EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

Access Point

Interface Control Document

WS plugin

Version [3.8]

Status [Final]

Date: 19/03/2024

Table of Contents

1. INTRODUCTION

1.1. Purpose of the Interface Control Document

This document will univocally define the participant's 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 eDelivery building block. This document describes the WSDL and the observable behaviour of the interface provided by Domibus and included in the default-ws-plugin.

This interface control document will be used for the understanding of the Access Point (Corner Two and Corner Three in the four-corner model) services provided by Domibus delivered by eDelivery.

There is 1 interface described in this document:

Interface	Description
WebServicePlugin.wsdl	The webservices interface for Domibus WS default plugin

1.2. Scope of the document

This document covers the service interface of the Access Point. It includes information regarding the description of the services available, the list of use cases, the information model and the sequence of message exchanges for the services provided. This specification is limited to the service interface of the Access Point. All other aspects of its implementation are not covered by this document. 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:

- *Interface Functional Specification*, this specifies the set of services and the operations provided by each service and this is represented by the flows explained in the use cases.
- *Interface Behavioural Specification,* this specifies the expected sequence of steps to be respected by the participants in the implementation when calling a service or a set of services and this is represented by the sequence diagrams presented in the use cases.
- *Interface Message standards,* this specifies the syntax and semantics of the data, and this is explained in the Annex §9.1 - WS plugin interface message standards.

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 systems and the Access Point. In particular:
    - Architects will find it useful for determining how to best exploit the Access Point 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 Access Point 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 Access Point plugin services.
- 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 outline
[REF1]	Access Point Component Offering Description	Overview of eDelivery Access Point
[REF2]	Using HTTP Methods for RESTful Services	Short description of HTTP Methods for RESTful Services
[REF3]	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.
[REF4]	ebXML (Electronic Business using eXtensible Markup Language)	ebXML (Electronic Business using eXtensible Markup Language)
[REF5]	Web Services Description Language (WSDL) 1.1	Web Services Description Language (WSDL) 1.1 WS-I Basic Profile Version 1.1
[REF6]	XML Schema 1.1	XML Schema 1.1
[REF7]	Extensible Markup Language (XML) 1.1	Extensible Markup Language (XML) 1.1
[REF8]	Hypertext Transfer Protocol 1.1	Hypertext Transfer Protocol 1.1
[REF9]	SOAP Messages with Attachments	SOAP Messages with Attachments
[REF10]	AS4 Profile of ebMS 3.0 Version 1.0	AS4 Profile of ebMS 3.0 Version 1.0
[REF11]	eDelivery AS4 profile	https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eDelivery+AS4
[REF12]	eDelivery – Pmode Configuration	eDelivery – Pmode Configuration *(will be available at a later stage)*
[REF13]	http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/	XSDs for ebms3
[REF14]	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)

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.

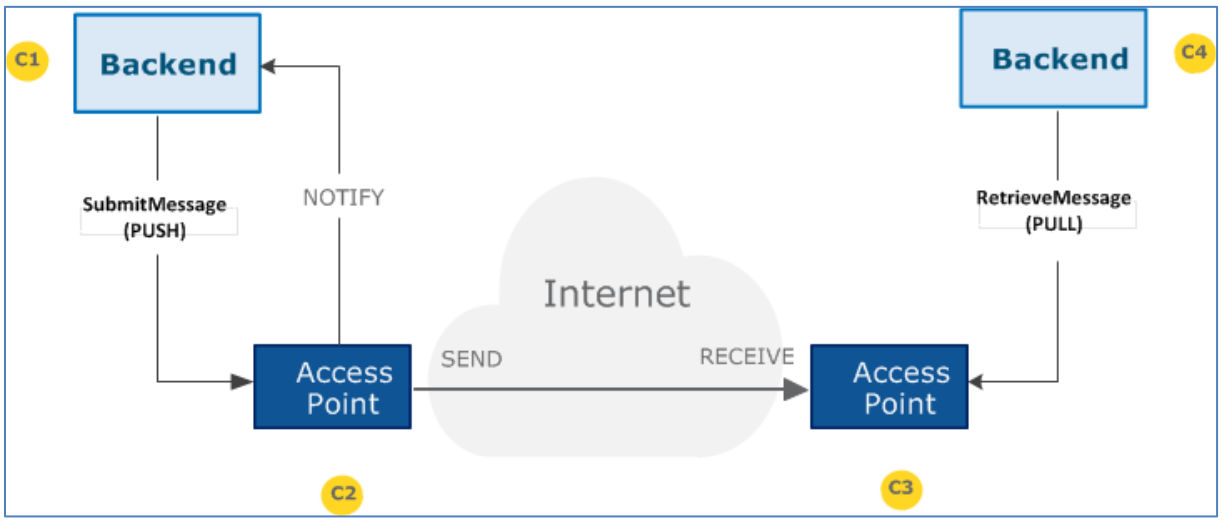

Figure 1 - The four corner model

In this model we have the following elements:

- Corner One (C1): Backend C1 is the system that will send messages to the sending AP (Access Point)
- Corner Two (C2): Sending Access Point C2
- Corner Three (C3): Receiving Access Point C3
- Corner Four (C4): Backend C4 is the system that received messages from the receiving AP (Access Point)

There are two backend adapters (i.e. corner one and corner four). They send messages to and download messages from the AS4 APs configured in the PMode configuration files.

2.1. Purpose of the Access Point component

The Access Point provides the functionality supporting Corner Two and Corner Three components.

2.2. Use case overview

2.2.1. Actors

Actor	Definition
Backend C1	Any participant submitting messages to any other Backend C4 and using the Sending AP C2 for that purpose.
Backend C4	Any participant retrieving messages from any other Backend C1 and using the Receiving AP C3 for that purpose.

Table 1 - Actor list

Note: greyed use cases in this paragraph show deprecated operations in the WSDL (in these diagrams, the use cases below these replace them). Since deprecated and replacing operations have the same functionality (only technical changes), in each case only one use case is presented for both.

2.2.2. Use cases diagram

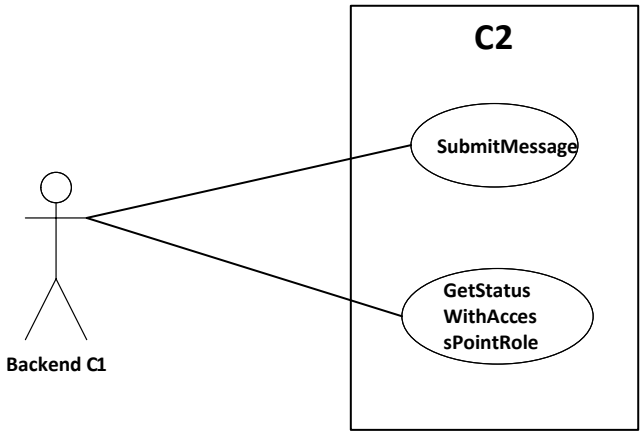

Figure 1 - Backend C1 Use cases diagram

ID	UC	Short description	Oper.	System
UC01	**Submit Message**	Submit any type of document from a Backend C1 to a Backend C4	submitMessage	Domibus 4.x.y
UC03	**Get Status of the Message**	Get the status of the Message with messageId and access point role.	getStatusWithAccessPointRole	Domibus 5.1

Table 2 - C2 Use cases

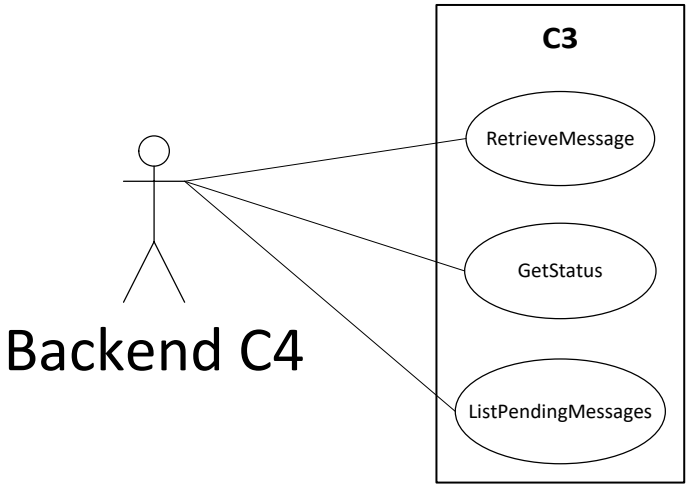

Figure 2 - backend C4 Use cases diagram

ID	UC	Short description	Oper.	System
UC02	**Download Message**	Retrieve the message from the Receiving AP C3	retrieveMessage	Domibus 4.x.y
UC03	**Get Status of the Message**	Get the status of the Message	getStatus	Domibus 4.x.y
UC04	**ListPending Messages**	Check the pending messages to be retrieved by the Backend C4 from C3	listPendingMessages	Domibus 4.x.y

Table 3 - C3 Use cases

2.3. Detailed uses cases

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

The *Interface Functional Specification* is described in the detailed uses cases using the Request and the Response examples. It is important to remark that the Inputs and Responses provided as examples for the use cases are based on a specific PMode configuration.

As defined in the eDelivery Specification Library, a PMode is the contextual information that governs the processing of a particular message (thus is basically a set of configuration parameters). The PMode associated with a message determines, among other things, which security and/or which reliability protocol and parameters, as well as which MEP (Message Exchange Pattern) is being used when sending a message. The technical representation of the PMode configuration is implementation dependent. C1 and C4 may be one or more participants.

The state machine diagrams presented below depict the various states in which a message may be during its lifecycle when submitting or downloading the message. These are presented in order to have a more comprehensive vision of the process that the messages go through. It is also important to remark that also the sequence diagram of the basic flow is presented in the use cases.

