EUROPEAN COMMISSION

DIGIT
Connecting Europe Facility

Domibus

Interface Control Document

File System Plugin

Version [2.0]

Status [Final]

Date: 31/07/2019

Document Approver(s):

Approver Name	Role
Adrien FERIAL	Project Manager

Document Reviewers:

Reviewer Name	Role
Yves ADAM	Quality Manager
Marcio SAMPAIO	Project Manager
Cosmin BACIU	Architect

Summary of Changes:

Version	Date	Created by	Short Description of Changes
0.01	12/09/2017	Henrique FERNANDES Bruno GONÇALVES	Initial version
0.02	25/09/2017	Henrique FERNANDES Bruno GONÇALVES	Document updated based on revision
0.03	04/10/2017	Henrique FERNANDES Bruno GONÇALVES	Document updated based on revision
0.04	13/10/2017	Henrique FERNANDES Bruno GONÇALVES	Document updated based on revision
0.05	30/10/2017	Henrique FERNANDES Bruno GONÇALVES	Document updated based on revision
1.00	31/10/2017	Yves ADAM	Small changes and final version.
1.10	20/03/2018	CEF Support	Reuse notice added; e-SENS AS4 profile replaced by eDelivery AS4 profile
1.11	23/07/2018	Tiago MIGUEL	Security Modifications for Multi-Tenancy and example_metadata.xml references
1.12	25/07/2018	Tiago MIGUEL	Update FS Plugin Properties
1.13	07/09/2018	Cosmin BACIU	Updates for Multi-tenancy
1.14	17/09/2018	Caroline AEBY	Multi-tenancy => Multitenancy
1.15	26/09/2018	Caroline AEBY	No more standby service
1.16	08/02/2019	Caroline AEBY	Update for Domibus 4.0.2
1.17	24/04/2019	Ioana DRAGUSANU Caroline AEBY	Update for Domibus 4.1: UserMessage/mpc attribute
1.18	10/05/2019	Ion Perpegel Caroline AEBY	Update for Domibus 4.1
1.19	13/05/2019	Catalin ENACHE	Domibus 4.1-RC1: added Send Queue
2.0	16/07/2019	Ion Perpegel	FS plugin general properties: updates

Table of Contents

1. INTRODUCTION

1.1. Purpose

The purpose of this document is to outline the file system messages exchange as part of the default File System (FS) backend integration solution for the Domibus Access Point[i].

This document will univocally define the FS Plugin that acts as an interface to the Access Point (Corner Two and Corner Three in the four corner topology that will be explained later in this document) component of the CEF eDelivery building block.

There is 1 interface described in this document:

Interface	Description	Version
FS backend integration	The FS Plugin	3.x.y

1.2. Scope

This document covers the service interface of the Access Point from the perspective of the File System backend integration. It includes information regarding the description of:

- FS Plugin Configuration
- Supported File Systems
- Folders Structure
- Send and Receive Workflow

This specification addresses no more than the file system interface of the Access Point FS Plugin. All other aspects of its implementation are not covered by this document.

1.3. Audience

This document is intended to:

The Directorate Generals and Services of the European Commission, Member States (MS) and also companies of the private sector wanting to set up a connection between their backend system and the Access Point. In particular:

- Architects will find it useful for determining how to best use the File System Plugin to create a fully-fledged solution and as a starting point for connecting a Back-Office system to the Access Point.

- Analysts will find it useful to understand the File System Plugin that will enable them to have a holistic and detailed view of the operations and data involved in the use cases.
- Developers will find it essential as a basis of their development concerning the File System Plugin interface.
- Testers can use this document in order to test the interface by following the use cases described.

1.4. References

The table below provides the reader with the list of reference documents.

#	Document	Contents
[REF1]	Introduction to the Connecting Europe Facility - eDelivery building block	Overview of eDelivery
[REF2]	Business Document Metadata Service Location - Software Architecture Document	This document is the Software Architecture document of the CIPA eDelivery Business Document Metadata Service Location application (BDMSL) sample implementation. It intends to provide detailed information about the project: 1) An overview of the solution 2) The different layers 3) The principles governing its software architecture.
[REF3]	ebXML (Electronic Business using eXtensible Markup Language)	ebXML (Electronic Business using eXtensible Markup Language)
[REF4]	Web Services Description Language (WSDL) 1.1	Web Services Description Language (WSDL) 1.1 WS-I Basic Profile Version 1.1
[REF5]	XML Schema 1.1	XML Schema 1.1
[REF6]	Extensible Markup Language (XML) 1.0	Extensible Markup Language (XML) 1.0
[REF7]	Hypertext Transfer Protocol 1.1	Hypertext Transfer Protocol 1.1

[REF8]	SOAP Messages with Attachments	SOAP Messages with Attachments
[REF9]	AS4 Profile of ebMS 3.0 Version 1.0	AS4 Profile of ebMS 3.0 Version 1.0
[REF10]	eDelivery AS4 profile	https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eDelivery+AS4
[REF11]	eDelivery - PMode Configuration	eDelivery – PMode Configuration (will be available at a later stage)
[REF12]	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/	XSDs for ebms3
[REF13]	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/cs02/ebms_core-3.0-spec-cs-02.pdf	ebXML (Electronic Business using eXtensible Markup Language)
[REF14]	Apache VFS - Supported File Systems	Description of Apache VFS supported file systems capabilities, URI formats and examples.
[REF15]	Interface Control Document of the default WS plugin	Description of WS Plugin interface and message state transitions
[REF16]	https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Domibus	Domibus documentation.

1.5. Acronyms

The table below provides the reader with the list of acronyms definitions.

Acronym	Definition
FS Plugin	File System Plugin
ebMS	ebXML Messaging Service Specification
MEP	Message Exchange Pattern A Message Exchange Pattern describes the pattern of messages required by a communications protocol to establish or use a communication channel.
ebXML	Electronic Business XML Project to use XML to standardise the secure exchange of business data.
PMode	Processing Mode
MSH	Message Service Handler The MSH is an entity that is able to generate or process messages that conform to the ebMS specification, and which act in at least one of the two ebMS roles: Sender and Receiver. In terms of SOAP processing, an MSH is either a SOAP processor or a chain of SOAP processors. In either case, an MSH has to be able to understand the eb:Messaging header (qualified with the ebMS namespace).
VFS	Virtual File System It presents a uniform view of the files from various different sources, such as the files on local disk or remote shares.
UNC	Universal Naming Convention

2. INTERFACE FUNCTIONAL SPECIFICATION

In order to understand the Use Cases that will be described below it is important to explain the topology; i.e. the four – corner model.

