items in any order-->
      <!--Optional:-->
      <!--<AgreementRef type="">A1</AgreementRef-->
      <Service type="tc1">SomeDomainService</Service>
      <Action>SomeDomainAction</Action>
   </CollaborationInfo>
   <MessageProperties>
      <!--1 or more repetitions:-->
      <!--originalSender and finalRecipient are mandatory-->
      <Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</Property>
      <Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C4</Property>
   </MessageProperties>
</UserMessage>
```

5.2.3. PMode Domain Configuration changes

Insert the following additions (see highlighted section below in Yellow) to the existing Pmode:

Remark:

- ***SomeDomainService, SomeDomainAction*** *and* ***SOMEDOMAIN*** *are user defined names.*
- ***SomeDomainService*** *describes the service specific to the new Domain.*
- ***SomeDomainAction*** *describes the action that the new domain service can perform.*
- ***SOMEDOMAIN*** *is the name of the subfolder of the File System Plugin folder where messages for the new domain will be stored.*

```
<?xml version="1.0" encoding="UTF-8"?>
        ……..
        ……..
        ……..
                    <services>
                        <service name="testService1" value="bdx:noprocess" type="tc1"/>
                        <service name="SomeDomainService" value="SomeDomainService"
type="tc1"/>
                    </services>
                    <actions>
                        <action name="tc1Action" value="TC1Leg1"/>
                        <action name="tc2Action" value="TC2Leg1"/>
                        <action name="SomeDomainAction" value="SomeDomainAction"/>
```

```
                    </actions>

        ……..
        ……..
        ……..
                    <legConfiguration name="pushTestcase1tc3Action"
                                      service="SomeDomainService"
                                      action="SomeDomainAction"
                                      defaultMpc="defaultMpc"
                                      reliability="AS4Reliability"
                                      security="eDeliveryPolicy"
receptionAwareness="receptionAwareness"
errorHandling="demoErrorHandling"
                                      compressPayloads="true"/>
                    </legConfigurations>
        <process name="tc1Process"
            agreement=""
            mep="oneway"
            binding="push"
            initiatorRole="defaultInitiatorRole"
            responderRole="defaultResponderRole">
                <initiatorParties>
                        <initiatorParty name="blue_gw"/>
                        <initiatorParty name="red_gw"/>
                </initiatorParties>
                <responderParties>
                        <responderParty name="blue_gw"/>
                        <responderParty name="red_gw"/>
                </responderParties>
                <legs>
                        <leg name="pushTestcase1tc1Action"/>
                        <leg name="pushTestcase1tc2Action"/>
                        <leg name="pushTestcase1tc3Action"/>
                </legs>
        </process>
                    </businessProcesses>
        </db:configuration>
```

Once the PMode are loaded in the sender and receiver Access points, the user will be able to drop the Payload into the /PATH/**SOMEDOMAIN/OUT** folder. The payload will be received in the receiver's /PATH/**SOMEDOMAIN/IN** folder.

5.2.4. Multitenancy

The FS Plugin can be used when Domibus is configured in Multitenancy mode.

In Multitenancy mode the plugins security is activated by default, regardless of the configured value in domibus.properties for the domibus.auth.unsecureLoginAllowed property. As a result, every request to Domibus to send a file must be authenticated via plugin username and password, which are configured in fs-plugin.properties per domain.

Please find below a configuration example for domain DOMAIN1:

```
# Mandatory in Multitenancy mode. The user that submits messages to Domibus. It is used to associate the current user
# with a specific domain.
fsplugin.domains.DOMAIN1.authentication.user=

# Mandatory in Multitenancy mode. The credentials of the user defined under the property username.
fsplugin.domains.DOMAIN1.authentication.password=
```

More information on how to create plugin users used for authentication can be found in the **Domibus Administration Guide** (see [REF1]), section **Plugin Users.**

5.2.5. Different recipients

To send multiple files to different recipients, create different sub-folders under the **Sender's OUT folder,** for each recipient. (One OUT/sub-folder per recipient).

Each of the Sub-folders would contain a metradata.xml file that contains details of its corresponding recipient.

The FS Plugin would then scan each of these sub-folders and send the message present in each using the metadata.xml details.

Remark:

To put the message to be distributed in each of the sub-folders, either create a script which copies the message to each sub-folder, using a pre-prepared list of recipients or develop a client program that does a similar task.

6. File System Permissions

Domibus must be able to write the incoming messages to the specified **IN folder at the receiver end**, whether a domain setup is used or not.

6.1. Local File Systems:

For **local** file systems, the **Domibus user** must have the necessary **write permissions** to the **IN, OUT, SENT and FAILED folders**.

6.2. Remote File Systems:

For remote file systems like SMB (Server Message Block), SFTP or FTP protocols**,** where the location access is secured at the file system level and where users from other domains cannot access its contents, the **fs-plugin.properties** file must include the credentials of a user (or users) allowed to access the **IN, OUT, SENT and the FAILED folders** specific to each domain.

The fields to setup include are described in the following example. See Section 3.1 of the Domibus Interface Control Document File System Plugin (see [REF1]) for more details:

```
……
fsplugin.domains.DOMAIN1.messages.user= user_name
fsplugin.domains.DOMAIN1.messages.password=user_password
…….
```

***Remark**:*

A typical error that would occur if the sent Payload cannot be written to the receiver's IN folder is described here:

Example:

```
2017-10-18 15:25:08,772 [] [] ERROR e.d.p.f.w.FSSendMessagesService:66 - Error setting up folders for domain: SOMEDOMAIN
eu.domibus.plugin.fs.exception.FSSetUpException: IO error setting up folders
    at eu.domibus.plugin.fs.FSFilesManager.getEnsureChildFolder(FSFilesManager.java:104)
    at
```

*In this situation, the message is moved to the **sender's SENT** folder, even though it has not actually been successfully delivered to the **receiver's IN** folder.*
*The message will only be released to the **receiver's IN** tray when the correct (e.g.: **write**) permissions are set.*

7. CONTACT INFORMATION

eDelivery Support Team

By email: EC-EDELIVERY-SUPPORT@ec.europa.eu

Support Service: 8am to 6pm (Normal EC working Days)