On C2, the state machine diagram for submitting the message:

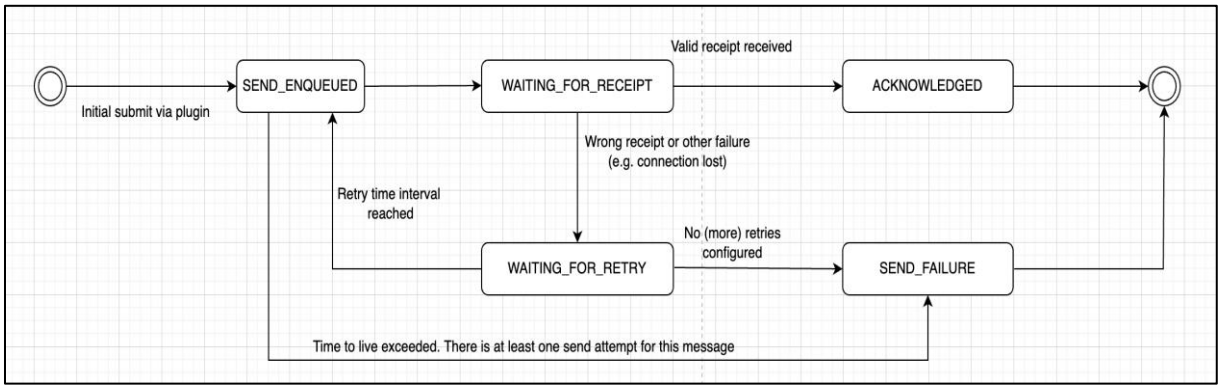

Figure 3 – State machine of C2

On C3, the state machine diagram for downloading the message:

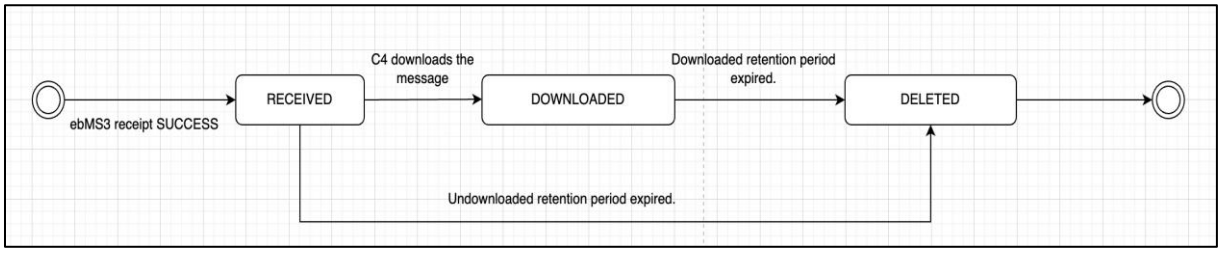

Figure 4 - State machine of C3

2.3.1. UC01 – Submit Message

Brief description

Submit any message with attachments from Backend C1 to the Backend C4. The response from C2 to C1 is synchronous and contains a messageId.

MTOM feature is required when sending large files so that the attachments are send outside the XML envelope, as parts. Otherwise, the attachments will be encoded base64 and sent inside the envelope and the maximum limit would be 128Mb.

The state machine of the outgoing messages is the following:

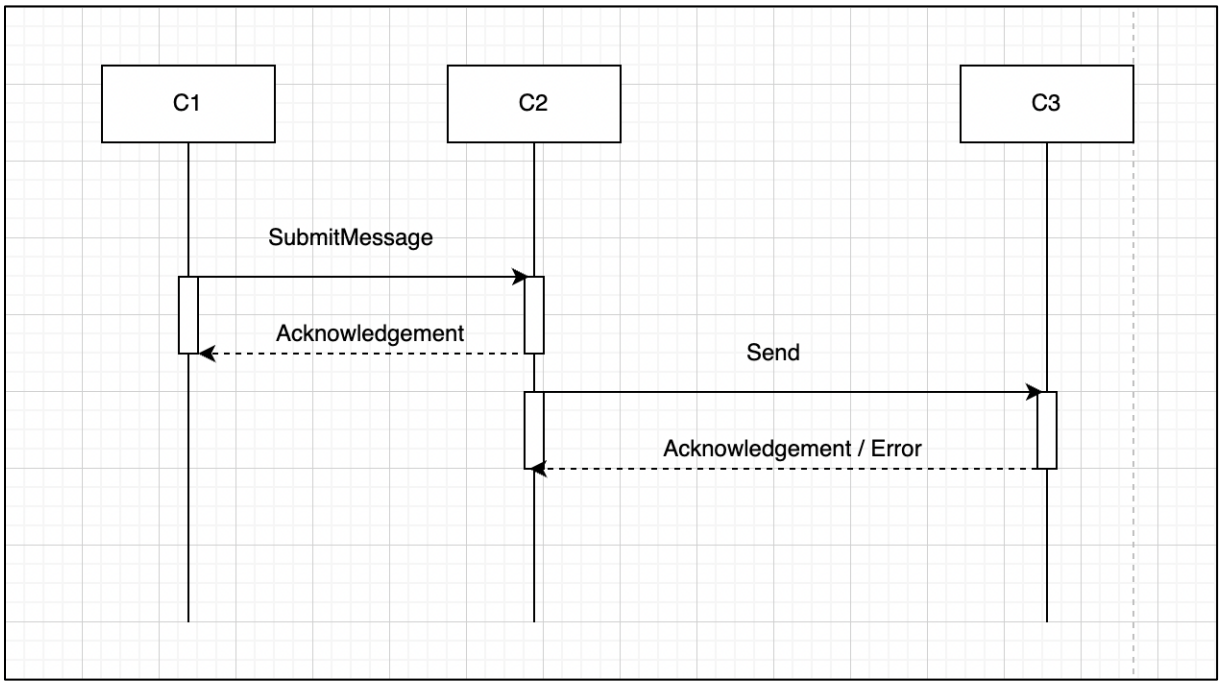

Figure 5 - Sequence Diagram C1 to C2 – SubmitMessage

Actors			
C1			Backend C1
	C2		Access Point C2
		C3	Access Point C3

Preconditions			
C1			Backend C1 has a message to submit.
C1			The message is valid. A message is valid if it respects the message standard format (see Annex §9.1 - WS plugin interface message standards)
	C2	C3	The Sending AP (C2) and the Receiving AP (C3) are up and running and properly configured.

Basic Flow			C2 Message State	C3 Message state
Actor	**Step**			
C1	1	Backend C1 submits the message.	n/a	
C2	2	C2 sends an ACK to C1 containing the ID of the message.	SEND_ENQUEUED	
C2	3	The message directly passes through to SEND_ENQUEUED, meaning that it is available for processing. All messages go through this step regardless of load	SEND_ENQUEUED	
C2	4	Once the sending process finishes the status changes to WAITING_FOR_RECEIPT.	WAITING_FOR_RECEIPT	
C3	5	Once the reception is finished by C3, the status changes to RECEIVED	WAITING_FOR_RECEIPT	RECEIVED
C3	6	The Receiving AP (C3) responds ACK to C2	WAITING_FOR_RECEIPT	RECEIVED
C2	7	The status of the message changes to ACKNOWLEDGED. If configured in the PMode for non-repudiation, the receipt SHOULD contain a single ebbpsig:NonRepudiationInformation child element. The value of eb:MessageInfo/eb:RefToMessageId MUST refer to the message for which this signal is a receipt.	ACKNOWLEDGED	RECEIVED
	8	Use case ends in successful condition.		

Exception flow			C2 Message State	C3 Message state
C2	**E2.1**	**The ID provided by C1 already exists**		
	E2.1.1	The parameter PMode[1].ReceptionAwareness.DuplicateDetection must be set to true		
	E2.1.2	The status of the message changes to SEND_FAILURE	SEND_FAILURE	
C2	**E11.1**	**Wrong receipt received or any other failure (e.g connection lost)**		
C2	E11.1.1	The status changes to WAITING_FOR_RETRY	WAITING_FOR_RETRY	
C2	E11.1.2	Continue to step 3		
C2	**E11.1.3.1**	**The maximum number or retries (Configurable via PMode on a by-usecase basis) has been reached**		
C2	E11.1.3.1.1	The status of the message changes to SEND_FAILURE.	SEND_FAILURE	
	E11.1.3.1.2	A notification can be sent to the Backend C1 that initially submitted the message.	SEND_FAILURE	
C2	E11.1.3.1.3	Use case ends in failure condition.		

Post conditions

	Successful conditions
	The operation is a success if getStatus in C2 is ACKNOWLEDGED and this means that the Receiving AP (C3) has received the message submitted by the Backend C1 and the status in C3 is RECEIVED. The method getStatus must be called with the identifier of the message received in the response or specified in the request.
	Failure Conditions
	Errors may be sent as SOAP Fault or as http:5XX.

Example of the request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ns="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:_1="http://eu.domibus.wsplugin/">
  <soap:Header>
    <ns:Messaging>
     <ns:UserMessage mpc=" http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultMPC">
        <ns:PartyInfo>
          <ns:From>
            <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-blue</ns:PartyId>
            <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
          </ns:From>
          <ns:To>
            <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-red</ns:PartyId>
            <ns:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder</ns:Role>
          </ns:To>
        </ns:PartyInfo>
        <ns:CollaborationInfo>
          <ns:Service type="tc1">bdx:noprocess</ns:Service>
          <ns:Action>TC1Leg1</ns:Action>
        </ns:CollaborationInfo>
        <ns:MessageProperties>
          <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</ns:Property>
          <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>
    </ns:Messaging>
  </soap:Header>
  <soap:Body>
    <_1:submitRequest>
      <!--Optional-->
      <bodyload>
           <value>cid:bodyload</value>
       </bodyload>
      <payload payloadId="cid:message" contentType="text/xml">
       <value>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPGhlbGxvPndvcmxkPC9oZWxsbz4=</value>
      </payload>
    </_1:submitRequest>
  </soap:Body>
</soap:Envelope>
<soap:Body>

    <_1:submitRequest>
      <payload payloadId="cid:message" contentType="text/xml">
       <value>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPGhlbGxvPndvcmxkPC9oZWxsbz4=</value>
      </payload>
      <payload payloadId="cid:attachment" contentType="application/octet-stream">
       <value>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPGhlbGxvPndvcmxkPC9oZWxsbz4=</value>
      </payload>
    </_1:submitRequest>
  </soap:Body>
```

</soap:Envelope>

Example of the response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <ns5:submitResponse xmlns:ns6="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:ns5="http://eu.domibus.wsplugin" xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
      <messageID>23dce7d9-2781-4623-beeb-6b43ab9e7d37@domibus.eu</messageID>
    </ns5:submitResponse>
  </soap:Body>
</soap:Envelope>
```

Special requirements

N/A

2.3.2. UC02 - Retrieve Message

Brief description

Retrieve any type of message sent from Backend C1 to Backend C4. The retrieval of the message is based on a PULL mechanism. C4 downloads the message from C3.