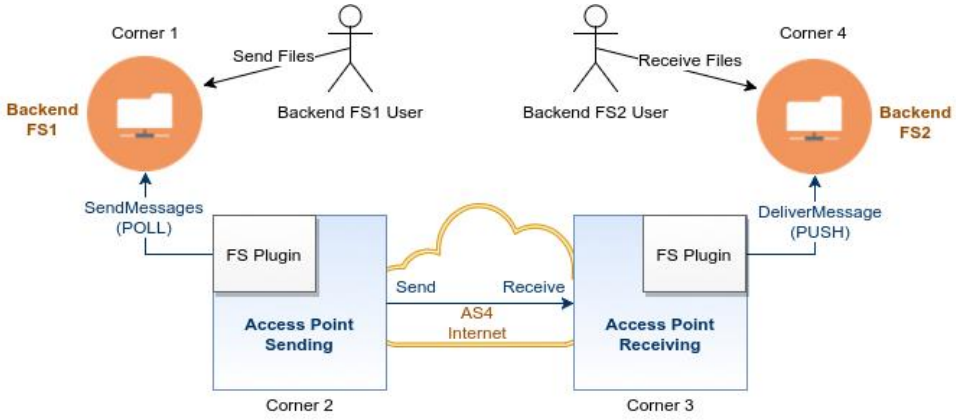

Figure 1 - the four corner model

In this model we have the following elements:

- Corner 1 (FS1): Backend FS1 is the file system that will be polled by the sending AP (Access Point) FS Plugin
- Corner 2 (C2): Sending Access Point C2
- Corner 3 (C3): Receiving Access Point C3
- Corner 4 (C4): Backend C4 is the file system where the Access Point will save the received payloads

There are two backend adapters (i.e. corner one and corner four). Corner 1 sends messages do Corner 2 and Corner 4 receives messages from Corner 3. Corners 2 and 3 communicate with each other via the information contained in the PMode configuration files. The FS backend is described in this document, it provides the facility to send and receive messages as files in the backends file system.

The ICD specification provides both the provider (i.e. the implementer) of the services and their consumers with a complete specification of the following aspects:

- Detailed use cases, which specifies the set of use cases available in the file system interface;
- Interface Behavioural Specification, which specifies the expected sequence of steps when interacting with the file system interface;

2.1. Key definitions

Access Point

According to CEF eDelivery, an Access Point is an implementation of the OpenPEPPOL AS2 Profile or the eDelivery AS4 Profile. The data exchange protocols of CEF eDelivery are profiles, meaning that several options of the original technical specifications were narrowed down in order to increase consistency, interoperability and to simplify deployment. The profile of AS2 was developed by OpenPEPPOL6, and the profile of AS4 was developed by e-SENS in collaboration with several service providers while being implemented in the e-Justice domain by e-CODEX. An Access Point exposes two interfaces:

- An interface to connect the Backend system with the Access Point. Typically, this interface is customisable as communication between Access Points and Backend systems may use any messaging or transport protocol.
- A standard messaging interface between Access Points, this interface is configurable according to the options of the profiles supported by CEF eDelivery. It is important to note that CEF eDelivery standardises the communication only between the Access Points.

eDelivery AS4 Profile

The eDelivery AS4 profile is an open technical specification for the secure and payload-agnostic exchange of data using Web Services. According to OASIS, the AS4 protocol is the modern successor of the AS2 protocol. AS4 itself is a profile of the ebXML Messaging Services (ebMS) v.3.0, a broader specification built on top of the SOAP with attachments specification.

2.2. Purpose of the File System Plugin

The purpose of the File System Plugin is to allow the Backends (corner 1 and corner 4) to exchange messages as files simply using the file system interface without the need of accessing web services.

2.3. Use case overview

2.3.1. Actors

Actor	Description
FS Plugin Admin	Any participant with administrative rights to configure the FS Plugin behaviour and access to the backend file system.
Backend FS1 User	Any participant (human or system) sending file messages to a recipient Backend C4 and using the Sending AP C2 in that purpose via the FS Plugin.
Backend FS2 User	Any participant (human or system) downloading file messages from any sender Backend C1 and using the Receiving AP C3 in that purpose via the

	FS Plugin.

Table 1 - List of Actors

2.3.2. Use cases diagram

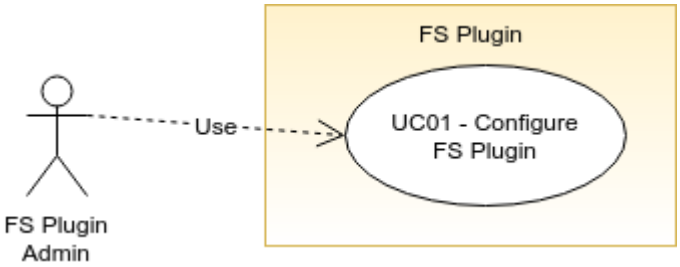

Figure 2 - FS Plugin Use cases diagram

ID	UC	Short Description	System
UC01	Configure FS Plugin	Configure the available properties for the FS Plugin file system access and behaviour	Backend File System 1 Backend File System 2

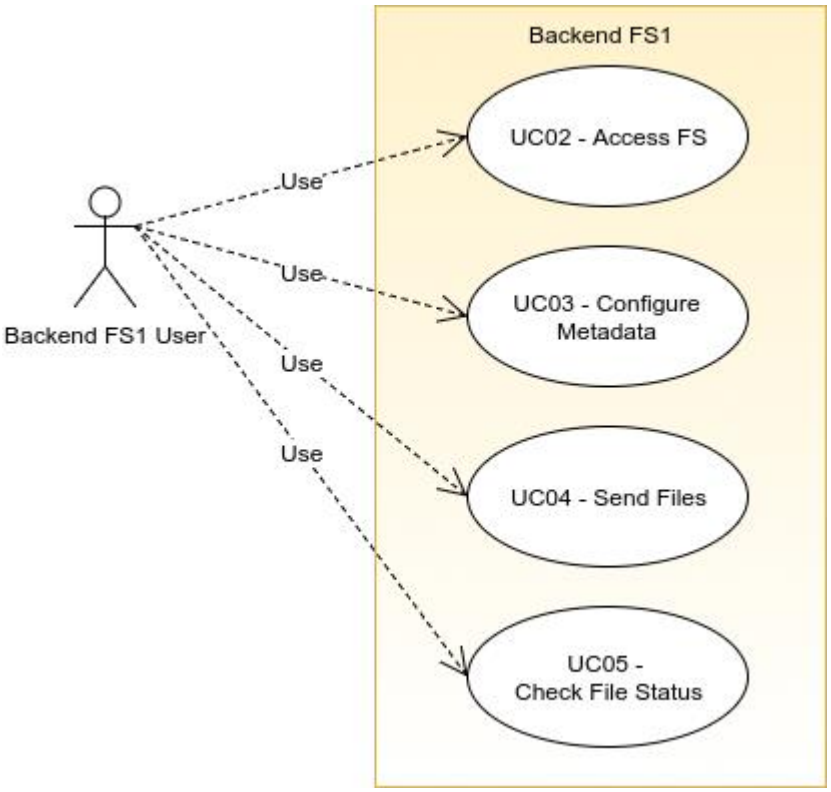

Figure 3 - Backend FS 1 Use cases diagram

ID	UC	Short Description	System
UC02	Access FS	Access or connect to the file system interface (local or remote). This operation may require authentication if the file system is secured. For more information see section 3.2 - Supported File Systems.	Backend File System 1
UC03	Configure Metadata	Configure the metadata.xml file in order to define the file messages submission properties (e.g., From PartyId, To PartyId).	Backend File System 1
UC04	Send Files	Drop the files in the backend file system OUT folder in order to trigger the sending process according to the existing metadata. The files are polled by the FS Plugin of C2.	Backend File System 1
UC05	Check File Status	Check the sending process status listing the file name that will be updated according to the messageId and status.	Backend File System 1

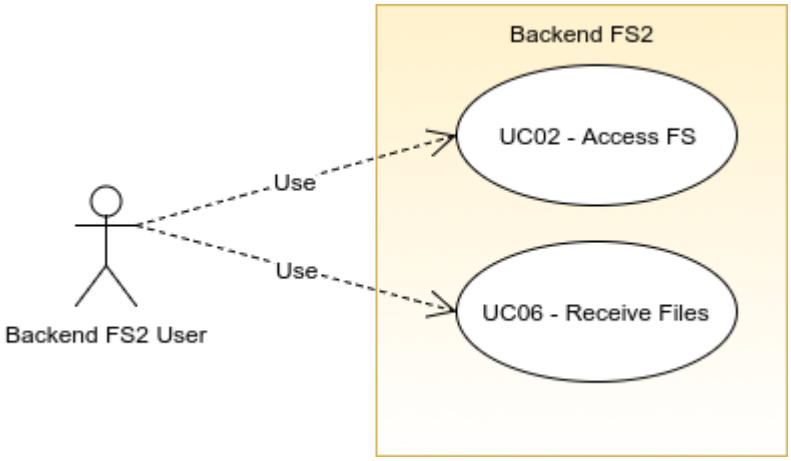

Figure 4 - Backend FS 2 Use cases diagram

ID	UC	Short Description	System
UC02	Access FS	Access or connect to the file system interface (local or remote). This operation may require authentication.	Backend File System 2
UC06	Receive Files	Receive the sent files in the configured backend file system location. These file messages are PUSHED by the FS Plugin into the receiving backend file system.	Backend File System 2

2.4. Detailed uses cases

The following paragraphs define the use cases listed above with more details.

The Interface Functional Specification is described in the detailed uses cases using the given examples. It is important to note that the Inputs and Outputs provided as examples for the uses cases are based on a specific PMode configuration.

As defined in the AS4 Profile of ebMS 3.0, a processing mode – or PMode – is a collection of parameters defining how user messages are exchanged between a pair of Access Points with respect to Quality of Service, Transmission Mode and Error Handling. A PMode maps the recipient Access Point from the partyId, which represents the backend offices associated to this Access Point.

2.4.1. UC01 - Configure FS Plugin

Brief description

The FS Plugin Admin configures the plugin behaviour by editing the fs-plugin.properties file.

Actors

FS Plugin Admin

Preconditions

- The FS Plugin must be installed on the Access Point.
- The system user (FS Plugin Admin) as read/write access to the fs-plugin.properties file.

Basic flow event

1. The system user (FS Plugin Admin) edits the fs-plugin.properties file.
2. The system user (FS Plugin Admin) updates the existing properties according to the full specification of configuration options is listed in Sections 3.1 & 3.2.

Alternative flows

N/A

Exception flows

2a. The system user (FS Plugin Admin) creates or updates a FS Plugin property with an invalid value.

2a1. The FS Plugin will decay the invalid configuration into the property default value as listed in section 3.1.

Post conditions

Successful conditions: The FS Plugin is successfully configured. After the Access Point restart, it is ready to exchange messages from the file system interface.

Failure conditions: The FS Plugin is not correctly configured and is not ready to properly send and receive files as AS4 messages.

2.4.2. UC02 - Access FS

Brief description

To use the file system interface, the Backend FS1 User needs to access the Access Point - FS Plugin via the configured file system folder within a domain. This use case is realized by the Backend FS1 User by the navigating into the configured folder (locally or remotely). If the file system domain location is secured then the user must provide its credentials to connect.

For detailed information about Domains and Supported File Systems please see chapters 3.1.3, 3.1.4 and 3.2.

Actors

Backend FS1 User

Preconditions

- The Backend FS1 User must have read/write access on its domain folder and subfolders.

Basic flow event

1. The Backend FS1 User navigates into the configured outgoing/incoming folder. This can be done using a graphical explorer or via command line.

Alternative flows

1a. If the folder is remote and secured by username and password then the credentials will be required and prompted in the first access.

1a1. The Backend FS1 User specifies his valid username and password.

Exception flows

1a2. The Backend FS1 User specifies invalid credentials.

Post conditions

Successful conditions: The Backend FS1 User accesses the contents of the filesystem folder.

Failure conditions: The Backend FS1 User cannot access the contents of the filesystem folder.

2.4.3. UC03 – Configure Metadata

Brief description

The Backend FS1 User writes the metadata.xml file which contains the basis of the AS4 metadata that will be used by the FS Plugin to create the User Message (mapped to the Submission) that will be sent to Access Point. The metadata.xml file name is reserved by the FS Plugin and it needs to always be present as a child of the OUT folder or any sub-directory under the OUT folder.

The metadata file format is detailed in Section 3.3 and is available an example (example_metadata.xml) in the config/ directory.

Actors

Backend FS1 User

Preconditions

- The Backend FS1 User must have read/write access on its domain folder and subfolders.

Basic flow event

1. The Backend FS1 User navigates to the OUT folder or any of its sub-directories.
2. The Backend FS1 User creates the metadata.xml file based on a template (Cf.: 5.2 - metadata.xml example) in the outgoing folder.
3. The Backend FS1 User edits the metadata.xml file to specify the AS4 metadata, namely the UserMessage, according to the format detailed in section 3.3.

Alternative flows

N/A

Exception flows

2a. The Backend FS1 does not create the metadata.xml file in the outgoing folder.

3a. The Backend FS1 creates a metadata.xml file containing an invalid structure.

Post conditions

Successful conditions: The metadata.xml is configured and prepared to define the UserMessage header in the FS Plugin AS4 messages.

Failure conditions: The metadata.xml is not properly configured and is not prepared to define the UserMessage header in the FS Plugin AS4 messages. Either because it does not exist or has an invalid structure.