Please note that retrieveMessage method replaces the deprecated method downloadMessage to support the retrieval of large files. MTOM feature is required when retrieving large files.

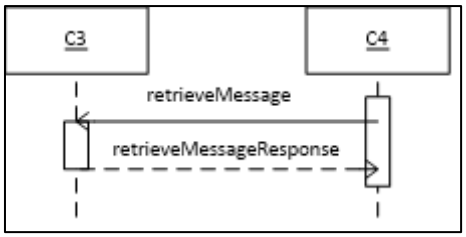

Figure 6 - Sequence Diagram C4 to C3 – retrieveMessage

Actors	
C3	Access Point C3
C4	Backend C4

Preconditions	
C3	There is at least one message sent by AP C2 and successfully received in the Receiving AP C3.
C3	The Receiving AP (C3) is up and running and properly configured.
C4	C4 Has previously requested information about pending messages from C3. C3 has returned a response containing the messageID('s) of the message(s) received (cf. UC04).

Basic Flow			C3 Message State
Actor	**Step**		
C4	1	Requests, to the Receiving AP C3, the service retrieveMessage by providing the messageId and the attibute markAsDownloaded with the value true (which is the default value when this optional attribute is missing)	RECEIVED
C3	2	Receiving AP C3 retrieves and sends the information retrieveMessageResponse to C4 as response to his request. This is the payload (message content and attachments) and metadata, analogous to the message sent from C1 to C2 in UC01. The status of the message changes to *DOWNLOADED* when the message is retrieved by the backend C4.	DOWNLOADED
C4	3	Receives the message as sent by C1	DOWNLOADED

C3		4	Deletes the payload of the message from the database if retention timeout for downloaded messages = 0. NB: While the message metadata is still recoverable by a Domibus administrator all payload data is purged. This is necessary to be able to prove message exchanges in case of disputes. It is possible to produce the signature of a payload but not the payload itself.	DELETED
C3		5	The status of the message changes to ***DELETED*** when the message is deleted by C3 after the configured retention timeout for downloaded messages (*retention_downloaded*) expired. Note: If *retention_downloaded* has a negative value, the message will never be deleted from C3. If the message was not downloaded, the *retention_undownloaded* value will be used as timeout for deletion.	DELETED
		6	Use case ends	DELETED

Alternative flow 1			**C3 Message State**
C3	A1.1	**Configured retention time has passed**	RECEIVED
C3	A1.1.1	Go directly to step 4	RECEIVED

Alternative flow 2				**C3 Message State**
	C4	A2.1	Requests, to the Receiving AP C3, the service retrieveMessage by providing the messageID and the attibute **markAsDownloaded with the value false**	RECEIVED
C3		A2.2	Receiving AP C3 retrieves and sends the information retrieveMessageResponse to C4 as response to his request. This is the payload (message content and attachments) and metadata, analogous to the message sent from C1 to C2 in UC01.	RECEIVED
	C4	A2.3	Receives the message as sent by C1	RECEIVED
	C4	A2.4	Requests, to the Receiving AP C3, the service markMessageAsDownloaded by providing the messageID	RECEIVED

C3	A2.5	C3 responds with a confirmation message containing the messageID if the message was not marked as DOWNLOADED until then or an error message otherwise. The status of the message changes to *DOWNLOADED* when the message is retrieved by the backend C4. Also, C3 deletes the downloaded message from the plugin table containing the pending messages.	DOWNLOADED
C3	A2.6	Go to step 4	DELETED

Exception flow			C3 Message State
C3	**E2.1**	**Wrong messageID (malformed or missing), this is a condition of failure.**	**(NOT FOUND)**
C3	E2.1.1	The NOT FOUND status is a pseudo state for messages that are not available for download (were never received or were rejected).	(NOT FOUND)
C3	E2.1.2	Use case ends in failure condition	(NOT FOUND)

Post conditions	
	Successful conditions
	It is a success if the Message Status is *ACKNOWLEDGED* on C2 and DOWNLOADED or *DELETED* on C3. Payload of the message may be deleted from C3's database.
	Failure Conditions
	No message payload is returned to C4. The response contains a description of the encountered error. The operation is not a success if GetStatus is *SEND_FAILURE* on C2 and *NOT FOUND* on C3 and this means that the Backend C4 has not been able to download the message submitted by the Backend C1. If the retrieveMessage method is called with a wrong messageID (malformed or missing), this is a condition of failure. The *NOT FOUND* status is a pseudo state for messages that are not available for download (were never received or were rejected).

Example of the request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:_1=" http://eu.domibus.wsplugin/">
  <soap:Header/>
  <soap:Body>
    <_1:retrieveMessageRequest>
      <messageID>${ResponseParameters#messageID}</messageID>
    </_1:retrieveMessageRequest>
  </soap:Body>
</soap:Envelope>
```

Example of the response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <ns6:Messaging mustUnderstand="false" xmlns:ns6="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:ns5="
http://eu.domibus.wsplugin/" xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
      <ns6:UserMessage> mpc=" http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultMPC">
        <ns6:MessageInfo>
          <ns6:Timestamp>2017-10-02T17:32:14.956+02:00</ns6:Timestamp>
          <ns6:MessageId>d05051c6-951c-4f40-90b5-459eca9d8302@domibus.eu</ns6:MessageId>
        </ns6:MessageInfo>
        <ns6:PartyInfo>
          <ns6:From>
            <ns6:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-blue</ns6:PartyId>
            <ns6:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator</ns6:Role>
          </ns6:From>
          <ns6:To>
            <ns6:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-red</ns6:PartyId>
            <ns6:Role>http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder</ns6:Role>
          </ns6:To>
        </ns6:PartyInfo>
        <ns6:CollaborationInfo>
          <ns6:Service type="tc1">bdx:noprocess</ns6:Service>
          <ns6:Action>TC1Leg1</ns6:Action>
          <ns6:ConversationId>52f1c57d-bd35-4ab2-a0a5-da9a15101dba@domibus.eu</ns6:ConversationId>
        </ns6:CollaborationInfo>
        <ns6:MessageProperties>
          <ns6:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4</ns6:Property>
          <ns6:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</ns6:Property>
        </ns6:MessageProperties>
        <ns6:PayloadInfo>
          <ns6:PartInfo href="cid:message">
            <ns6:Schema/>
            <ns6:PartProperties>
              <ns6:Property name="MimeType">text/xml</ns6:Property>
            </ns6:PartProperties>
          </ns6:PartInfo>
        </ns6:PayloadInfo>
      </ns6:UserMessage>
    </ns6:Messaging>
  </soap:Header>
  <soap:Body>
    <ns5:retrieveMessageResponse xmlns:ns6="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:ns5="
http://eu.domibus.wsplugin/" xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
      <payload payloadId="cid:message">
        <value>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPGhlbGxvPndvcmxkPC9oZWxsbz4=</value>
      </payload>
    </ns5:retrieveMessageResponse>
  </soap:Body>
</soap:Envelope>
```

Security

In Multitenancy mode, the general property **domibus.auth.unsecureLoginAllowed** is ignored and authentication is always required. More about authentication options can be read in the Administration Guide, in section “Domibus Authentication”.

2.3.3. UC03 - Get the status of the Message

Brief description

Get the status of the Message sent from Backend C1 or received by the Backend C4:

Figure 7 - Sequence Diagram – GetStatusWithAccessPointRole

Actors	
C1	Backend C1
C2	Access Point C2
C3	Access Point C3
C4	Backend C4

Preconditions	
	There is at least one message sent by Backend C1 or to be retrieved by Backend C4. The Sending AP C2 and the Receiving AP C3 are up and running and properly configured.

Basic flow event

Step

1. Backend C1 or the Backend C4 launch a statusRequest using the messageId and access point role.
2. The Access Point (Sending AP C2 or Receiving AP C3) retrieve the getStatusResponse.
3. Use case ends.

Exception flow

N/A

Post conditions

Successful conditions

The operation is a success if GetStatusResponse retrieves any status of the following:

- SEND_ENQUEUED
- WAITING_FOR_RECEIPT
- ACKNOWLEDGED
- SEND_FAILURE
- NOT_FOUND
- WAITING_FOR_RETRY
- RECEIVED
- DELETED
- DOWNLOADED

Failure Conditions

The message does not exist.

Special requirements

N/A

Security

In Multitenancy mode, the general property **domibus.auth.unsecureLoginAllowed** is ignored and authentication is always required. More about authentication options can be read in the https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Domibus, section “Domibus Authentication”.

Example of the request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:_1=" http://eu.domibus.wsplugin/">
  <soap:Header/>
  <soap:Body>
    <_1:statusRequest>
      <messageID>d05051c6-951c-4f40-90b5-459eca9d8302@domibus.eu</messageID>
    </_1:statusRequest>
  </soap:Body>
</soap:Envelope>
```

Example of the response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <ns5:getStatusResponse xmlns:ns6="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" xmlns:ns5="
http://eu.domibus.wsplugin/" xmlns:xmime="http://www.w3.org/2005/05/xmlmime">NOT_FOUND</ns5:getStatusResponse>
  </soap:Body>
</soap:Envelope>
```

2.3.4. UC04 – List Pending Messages

Brief description

List the status Messages pending to be received by the Backend C4.

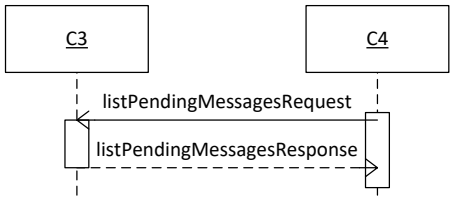

Figure 8 - Sequence Diagram C4 to C3 – ListPendingMessages

Actors	
C4	Backend C4
Preconditions	
	There is at least one message be downloaded by Backend C4. The Receiving AP C3 is up and running and properly configured.

Basic flow event

Step

1. Backend C4 launches the service listPendingMessages.
2. The Access Point (Receiving AP C3) retrieves the list of messageIds for messages with status RECEIVED.
3. Use case ends.

Exception flow	
N/A	
Post conditions	
	Successful conditions
	The operation is a success if listPendingMessagesResponse contains all the messageIDs of the pending messages to be retrieved or the list is empty in the case that there are no pending messages.
	Failure Conditions
	N/A

Special requirements

N/A

Security

In Multitenancy mode, the general property **domibus.auth.unsecureLoginAllowed** is ignored and authentication is always required. More about authentication options can be read in the Administration Guide, in the section “Domibus Authentication”.

When authentication is required, the returned messages will always be filtered having authenticated user overwriting finalRecipient value (if finalRecipient is provided as below).
The listPendingMessagesRequest accepts 8 *optional* parameters in order to filter the returned list of messages in status RECEIVED.

Date time optional parameters like “receivedFrom” and “receivedTo” can be provided in ISO-8601 format, with or without an offset. When the offset is provided, the user MUST NOT provide any additional timezone IDs so a value of “2021-07-21T14:27:00+02:00[Europe/Brussels]” is invalid. When the offset is missing, these parameter values are considered to be provided in UTC, having an offset of “+00:00”. For example, the following two values are valid and point to the same instant of 21st of July 2021, 12:27:00 in UTC:

1. “2021-07-21T14:27:00+02:00” will be interpreted to have an offset of 02:00;

2. “2021-07-21T12:27:00” will be interpreted to be in the UTC timezone with a +00:00 offset.

Example of the request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:_1=" http://eu.domibus.wsplugin/">
  <soap:Header/>
  <soap:Body>     <_1:listPendingMessagesRequest></_1:listPendingMessagesRequest>
         <messageId>4078cfea-74e9-4058-9d14-1dceee597abd@domibus.eu </messageId>
         <conversationId>52f1c57d-bd35-4ab2-a0a5-da9a15101dba@domibus.eu</conversationId>
         <refToMessageId>d05051c6-951c-4f40-90b5-459eca9d8302@domibus.eu</refToMessageId>
         <fromPartyId>domibus-blue</fromPartyId>
         <finalRecipient>urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4</finalRecipient>
          <originalSender>urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</originalSender>
         <receivedFrom>2021-01-15T09:00:00</receivedFrom>
         <receivedTo>2021-01-29T09:00:00</receivedTo>
      </_1:listPendingMessagesRequest>
  </soap:Body>
</soap:Envelope>
```

Example of the response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <ns6:listPendingMessagesResponse xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/" xmlns:eb="http://docs.oasis-open.org/ebx
msg/ebms/v3.0/ns/core/200704/" xmlns:S12="http://www.w3.org/2003/05/soap-envelope" xmlns:xmime="http://www.w3.org/2005/05/xr
xmlns:ns6=" http://eu.domibus.wsplugin/">
       <messageID>4078cfea-74e9-4058-9d14-1dceee597abd@domibus.eu</messageID>
    </ns6:listPendingMessagesResponse>
  </soap:Body>
  </soap:Envelope>
```

3. Interface Behavioural Specification

3.1. WS plugin configuration

The WS plugin configuration is done in the ws-plugin.properties file. Following properties are configurable in this file:

Property name	Default value	Description
wsplugin.mtom.enabled	false	Enable the support for MTOM.
wsplugin.schema.validation.enabled	false	Enable the schema validation. By default, the schema validation has been disabled due to performance reasons. For large files, it is recommended to keep the schema validation as disabled.
wsplugin.messages.pending.list.max	500	The maximum number of pending messages to be listed from the pending messages table. Setting this property is expected to avoid timeouts due to huge *resultsets* being served. Setting this property to zero returns all pending messages.
wsplugin.messages.notifications	MESSAGE_RECEIVED,MESSAGE_SEND_FAILURE,MESSAGE_RECEIVED_FAILURE,MESSAGE_SEND_SUCCESS,MESSAGE_STATUS_CHANGE	The notifications sent by Domibus to the plugin. The following values are possible: MESSAGE_RECEIVED,MESSAGE_FRAGMENT_RECEIVED,MESSAGE_SEND_FAILURE,MESSAGE_FRAGMENT_SEND_FAILURE,MESSAGE_RECEIVED_FAILURE,MESSAGE_FRAGMENT_RECEIVED_FAILURE,MESSAGE_SEND_SUCCESS,MESSAGE_FRAGMENT_SEND_SUCCESS,MESSAGE_STATUS_CHANGE,MESSAGE_FRAGMENT_STATUS_CHANGE
wsplugin.dispatcher.connectionTimeout	240000	Timeout values for communication between the ws plugin and the backend service ConnectionTimeOut - Specifies the amount of time, in milliseconds, that the consumer will attempt to establish a connection before it times out. 0 is infinite.
wsplugin.dispatcher.receiveTimeout	240000	ReceiveTimeout - Specifies the amount of time, in milliseconds, that the consumer will wait for a response before it times out. 0 is infinite.
wsplugin.dispatcher.allowChunking	false	Allows chunking when sending messages to the backend service
wsplugin.dispatcher.chunkingThreshold	104857600	If domibus.dispatcher.allowChunking is true, this property sets the threshold at which messages start getting chunked(in bytes). Messages under this limit do not get chunked. Defaults to 100 MB.
wsplugin.dispatcher.connection.keepAlive	true	Specifies if the connection will be kept alive between C2-C1 and C3-C4. Default value is true.
wsplugin.dispatcher.work	0 0/1 * * * ?	Specify concurrency limits via a "lower-upper" String,

Property name	Default value	Description
er.cronExpression		e.g. "5-10", or a simple upper limit String, e.g. "10" (the lower limit will be 1 in this case) when sending files
wsplugin.push.enabled	false	Enable push notification to backend. Properties wsplugin.push.rules.X, wsplugin.push.rules.X.recipient, wsplugin.push.rules.X.endpoint, wsplugin.push.rules.X.retry and wsplugin.push.rules.X.type needed with X finalRecipient
wsplugin.push.rules.X		Description of the rule X
wsplugin.push.rules.X.recipient		Recipient that will trigger the rule (ex: urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1). If empty, the rule is triggered for any recipient.
wsplugin.push.rules.X.endpoint		End point used to submit a message to the backend (ex: http://localhost:8080/backend)
wsplugin.push.rules.X.retry		Cron expression for the retry mechanism to push to backend
wsplugin.push.rules.X.type		Type of notifications to be sent to backend: RECEIVE_SUCCESS,RECEIVE_FAIL,SEND_SUCCESS,SEND_FAILURE,MESSAGE_STATUS_CHANGE,SUBMIT_MESSAGE (see §7.4-Notifications to backend).
wsplugin.push.auth.username		Basic authentication username that will be added to the http header of push notification requests to C4. If not specified, no authorization header will be added.
wsplugin.push.auth.password		Basic authentication password that will be added to the http header of push notification requests to C4. If not specified, no authorization header will be added.
wsplugin.push.markAsDownloaded	true	If true, the SUBMIT_MESSAGE notification also pushes the message. If false, the backend will be able to retrieve the same message multiple times and explicitly set the message status to downloaded.

WSDL model for Domibus 5.x.y

3.2. WSDL model for Domibus 5.x.y

3.2.1. The WSDL schema

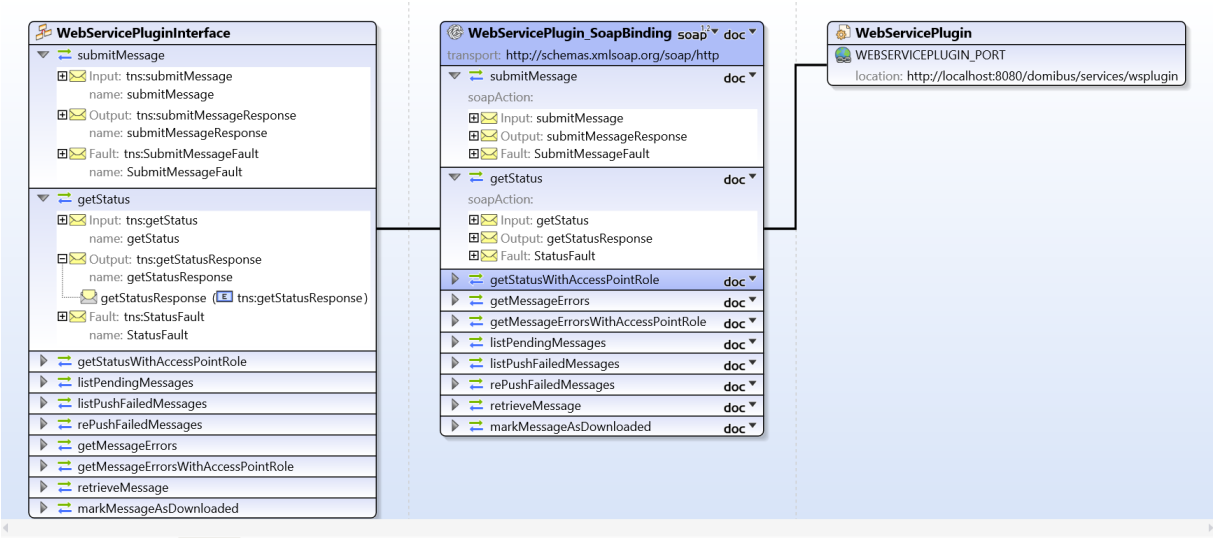

Figure 9 - WSDL model for Domibus 5.x.y

The WSDL defines the envelope that consists of one AP Header and one AP Body.

The service sends a message and receives a response. There are **ten operations:**

- submitMessage
- getStatus: this method is marked as deprecated, but it is maintained for backwards compatibility. For selfsending message, this method will throw duplicate messages found as exceptions. Instead of this method, use the newly added method getStatusWithAccessPointRole where you can specify the access point role.
- getStatusWithAccessPointRole: this newly added method is to get the status based on the messageId and the role of the access point. For empty messageId or invalid AP role, it throws exception.
- listPendingMessages
- getMessageErrors: it can be used if you get a SEND_FAILURE status as response from the getStatus service, in which case this operation can be used to get the details of the encountered errors. There can be multiple errors as each retry might produce one. For selfsending message, this method will throw duplicate messages found as exceptions. Instead of this method, use the newly added method getMessageErrorsWithAccessPointRole where we can specify the access point role.
- getMessageErrorsWithAccessPointRole: this newly added method is used to get the details of the encountered errors of the failed messages with messageId and accesspoint role. There can be multiple errors as each retry might produce one. For empty messageId or invalid AP role, it throws exception.
- retrieveMessage: this operation retrieves the message and also download the message based on a query parameter 'markAsDownloaded'. By default the value is 'true' and the method marks the message as downloaded. Also, it delete the downloaded message from the plugin table containing the pending messages. If markAsDownloaded=false, WS Plugin will not mark the message as downloaded.
- listPushFailedMessages: this operation returns the list of messagesId's which are pushed to C4 but still in the failed status. We can also check if messages are pushed but still in failed status by its messageId, originalSender, finalRecipient, receivedFrom or receivedTo date.