2.4.4. UC04 – Send Files

Brief description

This Use Case is triggered by the Backend FS1 User, by dropping a content file into the outgoing folder (C1) of one of the locations configured in fs-plugin.properties file. It is assumed that valid metadata has been configured via UC03 beforehand. This Use Case should result in an AS4 message delivery from C2 to C3.

On its own schedule, Domibus core eventually connects to the applicable destination gateway and delivers the message in conformance to the AS4 protocol.

For detailed information about the Send Files sequence see chapter 3.4.

Actors

Backend FS1 User

Preconditions

- The FS Plugin is properly configured and activated in the Domibus Access Point.
- The outgoing folder is prepared to send files as AS4 messages having the metadata.xml file properly created and configured.

Basic flow event

1. The Backend FS1 User navigates into the outgoing folder of C1.
2. The Backend FS1 User drops one or more content files into the outgoing folder.
3. The FS Plugin (C2) continually polls the various configured locations for a list of existing files in the outgoing folders.
1. The FS Plugin (C2) creates as AS4 message for each content file and submits to Domibus core. If the content file (i.e file1.doc) is accompanied by a corresponding .lock file (i.e. file1.doc.lock), the content file is not touched. Once the content file starts being processed, the FS Plugin creates the corresponding. lock file.
2. The FS Plugin (C2) renames the submitted content file adding the AS4 message id entifier. It also deletes the corresponding. lock file.
3. On its own schedule, Domibus core eventually connects to the applicable destination gateway and delivers the message in conformance to the AS4 protocol.

Alternative flows

4a. The FS Plugin (C2) skips to create the AS4 message because metadata.xml file is missing.

4a1. The FS Plugin (C2) logs the skipped files warning that the metata.xml is missing.

Exception flows

4b. The FS Plugin (C2) fails to create the AS4 message because metadata.xml file is invalid.

4b1. The FS Plugin (C2) logs the schema validation error.

4c. The FS Plugin (C2) fails to create the AS4 message because the content file can't be accessed.

4c1. The FS Plugin (C2) logs the file access error.

4c2. The FS Plugin (C2) marks the current message transaction for rollback (and let Domibus retry in its next polling).

Post conditions

Successful conditions: The content file is successfully submitted as an AS4 message and delivered from (C2) to (C3).

Failure conditions: The AS4 message is not successfully created and can't be submitted to Domibus core (C2). Error details are logged.

2.4.5. UC05 – Check File Status

Brief description

Backend FS1 User may periodically refresh his view of Backend FS1 so he gets an update on the status of files sent using UC04. The sending status is indicated by the names and locations of the various content files which the FS Plugin modifies according to the rules and states described in section 3.5 - Check File Status Sequence.

Actors

Backend FS1 User

Preconditions

- The content file is successfully submitted to Domibus (C2) as an AS4 message.

Basic flow event

1. The Backend FS1 User lists the existing files under the outgoing folder of C1 until the observed files are successfully or failed sent.
2. The FS Plugin (C2) renames the processed files according to the status change event for each message, see section 3.5 - Check File Status Sequence.
3. The FS Plugin (C2) receives a Send Success event and deletes or archives the processed file (according to the FS Plugin configuration).

Alternative flows

3a. The FS Plugin (C2) receives a Send Failed event and deletes or archives the processed file (according to the FS Plugin configuration).

Exception flows

2a. The FS Plugin (C2) cannot access and rename the processed file.

2a1. The FS Plugin (C2) logs the rename error.

3b. The FS Plugin (C2) cannot access and rename the successfully sent file.

3b1. The FS Plugin (C2) logs the rename error.

3a1. The FS Plugin (C2) cannot access and rename the failed sent file.

3a1b. The FS Plugin (C2) logs the rename error.

Post conditions

Successful conditions: The content file is successfully submitted as an AS4 message and it is successfully deleted or archived.

Failure conditions: The content file name is not properly renamed according to the message status change events; the event is registered in the log files.

2.4.6. UC06 – Receive Files

Brief description

Backend FS2 User initiates this Use Case by listing the files within the incoming folder of any domain configured for the FS Plugin on Backend FS2. Assuming files were successfully received and are present, the user may obtain any of them via regular file system operations.

For detailed information about the Receive Files sequence, see chapter 3.6.

Actors

Backend FS2 User

Preconditions

- The content file is successfully sent from Access Point (C2) to (C3) as an AS4 message.

Basic flow event

1. The FS Plugin downloads the received AS4 message from the Access Point (C3).
2. The FS Plugin resolves the destination domain from the AS4 metadata Service and Action values.
3. The FS Plugin creates the received AS4 message as a file in the resolved incoming folder of the Backend filesystem (C4).
4. The Backend FS2 User lists the received files under the incoming folder of C4.

Alternative flows

N/A

Exception flows

2a. If no domain is found to match the pair Service/Action, the main location is selected as the destination folder.

3a. The FS Plugin (C3) can't access and create the received message as a file.

3a1. The FS Plugin (C3) logs the access error and marks the transaction for rollback. The deliver message operation will be retried by the receiving Access Point.

Post conditions

Successful conditions: The content file is successfully received as an AS4 message.

Failure conditions: The fail message is logged by the FS Plugin.

3. INTERFACE BEHAVIOURAL SPECIFICATION

3.1. FS Plugin Configuration

The FS plugin configuration is done in the fs-plugin.properties file with the following items:

3.1.1. Workers Scheduling

These properties define the FS Plugin workers scheduling.

Property name	Default value	Description
fsplugin.messages.send.worker.repeatInterval	10000	The time interval (in milliseconds) used to poll the sending File System for new files.
fsplugin.messages.sent.purge.worker.cronExpression	0/60 * * * * ?	The cron expression used to trigger the worker to purge the sent files that were archived.
fsplugin.messages.failed.purge.worker.cronExpression	0/60 * * * * ?	The cron expression used to trigger the worker to purge the failed files that were archived.
fsplugin.messages.received.purge.worker.cronExpression	0/60 * * * * ?	The cron expression used to trigger the worker to purge the received files.

Table 2 - Workers Scheduling properties

3.1.2. General properties

The general properties described below can be overridden per domain according to the format described in the paragraph §3.1.3 - "Domain specific properties".

Property name	Default value	Description
fsplugin.messages.location	/home/domibus/fs_plugin_data/MAIN	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. See section 3.2 - "Supported File Systems".
fsplugin.messages.sent.action	delete	The file action executed when the file is successfully sent: 'delete' to permanently remove the file or 'archive' to move it into the SENT folder.
fsplugin.messages.sent.purge.expired	600	The expiration limit (expressed in seconds) used to purge the older files in the SENT folder.
fsplugin.messages.failed.action	delete	The file action executed when the file is fails to send: 'delete' to permanently remove the file or 'archive' to move it into the FAILED folder.
fsplugin.messages.failed.purge.expired	600	The expiration limit (expressed in seconds) used to purge the older files in the FAILED folder
fsplugin.messages.received.purge.expired	600	The expiration limit (expressed in seconds) used to purge the older files in the IN folder.
fsplugin.authentication.user	(none)	The user name to be used to authenticate on domains by default
fsplugin.authentication.password	(none)	The password used to authenticate on domains by default
fsplugin.messages.payload.id	cid:message	The payload identifier for messages processed on the default domain
fsplugin.send.queue	domibus.fsplugin.send.queue	This queue is used by the plugin to send the files in parallel
fsplugin.send.queue.concurrency	5-20	Specify queue concurrency limits when sending files via a "lower-upper" String, e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case)
fsplugin.messages.send.delay	2000	The delay (in milliseconds) to allow the writing process to finish writing.