- rePushFailedMessages: this operation allows to repush the list of failed messages by the message id's.
- markMessageAsDownloaded: This operation marks the message as downloaded. Also, it will delete this message from the WS plugin table.

To encapsulate errors, the *fault* following elements are specified for the respective services:

- <wsdl:fault name="SubmitMessageFault"/>
- <wsdl:fault name="RetrieveMessageFault"/>
- <wsdl:fault name="StatusFault"/>
- <wsdl:fault name="MarkMessageAsDownloadedFault"/>
- <wsdl:fault name="getMessageErrorsFault"/>
- <wsdl:fault name="listPendingMessagesFault"/>
- <wsdl:fault name="listPushFailedMessagesFault"/>
- <wsdl:fault name="rePushFailedMessagesFault"/>

It must be generated and processed according to the [SOAP1.2] specification. In this case, SOAP protocol is used and the binding is **<soap:binding>**. The transport is SOAP messages on top of HTTP protocol:

transport="*http://schemas.xmlsoap.org/soap/http*"/>

3.2.2. The data model applicable to SubmitMessage from C1 to C2 (domibus-submission.xsd)

In this section the data model is explained.

3.2.2.1. Messaging/UserMessage mpc attribute:

The Optional attribute occurs once and contains the qualified name of the MPC (Message Partition Chanel). 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.

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
mpc	/*[local-name()='schema']/*[local-name()='complexType' and @name=UserMessage]//*[local-name()='attribute' and @name='mpc']	**N**	**Max 1**	• It is a non-empty string, • Max length:255 characters • Configuration in PMode: PMode.mpcs.mpc	*Default value:* **http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultMPC**

In this section, the data model is explained.

3.2.2.2. Messaging/UserMessage/MessageInfo:

This Optional element occurs once and contains the identifier of the current message, and (may) relate to other messages' identifiers.

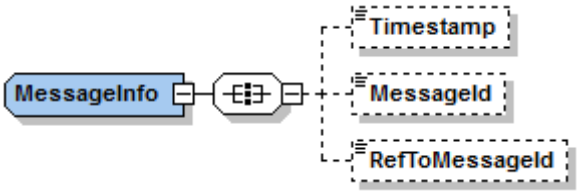

Figure 10 – MessageInfo type

- **Timestamp** element has a value representing the date at which the message header was created.

- **MessageId** has a value representing – for each message - a globally unique identifier.

- **RefToMessageId** contains the MessageId value of an ebMS Message to which this message relates, in a way that conforms to the MEP in use.

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
Timestamp	*[local-name()='schema']/*[local-name()='complexType' and @name='MessageInfo']/*[local-name()='all']/*[local-name()='element' and @name='Timestamp']	**N**	**Max 1**	• It MUST be expressed as *YYYY-MM-DDTHH:MM:SS.msmsmsZ*	*2016-03-31T09:00:44.418Z*
MessageId	*[local-name()='schema']/*[local-name()='complexType' and @name='MessageInfo']/*[local-name()='all']/*[local-name()='element' and @name='MessageId']	**N**	**Max 1**	• A globally unique identifier • In the Message-Id and Content-Id MIME headers, values are always surrounded by angle brackets. However references in mid: or cid: scheme URI's and the MessageId and RefToMessageId elements MUST NOT include these delimiters. • It is a non-empty string. • Max length: value should not be more than 255 characters.	*346ea37f-7583-40b0-9ffc-3f4cfa88bf8b@domibus.eu*
RefToMessageId	/*[local-name()='schema']/*[local-name()='complexType' and @name='MessageInfo']/*[local-name()='all']/*[local-name()='element' and @name='RefToMessageId']	**N**	**Max 1**	• A globally unique identifier. • In the Message-Id and Content-Id MIME headers, values are always surrounded by angle brackets. However, references in mid: or cid: scheme URI's and the MessageId and RefToMessageId elements MUST NOT include these delimiters. • It is a non-empty string. • Max length: value should not be more than 255 characters.	*346ea37f-7583-40b0-9ffc-3f4cfa88bf8b@domibus.eu*

3.2.2.3. Messaging/UserMessage/PartyInfo

This REQUIRED element occurs once, and contains data about originating and destination parties. This element has the following children elements:

- **From:** This REQUIRED element occurs once, and contains information describing the originating party. It can be either endpoint C1 or endpoint C2.
- **To:** This REQUIRED element occurs once, and contains information describing the destination party and it can be either endpoint C3 or endpoint C4.

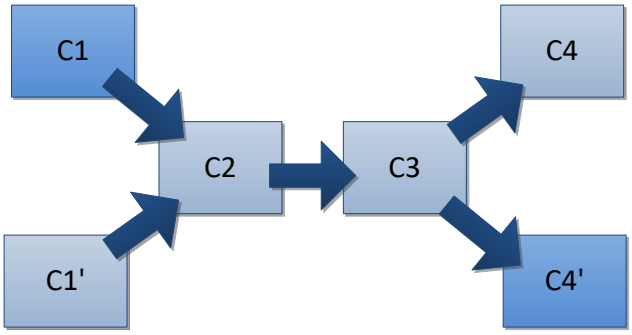

Figure 11 - The From - To PartyInfo

If the 'From' and 'To' are C1, C1' and C4, C4' respectively, the private keys of the certificates of C1 and C1' are stored in C2 and the public keys of the certificates of C4 and C4' are stored in C3. But if the 'From' and 'To' are C2 and C3, the private key of the certificate of C2 is stored in C2 and the public key of C3 is stored in C3.

From	To	Private key of	Private key stored in	Public key of	Public key stored in
C1, C1'	**C4, C4'**	**C1, C1'**	**C2**	**C4,C4'**	**C3**
C2	**C3**	**C2**	**C2**	**C3**	**C3**

- **Role:** This REQUIRED element identifies the authorized role of the Party sending or receiving the message.
- **Type:** This element indicates the domain of names to which the string in the content of the PartyId element belongs.

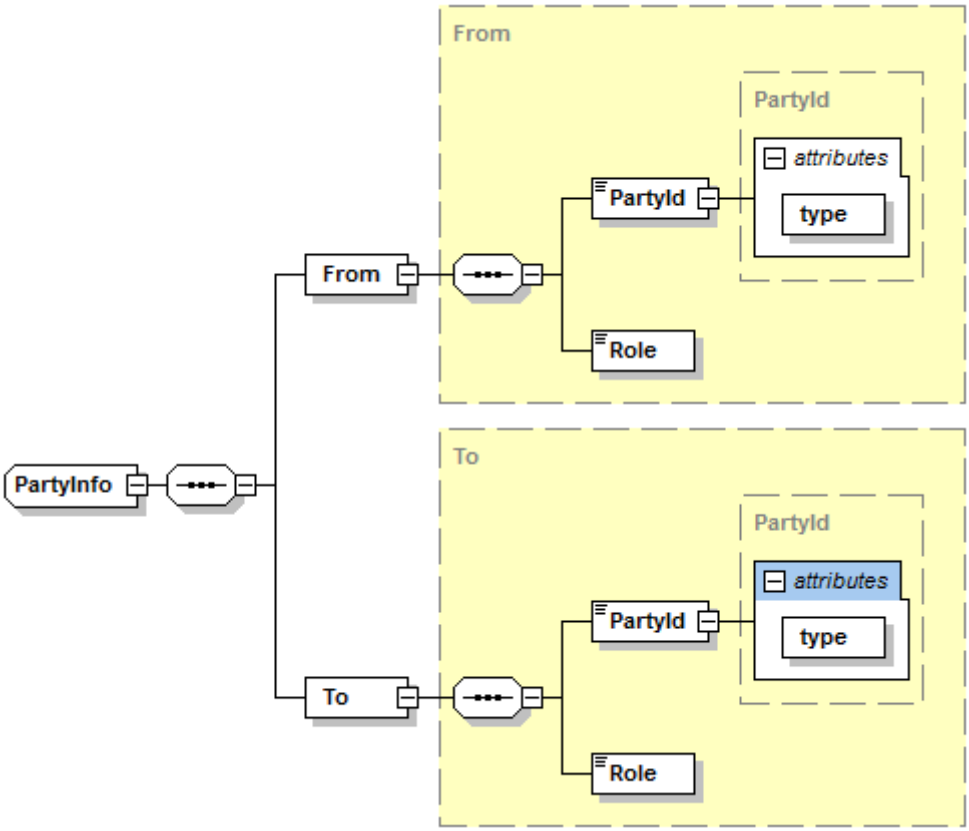

Figure 12 – PartyInfo type

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
From/PartyId	/*[local-name()='schema']/*[local-name()='complexType' and @name='From'] /*[local-name()='all']/*[local-name()='element' and @name='PartyId']	**Y**	**Max 1**	• The content of the PartyId element MUST be a URI if Type is not used. • The PartyID should be the same that is used in the PMode configuration: • It is a non-empty string. • Max length:255 characters • Configuration in PMode: PMode.Initiator.Party	*C2*
From/Role	/*[local-name()='schema']/*[local-name()='complexType' and @name='From'] /*[local-name()='all']/*[local-name()='el	**Y**	**Max 1**	• It is a non-empty string, • Max length:255 characters • Configuration in PMode: PMode.Initiator.Role	*http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/initiator*

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
	ement' and @name='Role']				
From/PartyType	/*[local-name()='schema']/*[local-name()='complexType' and @name='PartyId']//*[local-name()='attribute' and @name='type']	**N**	**Max 1**	• It is a non-empty string, • Max length:255 characters	*urn:oasis:names:tc:ebcore:partyid-type:unregistered*
To/PartyId	/*[local-name()='schema']/*[local-name()='complexType' and @name='To'] /*[local-name()='all']/*[local-name()='element' and @name='PartyId']	**N**	**Max 1**	• The content of the PartyId element MUST be a URI if PartyType is not used. • The PartyID should be the same that is used in the PMode configuration. • Max length:255 characters • Configuration in PMode: PMode.Responder.Party • If the AccessPoint at C2 is configured for Dynamic Discovery, the To/PartyId need not be specified by the backend; Domibus will identify the To/PartyId. In all other scenarios the backend C1 must specify the To/PartyId.	*C3*
To/Role	/*[local-name()='schema']/*[local-name()='complexType' and @name='To'] /*[local-name()='all']/*[local-name()='element' and @name='Role']	**N**	**Max 1**	• It is a non-empty string, • Max length:255 characters • Configuration in PMode: PMode.Responder.Role • If the AccessPoint at C2 is configured for Dynamic Discovery, the To/Role need not be specified by the backend; Domibus will identify the To/Role. In all other scenarios the backend C1 must specify the To/Role.	*http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/responder*
To/PartyType	/*[local-name()='schema']/*[local-name()='complexType	**N**	**Max 1**	• It is a non-empty string, • Max length:255 characters	*urn:oasis:names:tc:ebcore:partyid-type:unregistered*

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
	' and @name='PartyId']//*[local-name()='attribute' and @name='type']				

3.2.2.4. Messaging/UserMessage/CollaborationInfo

This REQUIRED element occurs once, and contains elements that facilitate collaboration between parties.