Table 3 - General properties

3.1.3. Domain specific properties

The FS Plugin configuration allows to define specific properties per domain (e.g., the messages location in the file system, the file actions, etc.). The domain properties have the following convention:

`fsplugin.domains.<domain_identifier>.<property_name>=<value>`

- domain_identifier: Represents the domain identifier e.g., DOMAIN1
- property_name: The domain property name which is being defined or redefined from the general properties
- value: represents the specific domain property value

E.g.: `fsplugin.domains.DOMAIN1.messages.location=/home/domibus/fs_plugin_data/DOMAIN1`

This means that all messages belonging to DOMAIN1 will be stored in the appropriate file system location.

3.1.4. Domain resolution

In order to support multiple domains configurations, the domain resolution is done according to the values of the domain specific properties:

- `fsplugin.domains.<domain_id>.order`
  Defines the order in which the domains will be evaluated.
- `fsplugin.domains.<domain_id>.messages.expression`
  Regular expression[1] used to match the domain for the reception of messages. This regular expression will be evaluated against a concatenation of the Service and Action values from the incoming message using the character # as separator, for example:

  `BRISReceptionService#SendEmailAction`

  An expression per domain will define to which domain the message is intended for. As a convention for the PMode, it is recommended to prefix the Service with the identifier of the business domain.

  E.g.: `fsplugin.domains.BRIS.messages.expression=BRISReceptionService#.*`
  This regular expression means that all messages containing AS4 headers where the Service is set to BRISReceptionService and having any Action value will be mapped into the domain BRIS. A sample match is BRISReceptionService#WelcomeAction.

Property name	Default value	Description

[1] Java Regular Expression Syntax: https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

fsplugin.domains.<domain_id>.order	(none)	Integer defining the order in which the domains will be evaluated. This property is not mandatory – domains without order definition will be resolved randomly after the ordered domains. The order is defined with numeric values greater than 0. E.g.: 1
fsplugin.domains.<domain_id>.messages.expression	(none)	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.<domain_id>.messages.location	(none)	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. For more information about the location format and values see section 3.2 - Supported File Systems. E.g.: /home/domibus/fs_plugin_data/DOMAIN1
fsplugin.domains.<domain_id>.messages.user	(none)	The user used to access the domain location specified by 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 domains cannot access its contents. In a secured File System, e.g.: SFTP, if the credentials are not provided, the FS Plugin will not be able to access it to perform the file messages exchange. For more information see section 3.2 - Supported File Systems.
fsplugin.domains.<domain_id>.messages.password	(none)	The password used to access the domain location. This value must be provided if the location access is secured at the file system level.
fsplugin.domains.<domain_id>.authentication.user	(none)	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.<domain_id>.authentication.password	(none)	Mandatory in Multitenancy mode. The credentials of the user defined under the property username.

fsplugin.domains. .<domain_id>.messages. payload.id	cid:message	The payload identifier for messages processed on a particular domain.
fsplugin.domains.<domain_id>.send.queue.concurrency	5-20	Specify queue concurrency limits on a particular domain when sending files via a "lower-upper" String, e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case)
fsplugin.domains.<domain_id>messages.send.delay	2000	The delay (in milliseconds) on a particular domain to allow the writing process to finish writing.

Table 4 - Domain specific properties

3.2. Supported File Systems

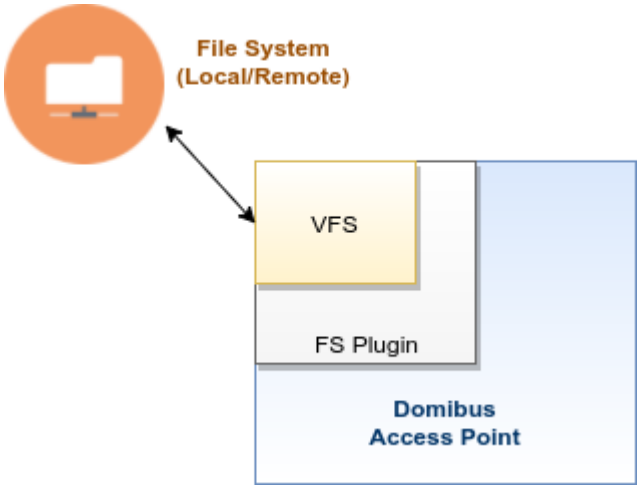

Figure 5 - FS Plugin - VFS abstraction

The FS Plugin supports multiple file system types via Apache VFS. There are 4 file systems currently supported: Local, SMB/CIFS, SFTP and FTP, detailed 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.

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.

Examples:

```
smb://somehost/shareA
smb://somehost/shareB/nesteddir
smb://otherhost:445/shareC
```

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 supports authentication via user and password domain properties. See section 3.1 – "FS Plugin Configuration".

Examples:

```
sftp://somehost/pub/downloads/
sftp://somehost:22/pub/downloads/
```

3.2.4. FTP

An FTP file system is a remote directory shared via FTP. 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 supports authentication via user and password domain properties. See section 3.1 – "FS Plugin Configuration"..