- The **AgreementRef** element is a string that identifies the entity or artifact governing the exchange of messages between the parties.
- **Service** SHOULD identify a set of related business transactions or other message exchanges in the context of a business process or use case.
- **Action** SHOULD identify the different types of business transactions or other message exchanges in the context of an identified Service.
- **ConversationId** element is a string identifying the set of related messages that make up a conversation between Parties. So, as defined in the eDelivery Specifications Library, it provides a more general way to associate a message with an ongoing conversation, without requiring a message to be a response to a single specific previous message but allowing update messages to existing conversations from both Sender and Receiver of the original message.

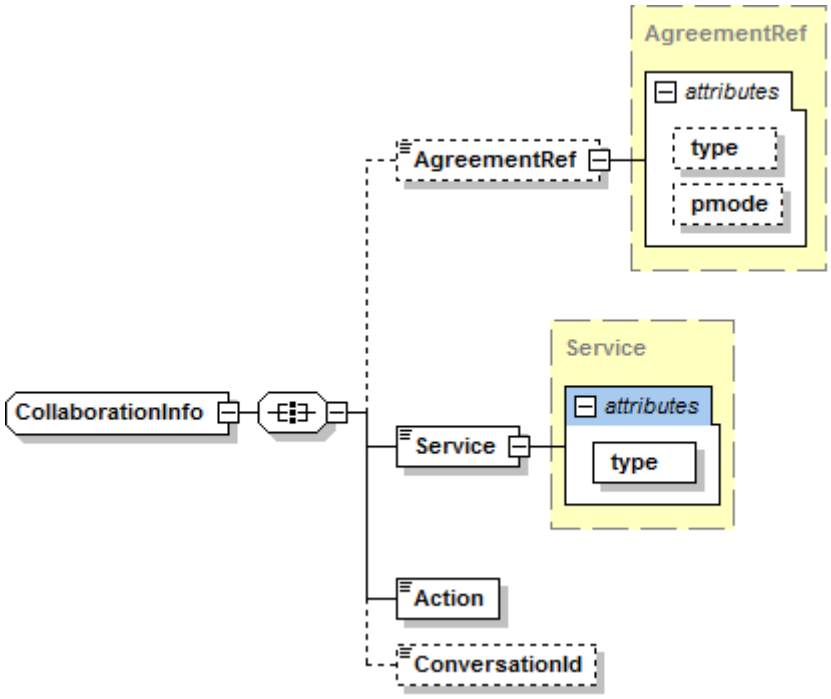

Figure 13 – CollaborationInfo type

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
AgreementRef	/*[local-name()='schema']/*[local-name()='complexType' and @name='CollaborationInfo'] /*[local-name()='all']/*[local-name()='element' and @name='AgreementRef']	**N**	**Max 1**	• It is a non-empty string. • The value of an AgreementRef element MUST be unique within a namespace mutually agreed by the two parties. This could be a concatenation of the From and To PartyId values, a URI containing the Internet domain name of one of the parties, or a namespace offered and managed by some other naming or registry service. It is RECOMMENDED that the AgreementRef be a URI. • AgreementRef is a string value that identifies the agreement that governs the exchange. The P-Mode under which the MSH operates for this message should be aligned with this agreement. • Max length:255 characters	*https://joinup.ec.europa.eu/*
agreementRef @type	/*[local-name()='schema']/*[local-name()='complexType' and @name='AgreementRef']/*[local-name()='simpleContent']/*[local-name()='extension']/*[local-name()='attribute' and @name='type']	**N**	**Max 1**	• It is a non-empty string. • Max length:255 characters • Indicates how the parties sending and receiving the message will interpret the value of the reference. There is no restriction on the value of the 'type' attribute. • If the 'type' attribute is not present, the content of the AgreementRef element MUST be a URI.	*MyServiceTypes*
agreementRef @pmode	/*[local-name()='schema']/*[local-name()='complexType' and @name='AgreementRef']/*[local-name()='simpleContent']/*[local-	**N**	**Max 1**	• It is a non-empty string. • Max length:255 characters • Allows for explicit association of a message with a P-Mode. When used, its value contains the PMode.ID parameter (i.e. the identifier for the P-Mode. This identifier is user-defined and optional, for the convenience of P-Mode management. It must uniquely identify the P-Mode among all P-Modes deployed on the same AP, and may be absent if the P-Mode is identified by other means, e.g.	*PurchaseOrderFromACME*

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
	name()='extension']/*[local-name()='attribute' and @name='type']			embedded in a larger structure that is itself identified, or has parameter values distinct from other P-Modes used on the same AP. If the ID is specified, the AgreementRef/@pmode attribute value is also expected to be set in associated messages.).	
Service	/*[local-name()='schema']/*[local-name()='complexType' and @name='CollaborationInfo'] /*[local-name()='all']/*[local-name()='element' and @name='Service']	**Y**	**Max 1**	• It is a non-empty string, • Max length:255 characters • Configuration in PMode: PMode[1].BusinessInfo.Service	*SupplierOrderProcessing*
Service@Type	/*[local-name()='schema']/*[local-name()='complexType' and @name='Service'] /*[local-name()='simpleContent']//*[local-name()='extension']/*[local-name()='attribute']	**N**		• Indicates how the parties sending and receiving the message will interpret the value of the element. • It is a non-empty string, • Max length:255 characters • Only optional if the service is untyped	
Action	/*[local-name()='schema']/*[local-name()='complexType' and @name='CollaborationInfo'] /*[local-name()='all']/*[local-	**Y**	**Max 1**	• It is a non-empty string. • Action SHALL be unique within the Service in which it is defined. • Max length:255 characters • Configuration in PMode: PMode[1].BusinessInfo.Action	*NewOrder*

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
	name()='element' and @name='Action']				
ConversationId	/*[local-name()='schema']/*[local-name()='complexType' and @name='CollaborationInfo'] /*[local-name()='all']/*[local-name()='element' and @name='ConversationId']	**N**	**Max 1**	• It is a non-empty string. Represents an immutable universally unique identifier (UUID). • Created randomly in the Receiving Access Point C2 • Max length:255 characters	*06689621-428e-48a4-86e6-4a86539363f5*

3.2.2.5. MessageProperties

This element is REQUIRED in the 4-corner model. It 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', as in the example below:

```
<ns:MessageProperties>
   <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1</ns:Property>
   <ns:Property name="finalRecipient">urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4</ns:Property>
</ns:MessageProperties>
```

The property value (e.g. urn:oasis:names:tc:ebcore:partyid-type:unregistered:C1) is limited to 1024 characters length. If this value is overpassed and the schema validation is enabled, an error message will appear and the message will not be submitted.

If the schema validation is not enabled and the value overpassed, an EbMS3Exception will be raised by the AP (Domibus) and the message will also not be submitted.

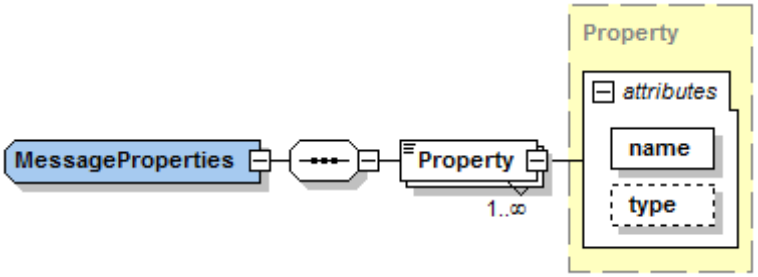

Figure 14 – MessageProperties type

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
Property name	/*[local-name()='schema']/*[local-name()='complexType' and @name='Property']//*[local-name()='attribute' and @name='name']	**Y**	**Max 1**	• It is a non-empty string. • Max length:255 characters • Configuration in PMode: PMode[1].BusinessInfo.Properties	*originalSender*
Property type	/*[local-name()='schema']/*[local-name()='complexType' and @name='Property']//*[local-name()='attribute' and @name='type']	**N**	**Max 1**	• It is a non-empty string • Max length: 255 characters	*String*

3.2.2.6. Messaging/UserMessage/ProcessingType

This OPTIONAL element specifies if the message should be sent directly or wait to be pulled by another Acces Point.

Possible values are PULL and PUSH.

By default, the value is PUSH.

Valid Example: `PULL`.

3.2.2.7. PayloadInfo

This REQUIRED element identifies payload data associated with the message. The payload themselves are carried in separate MIME parts, ***PartInfo*** elements reference the corresponding MIME parts by using the Content-ID value of those parts in their ***href*** attribute.

When a message with multiple payloads is submitted, the order of the corresponding PartInfo elements is preserved.

In any exchange involving a message that has a structured document payload (e.g XML, JSON) and any number of associated payloads, the structured document must be referenced by the first ***PartInfo*** element and it represents the leading payload part for business processing.

- **href:** This attribute has a value that is the Content-ID URI of the payload object referenced. The absence of the attribute href in the element PartInfo indicates that the payload part being referenced is the SOAP Body element itself.

  IMPORTANT
  Payloads are expected to be exchanged in separate MIME parts and not in the SOAP Body. Due to requirements from different domains, Domibus allows the sending of one structured payload in the SOAP Body. This payload is sent along by the Access Point, via the AS4 protocol, in the SOAP Body as well. This practice is not conformant to the eDelivery AS4 profile and therefore it is discouraged. It is recommended to leave the Soap Body always empty.

- **PartProperties:** This element contains a list of properties describing the payload. Every **Property** has a REQUIRED **@name** attribute. **@name** attribute with value *MimeType* is REQUIRED to identify the MIME type of the payload before compression was applied.

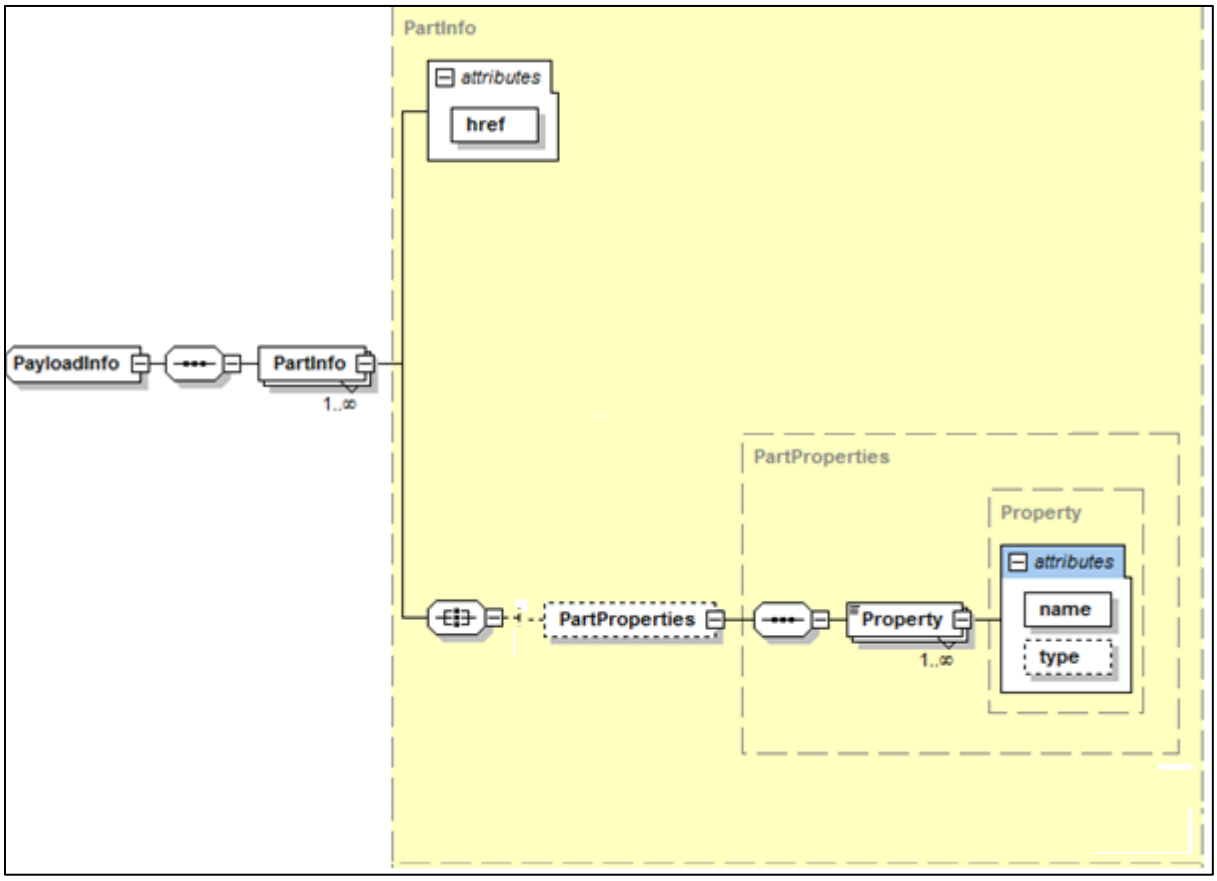

Figure 15 – PayloadInfo type

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
href	/*[local-name()='schema']/*[local-name()='complexType' and @name='PartInfo'] /*[local-name()='attribute' and @name='href']	**N**	**Max 1**	• Max length:255 characters	*cid:message*
PartProperties / Property/ MimeType	/*[local-name()='schema']/*[local-name()='complexType' and @name='PartProperties'] /*[local-name()='sequence']/*	**N**	**Max 1**	• Max length:255 characters • If the PMode compression is enabled this field is mandatory	*application/xml*

Description	Field (xpath)	Mandatory	Occurrences	Constraints	Valid example
	[local-name()='element' and @name='Property']				
PartProperties / Property/ Description	/*[local-name()='schema']/*[local-name()='complexType' and @name='PartProperties'] /*[local-name()='sequence']/*[local-name()='element' and @name='Property']	**N**	Max 1	• Max length:255 characters	*Message Payload*

3.3. Referencing Payloads

Files can be either:

1. Sent as attachment in the SOAP request,
2. Sent as part of the value of a payload element embedded in the SOAP body,
3. Referenced in the SOAP header. This is achieved by:
    a. Adding a **PartInfo** element within **PayloadInfo** properties. This is similar to sending a normal message except for the addition of the extra **PartInfo** special property.
    b. The payload value must be left empty and will be ignored if provided.
    c. Identify the **PartInfo** with href="cid:message" and provide a mimeType for the referenced file as well as the filepath reference pointing to where the file is stored.
    d. In the soap body, we need to provide a payload with payloadId="cid:message" and the appropriate contentType attribute.

Example:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/"
xmlns:eu="http://eu.domibus.wsplugin/">
  <soap:Header>
    <ns:Messaging>
      <ns:UserMessage mpc="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultMPC">
        <ns:PartyInfo>
          <ns:From>
            <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
blue</ns:PartyId>
            <ns:Role>http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/initiator</ns:Role>
          </ns:From>
          <ns:To>
            <ns:PartyId type="urn:oasis:names:tc:ebcore:partyid-type:unregistered">domibus-
red</ns:PartyId>
            <ns:Role>http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/responder</ns:Role>
          </ns:To>
        </ns:PartyInfo>
        <ns:CollaborationInfo>
          <ns:Service type="tc1">bdx:noprocess</ns:Service>
          <ns:Action>TC1Leg1</ns:Action>
        </ns:CollaborationInfo>
        <ns:MessageProperties>
          <ns:Property name="originalSender">urn:oasis:names:tc:ebcore:partyid-
type:unregistered:C1</ns:Property>
          <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="filepath">file:/some_filepath/some_file</ns:Property>
```