Examples:

```
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. Metadata format

Below is the metadata UserMessage schema. See an example in Annexes section 5.2– "metadata.xml example". An example of this file (example_metadata.xml) is also distributed in the config/ directory.

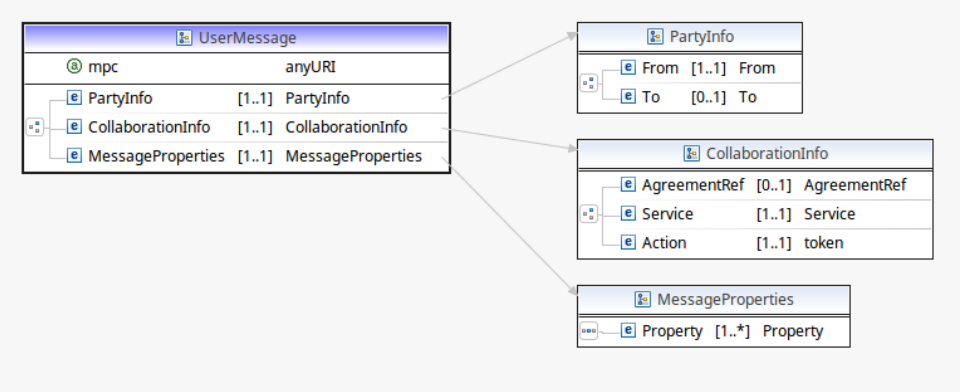

Figure 6 - UserMessage schema

Name	Description
UserMessage/mpc	This OPTIONAL attribute occurs once, and contains data about the Message Partition Channel (MPC). MPCs allow for partitioning the flow of messages from a Sending MSH to a Receiving MSH into several flows that can be controlled separately and consumed differently.
UserMessage/PartyInfo	This REQUIRED element occurs once, and contains data about originating party and destination party.
UserMessage/CollaborationInfo	This REQUIRED element occurs once, and contains elements that facilitate collaboration between parties.
UserMessage/MessageProperties	This REQUIRED element occurs once, and contains message properties that are user-specific. These properties allow for more efficient monitoring, correlating, dispatching and validating functions (even if these are out of scope of ebMS specification) which would otherwise require payload access.

3.3.1. *PartyInfo*

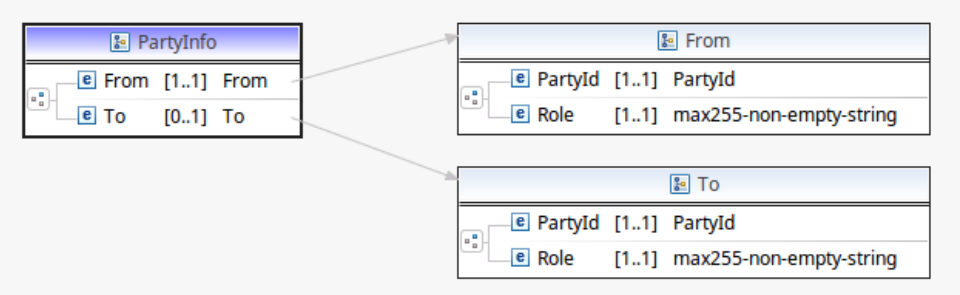

Figure 7 - PartyInfo schema

Name	Description
PartyInfo/From	The REQUIRED element occurs once, and contains information describing the originating party.
PartyInfo/From/PartyId@type	The type attribute indicates the domain of names to which the

Name	Description
	string in the content of the PartyId element belongs. E.g.: urn:oasis:names:tc:ebcore:partyid-type:unregistered
PartyInfo/From/PartyId	The REQUIRED PartyId element occurs one or more times. If it occurs multiple times, each instance MUST identify the same organization. E.g.: domibus-blue
PartyInfo/From/Role	The REQUIRED eb:Role element occurs once, and identifies the authorized role (fromAuthorizedRole or toAuthorizedRole) of the Party sending (when present as a child of the From element) or receiving (when present as a child of the To element) the message. E.g.: http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator
PartyInfo/To	The REQUIRED element occurs once, and contains information describing the destination party.
PartyInfo/To/PartyId	(Same as PartyInfo/From/PartyId)
PartyInfo/To/Role	(Same as PartyInfo/From/Role)

3.3.2. CollaborationInfo

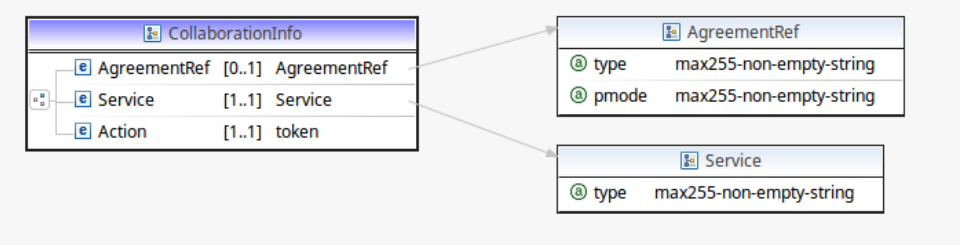

Figure 8 - CollaborationInfo schema

Name	Description
CollaborationInfo/AgreementRef	This OPTIONAL element occurs at most once. The AgreementRef element is a string that identifies the entity or artifact governing the exchange of messages between the parties. E.g.: https://joinup.ec.europa.eu/
CollaborationInfo/Service	This REQUIRED element occurs once. It is a string identifying the service that acts on the message and it is specified by the designer of the service. It SHOULD identify a set of related business transactions[2] or other message exchanges in the context of a business process or use case. E.g.: SupplierOrderProcessing

[2] "Business transactions" can be mapped into "Actions" in the AS4 context.

Name	Description
CollaborationInfo/Action	This REQUIRED element occurs once. The element is a string identifying an operation or an activity within a Service that may support several of these. It SHOULD identify the different types of business transactions or other message exchanges in the context of an identified Service. E.g.: NewOrder

3.3.3. *MessageProperties*

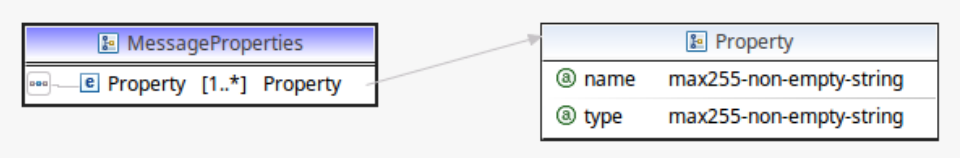

Figure 9 - MessageProperties schema

This REQUIRED element occurs at most once, and contains message properties that are implementation specific. As parts of the header such properties allow for more efficient monitoring, correlating, dispatching and validating functions (even if these are out of scope of ebMS specification) which would otherwise require payload access.

These elements hold a set of name-value properties that will hold for instance the identifiers for the 'originalSender' and 'finalRecipient'.

Name	Description
Property	Element is of xs:anySimpleType (e.g. string, URI). message properties that are implementation specific. As parts of the header such properties allow for more efficient monitoring, correlating, dispatching and validating functions (even if these are out of scope of ebMS specification) which would otherwise require payload access. These elements hold a set of name-value properties that will hold for instance the identifiers for the 'originalSender' and 'finalRecipient'. E.g.: C1
Property@name	The value of this REQUIRED attribute must be agreed upon between partners. E.g.: originalSender
Property@type	This OPTIONAL attribute allows for resolution of conflicts between properties with the same name, and may also help with Property grouping, e.g. various elements of an address.

3.4. Send Files Sequence

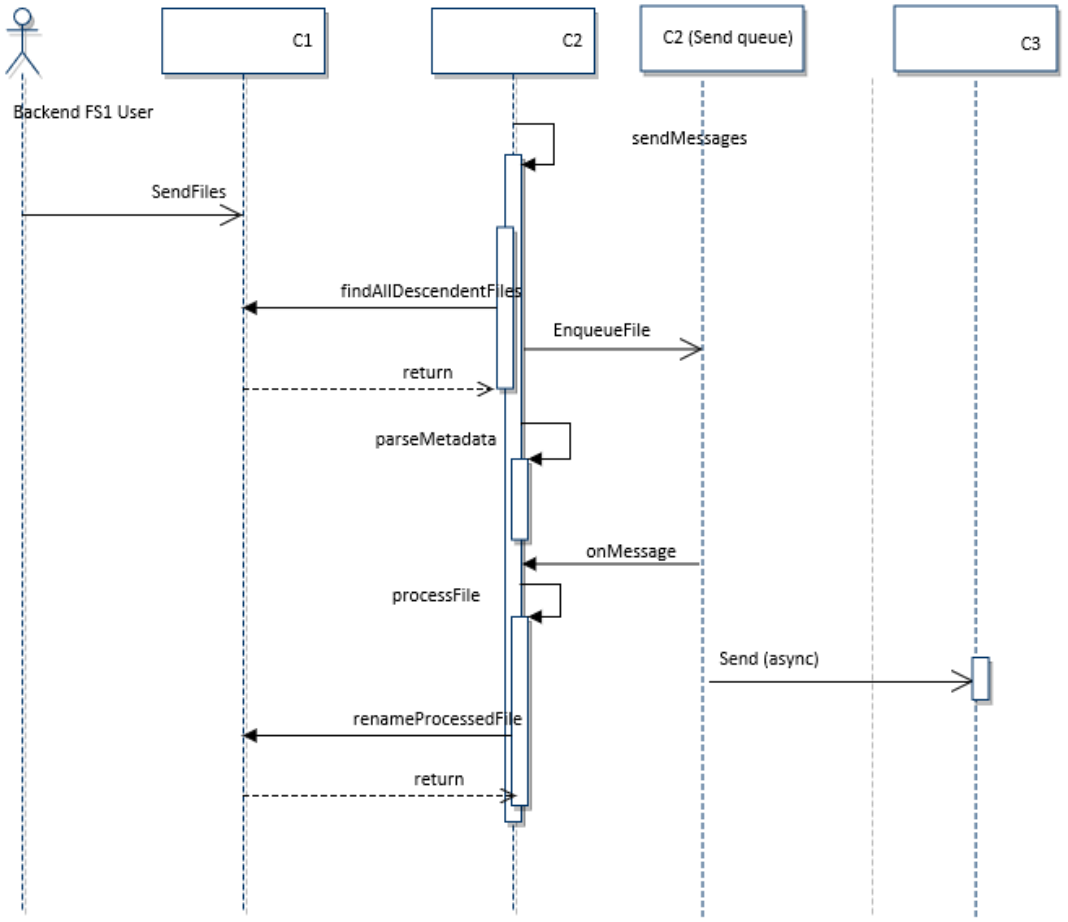

Figure 10 - SendFiles sequence diagram

The FS Plugin (C2) continually polls the various configured locations for a list of existing files in the outgoing folder and subfolders, multiple files can be found simultaneously in the same folder. It then filters that list by excluding metadata files,content files that have been processed previously (those which file name contains a message id) and files with .lock suffix to create a set of eligible content files. For each eligible content file, the plugin uses the metadata file named "metadata.xml" within the same directory than the considered content file.

For each content file, the plugin put a message to an internal queue – FSPluginSendQueue and the messages are consumed from JMS queue by several consumers – the numbers of these consumers could be changed in fs-plugin.properties , property:

```
fsplugin.send.queue.concurrency=5-20
```

The value could be changed as well per domain:

```
fsplugin.domains.DOMAIN1.send.queue.concurrency=5-20
```

The name of the queue could be changed in the property:

```
fsplugin.send.queue=domibus.fsplugin.send.queue
```

For each message in the JMS queue, the plugin then creates an AS4 message for each eligible file by retrieving information from metadata file ("metadata.xml" within the same directory) to populate the message properties and adding the content file as AS4 payload. The MIME type for the payload is derived from the content file's extension (or file content itself if the file has no extension) according to the supported formats[3]. This message is then submitted to the Domibus core and the content file is renamed by adding the message id to the original file name, as follow:

```
originalName_messageId.originalExtension
```

On its own schedule, Domibus core (C2) eventually connects to the applicable destination gateway (C3) to deliver the message.

3.5. Check File Status Sequence

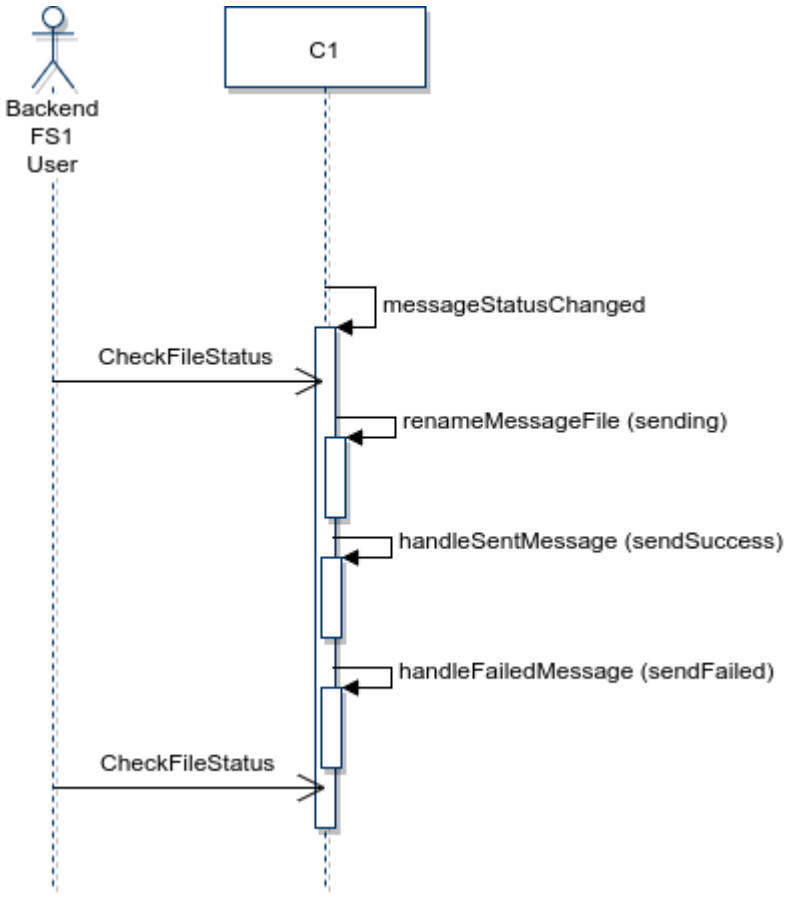

Figure 11 - Check File Status sequence diagram

Every time the Domibus core delivers a message status change event to the FS Plugin, the plugin classifies it into one of three categories, according to the new status:

- **Sending events**: set of events through which the message navigates while transfer is in progress

[3] Apache Tika list of supported formats:

http://tika.apache.org/1.16/formats.html#Supported_Document_Formats

- **Sent successfully events**: set of final events into which messages ends after a successful transfer
- **Sent failed events:** final event into which messages ends after an unsuccessful transfer

For detailed information about available states and transitions, please check the WS Plugin Interface Control Document Cf. [REF15].

3.5.1. Sending events

Corresponding set of events : {READY_TO_SEND, SEND_ENQUEUED, SEND_IN_PROGRESS, WAITING_FOR_RECEIPT, WAITING_FOR_RETRY, SEND_ATTEMPT_FAILED}

Sending events result in a rename of the content file according to the rule:

```
originalName_messageId.ext.NEW_STATUS
```

Example:

```
message1_3c5558e4-7b6d-11e7-bb31-be2e44b06b34@domibus.eu.xml.READY_TO_SEND
```

3.5.2. Sent successfully events

Corresponding set of events: {ACKNOWLEDGED, ACKNOWLEDGED_WITH_WARNING}

When a message is sent successfully, the original content file is either deleted or archived, according to the value configured in property "fsplugin.messages.sent.action" or its domain-specific variation.

When archiving, the original file is moved from the "OUT" folder to the "SENT" folder, while maintaining the same sub-folder hierarchy, e.g.:

```
<domain root>/OUT/subfolder1/subfolder2/file
```

is moved to:

```
<domain root>/SENT/subfolder1/subfolder2/file
```

When archiving, the file is also renamed according to the rule:

```
originalName_messageId.originalExtension
```

Example:

```
message1_3c5558e4-7b6d-11e7-bb31-be2e44b06b34@domibus.eu.xml
```

3.5.3. Send failed events

Corresponding set of events : {SEND_FAILURE}

When a message suffers a send failure, the original content file is either deleted or archived, according to the value configured in property "fsplugin.messages.failed.action" or its domain-specific definition. In all cases, an additional error file is created detailing the error (more details below).

When archiving, the original file is moved from the "OUT" folder to the " FAILED" folder, while maintaining the same sub-folder hierarchy, e.g.:

<domain root>/OUT/subfolder1/subfolder2/file

is moved to:

<domain root>/FAILED/subfolder1/subfolder2/file

When archiving, the file is also renamed as follows:

originalName_messageId.originalExtension

An error file is always created and placed in the "FAILED" folder while maintaining the original sub-folder hierarchy. Its name follows the form:

originalName_messageId.originalExtension.error

Example:

message1_3c5558e4-7b6d-11e7-bb31-be2e44b06b34@domibus.eu.xml.error

The error file's contents have the following format:

```
errorCode: <error_code_value>
errorDetail: <error_detail_value>
messageInErrorId: <message_id_value>
mshRole: <msh_role_value>
notified: <notified_null>
timestamp: <timestamp_value>
```

All the error values are provided to the FS Plugin by the Domibus Access Point. To see more details about the error codes and messages please check the WS Plugin Interface Control Document Cf. [REF15].

Example:

```
errorCode: EBMS:0005
errorDetail: Error dispatching message to
http://192.168.1.66:8080/domibus/services/msh
messageInErrorId: 92620d75-c900-4007-b1f9-fc2c3fae0c61@domibus.eu
mshRole: SENDING
notified: null
timestamp: 2017-09-07 11:48:03.0
```

3.6. Receive Files Sequence

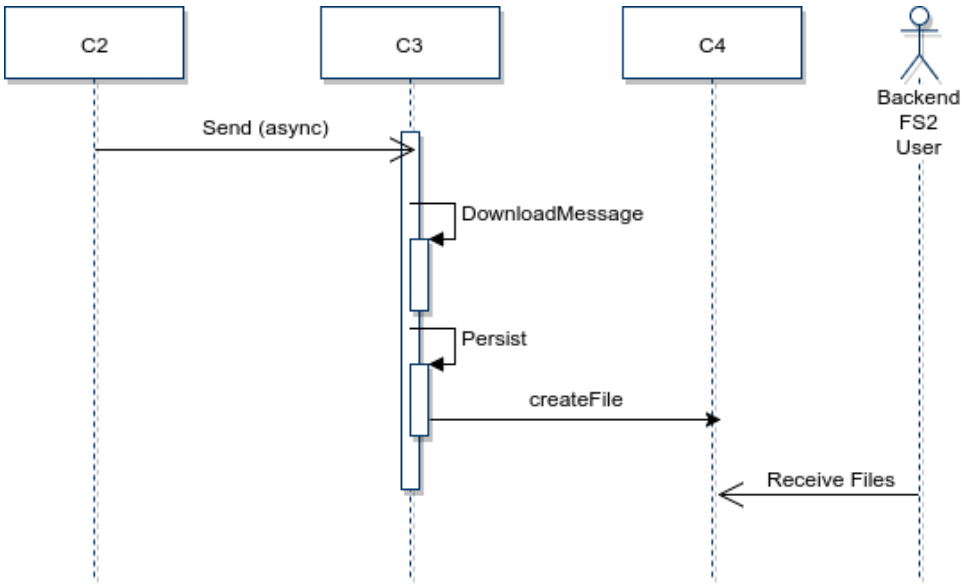

Figure 12 - Receive Files sequence diagram

When the Domibus core (C3) receives an AS4 message it passes it to the FS Plugin. The FS plugin starts by extracting the Service and Action values from the message and crosses that information with the various domain expressions to select the applicable domain location defining the destination directory. The domain expressions are taken from properties "fsplugin.domains.<domain_id>.messages.expression" and are evaluated in order, according to properties "fsplugin.domains.<domain_id>.order". If no domain is found to match the pair Service/Action, the main location is selected as the destination folder.

If the AS4 message contains a single payload, a new file is created in the incoming folder of the location identified above. This file will have a name in the form:

messageID.extension

Where ***messageId*** is the id of the message and ***extension*** is a file extension derived from the MIME type of the payload.

If the AS4 message contains multiple payloads[4], multiples files are created. Those files will be named similar to the above but also contain the CID of the payload, in the form:

messageID_partInfoCid.extension

Example:

6d38e798-26d7-45a9-9314-3a280cf02c8d_message.pdf

[4] Though only one payload can be sent at once using the FS Plugin, reception of multiple payloads is possible if initiated with other plugins.

4. MULTITENANCY

The Default 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 §1.4-References), section **Plugin Users.**

5. ANNEXE 1 – DOCUMENT PARTS

5.1. Diagrams Source

FS_Plugin_Diagrams.svg

5.2. metadata.xml example

```
<?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>
</UserMessage>
```

6. List of Figures

7. List of Tables

8. CONTACT INFORMATION

CEF Support Team

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

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