```
            </ns:PartProperties>
          </ns:PartInfo>
        </ns:PayloadInfo>
      </ns:UserMessage>
    </ns:Messaging>
  </soap:Header>
  <soap:Body>
    <eu:submitRequest>
          <payload payloadId="cid:message" contentType="text/xml">
          <value></value>
        </payload>
    </eu:submitRequest>
  </soap:Body>
</soap:Envelope>
```

To use payload file reference, Domibus must be configured to store the payloads on the file system. Please check the Domibus Administration Guide for more information on how to do this.

4. SECURITY

4.1. Authentication

The **Default WS Plugin** implements authentication and authorization. By default, the plugins security is disabled and all the methods of the **Default WS Plugin** can be called with no authentication credentials.

The **Defaul WS Plugin** supports 3 authentication methods:

- Basic Authentication
- X509 Certificates Authentication
- Blue Coat Authentication

***Remark**:*

Blue Coat is the name of the reverse proxy at the Commission. It forwards the request in HTTP with the certificate details inside the request ("Client-Cert" header key).

Basic authentication is the most common used method used for the **Default WS Plugin**. An existing user defined in the **Plugin User** UI page can authenticate with basic authentication and call any operation of the **Default WS Plugin**. More details on how to create plugin users used for basic authentication can be found in the **Domibus Administration Guide**, section **Plugin Users.**

The **Default WS Plugin** uses a custom interceptor **`eu.domibus.plugin.webService.impl.CustomAuthenticationInterceptor`** to intercept the incoming requests and perform authentication. Once the request is intercepted the **`CustomAuthenticationInterceptor`** delegates the authentication to the service **eu.domibus.ext.services.AuthenticationExtService** provided in the Plugin API.

4.2. Authorization

The Default WS Plugin uses the authorization mechanism described in the Plugin Cookbook document, section "Plugin Authorization" (check Domibus documentation per release on Domibus releases).

There are two default users already inserted in the database (make sure you already ran the migration scripts):

- *admin* has the role ROLE_ADMIN and *user* has the role ROLE_USER.

Roles:

ROLE_ADMIN has the permission to call:

- submitMessage with any value for originalSender property
- retrieveMessage (any message among messages notified to this plugin)

- listPendingMessages will list all pending messages for this plugin
- getStatus, getStatusWithAccessPointRole
- getMessageErrors, getMessageErrorsWithAccessPointRole
- markMessageAsDownloaded
- listPushFailedMessages - with any value for originalSender property
- rePushFailedMessages

ROLE_USER has the permission to call:

- submitMessage with originalSender equal to the originalUser
- retrieveMessage, only if finalRecipient equals the originalUser
- listPendingMessages, only messages with finalRecipient equal to the originalUser
- getStatus, getStatusWithAccessPointRole and getErrors for its own messages
- markMessageAsDownloaded
- listPushFailedMessages, only if finalRecipient equals the originalUser
- rePushFailedMessages

5. PLUGIN NOTIFICATIONS

Domibus core notifies the WS Plugin on the following events: MESSAGE_RECEIVED, MESSAGE_SEND_FAILURE, MESSAGE_RECEIVED_FAILURE, MESSAGE_SEND_SUCCESS, MESSAGE_STATUS_CHANGE.

The type of events received can be configured using the WS Plugin property *wsplugin.messages.notifications.* You will find that property in the file 'ws-plugin.properties' under \domibus\plugins\ in the Domibus configuration folder. More details can be found in the Plugin Cookbook.

6. MULTITENANCY

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

In Multitenancy mode the plugins security is activated by default, regardless of the value configured in **domibus.properties** for the **domibus.auth.unsecureLoginAllowed** property.

As a result, every request sent to Domibus via the **Default WS Plugin** needs to be authenticated and it will affect only the domain associated to the authenticated user. More information about the Default WS Plugin authentication can be found in section **4.1 Authentication.**

More details on how to create plugin users used for basic authentication can be found in the **Domibus Administration Guide**, section **Plugin Users.**

7. PUSH TO BACKEND

7.1. Introduction

Push to backend functionality produces soap calls towards a pre-defined URL triggered by Domibus events depending on the final recipient of the user message it concerns. For example, after successfully sending a message from C2 to C3, WS plugin might be notified by domibus with a notification type MESSAGE_SEND_SUCCESS.

The following chapters will describe the triggers (notifications to WS plugin), the configuration per final recipients (rules) and the different type of soap calls available (notifications).

The final recipient of a user message is the party to which the message is being sent to.

7.2. Notifications to plugin

- MESSAGE_RECEIVED

Domibus notifies the plugin when it receives successfully a UserMessage from C2. Domibus notifies the plugin using the java method receiveSuccess for each final recipient. The SOAP method “submitMessage” will be used to send the message to the back-end. If the property wsplugin.push.markAsDownloaded=false, the backend will be able to retrieve the same message multiple times and explicitly set the message status to downloaded.

- MESSAGE_SEND_FAILURE

Domibus notifies the plugin when it fails to send a UserMessage to C3. Domibus notifies the plugin using the java method messageSendFailed for each final recipient.

- MESSAGE_RECEIVED_FAILURE

Domibus notifies the plugin when it fails to receive a UserMessage from C2. Domibus notifies the plugin using the java method messageReceiveFailed for each final recipient.

- MESSAGE_SEND_SUCCESS

Domibus notifies the plugin when it sends successfully a UserMessage to C3. Domibus notifies the plugin using the java method deliverMessage for each final recipient.

- DELETE (not configurable)

Domibus notifies the plugin when a user message changes status. Domibus notifies the plugin using the java method messageStatusChanged for each final recipient.

- DELETE_BATCH (not configurable)

Domibus notifies the plugin when a user message changes status. Domibus notifies the plugin using the java method messageStatusChanged for each final recipient.

7.3. Rules configuration

In order to enable the push of notifications to a backend url, the property ‘wsplugin.push.enabled’ should be set to ‘true’ (default false). The notification push requests can optionally have basic autentication configured in the http Authorization header, if the properties ‘wsplugin.push.auth.username’ and ‘wsplugin.push.auth.password’ are defined.

Then, for each recipient, a set of properties should be set to properly configure a rule to follow. For example, ‘red1’ is an arbitrary rule name:

wsplugin.push.rules.red1=first rule description domibus-red

wsplugin.push.rules.red1.recipient=urn:oasis:names:tc:ebcore:partyid-type:unregistered:C4

wsplugin.push.rules.red1.endpoint=http://localhost:8080/backend

wsplugin.push.rules.red1.retry=1;5;CONSTANT

wsplugin.push.rules.red1.type=RECEIVE_SUCCESS,RECEIVE_FAIL

The detail of those properties can be found in §3.1-WS plugin configuration.

7.4. Notifications to backend

- Receive success

In case of a notification MESSAGE_RECEIVED from Domibus (message successfully received in C3 from C2), for a given final recipient, the soap method 'receiveSuccess' will be used to notify the backend.

- Receive fail

In case of a notification MESSAGE_RECEIVED_FAILURE from Domibus (message **not** successfully received in C3 from C2), for a given final recipient, the soap method 'receiveFailure' will be used to notify the backend.

- Send success

In case of a notification SEND_SUCCESS from Domibus (message successfully sent by C2 to C3), for a given final recipient, the soap method 'sendSuccess' will be used to notify the backend.

- Send Failure

In case of a notification SEND_FAIL from Domibus (message **not** successfully sent by C2 to C3), for a given final recipient, the soap method 'sendFailure' will be used to notify the backend.

- Message status change

In case of a notification MESSAGE_STATUS_CHANGE from Domibus, for a given final recipient, the soap method 'messageStatusChange' will be used to notify the backend.

- Submit Message

In case of a notification MESSAGE_RECEIVED from Domibus (message successfully received in C3 from C2), for a given final recipient, the soap method 'submitMessage' will be used to send the message to the backend.

- Delete message

In case of a message deletion in Domibus (retention worker), for a given final recipient, the soap method 'delete' will be used to send the notification to the backend.

- Delete messages (batch)

In case of messages deletion by batch in Domibus (retention worker), for a given final recipient, the soap method 'deleteBatch' will be used to send the notification to the backend.

8. BACKWARD COMPATIBILITY

8.1. Domibus 5.x: new Web Service

Starting with Domibus 5.0, the default WS plugin is using the endpoint **/wsplugin** and not **/backend** anymore. The previous endpoint is still available, but it has been deprecated. Users still using the deprecated version will notice in the logs log statements indicating the usage of a deprecated version.

For more information about the endpoint /backend, please refer to Domibus documentation on the Digital page.

The table below is describing the main differences between the two web-services implementations:

	Domibus 4.x and prior	Domibus 5.x and later
End point name	/backend	/wsplugin
wsdl	BackendService_1_1.wsdl	WebServicePlugin.wsdl
namespace	http://org.ecodex.backend/1_1/	http://eu.domibus.wsplugin/

8.2. Migration guide from the old WS Plugin interface to the new WS Plugin interface

In this section we will guide you step by step to migrate from the old WS Plugin interface ..services/backend?wsdl to the new interface ../services/wsplugin?wsdl.

Note

In this guide we will use the several keywords to refer to the target URL of your WS Plugin endpoint. For instance, for the following WS Plugin target URL https://localhost:8080/domibus/services/backend?wsdl we will use the following keywords:

- YOUR_HTTP_SCHEME to refer to the http scheme https. Possible values: http or https
- YOUR_HOST to refer to the domain name or the hostname
- YOUR _PORT to refer to the port number
- YOUR _DOMIBUS_VERSION to refer to the Domibus version you are using

8.2.1. Prerequisites

- A running version of Domibus version at least 5.0 accessible at YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus
- The WS Plugin corresponding to the running Domibus version is deployed

8.2.2. Migration

To migrate from the old WS Plugin to the new WS Plugin you need to follow these steps:

- Change in your backend application the target URL of the WS Plugin from YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/backend?wsdl to YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/wsplugin?wsdl
- in your backend application which is integrated with the WS Plugin, add the following Maven dependency to use the stubs for the new WS Plugin:

```
<dependency>
    <groupId>eu.domibus</groupId>
    <artifactId>domibus-default-ws-plugin-stubs</artifactId>
    <version>YOUR_DOMIBUS_VERSION</version>
</dependency>
```

NOTE

If you do not use Java and Maven, you need to generate your own stubs based on the WSDL file published under YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/wsplugin?wsdl

- The following changes are required in your backend application which integrates with the WS Plugin:
    - adapt the namespaces of your Soap requests from xmlns:_1="http://org.ecodex.backend/1_1/ to xmlns:eu="http://eu.domibus.wsplugin/"
    - replace eu.domibus.plugin.webService.generated.BackendInterface with eu.domibus.plugin.ws.generated.WebServicePluginInterface
    - replace eu.domibus.plugin.webService.generated.BackendService11 with eu.domibus.plugin.ws.generated.WebServicePlugin
    - replace the java imports from eu.domibus.plugin.webService.generated.* to eu.domibus.plugin.ws.generated.* and from eu.domibus.common.model.org.oasis_open.docs.ebxml_msg.ebms.v3_0.ns.core._200704.Messaging to eu.domibus.plugin.ws.generated.header.common.model.org.oasis_open.docs.ebxml_msg.ebms.v3_0.ns.core._200704.Messaging
    - adapt the listPendingMessages operation

FROM

```
ListPendingMessagesResponse listPendingMessagesResponse =
backendInterface.listPendingMessages("");
```

TO

```
ListPendingMessagesRequest request = new ListPendingMessagesRequest();

ListPendingMessagesResponse listPendingMessagesResponse =
backendInterface.listPendingMessages(request);
```

- change how you instantiate the WS Plugin port:

FROM

```
BackendService11 backendService = new BackendService11(new
URL("YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/backend?wsdl")
,  new QName("http://org.ecodex.backend/1_1/", "BackendService_1_1"));

BackendInterface backendPort = backendService.getBACKENDPORT();
```

TO

```
WebServicePlugin backendService = new WebServicePlugin(new
URL("YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/wsplugin?wsdl"
),  new QName("http://eu.domibus.wsplugin/", "WebServicePlugin"));

WebServicePluginInterface backendPort=
backendService.getWEBSERVICEPLUGINPORT();
```

An alternative to instantiate the WebServicePluginInterface is to use the already available class WebserviceClient. Here are the steps to use it:

- add the following Maven dependency

```
<dependency>
    <groupId>eu.domibus</groupId>
    <artifactId>Domibus-default-ws-plugin-client</artifactId>
    <version>YOUR_DOMIBUS_VERSION</version>
</dependency>
```

- Get the WebServicePluginInterface

```
WebserviceClient webserviceExample = new
WebserviceClient("YOUR_HTTP_SCHEME://YOUR_HOST:YOUR_PORT/domibus/services/w
splugin?wsdl", logMessages);

WebServicePluginInterface webServicePluginInterface =
webserviceExample.getPort();
```

9. ANNEXES

9.1. WS plugin interface message standards

AS4 does not define a maximum message size, though implementations will have practical limits based on available memory, disk, or database storage etc.

9.1.1. Errors codes table

Ebms error codes contained in the backend.wsdl:

Example:

```
<eb:Error origin="ebMS" category="Unpackaging"
shortDescription="InvalidHeader"
errorCode="EBMS:0009" severity="fatal">
<eb:Description xml:lang="en"> … </eb:Description>
</eb:Error>
```

Error Code	Short Description	Recommended Severity	Category Value	Description or Semantics
EBMS_0001	ValueNotRecognized	failure	Content	Although the message document is well formed and schema valid, some element/attribute contains a value that could not be recognized and therefore could not be used by the MSH.
EBMS_0002	FeatureNotSupported	warning	Content	Although the message document is well formed and schema valid, some element/attribute value cannot be processed as expected because the related feature is not supported by the MSH.
EBMS_0003	ValueInconsistent	failure	Content	Although the message document is well formed and schema valid, some element/attribute value is inconsistent either with the content of other element/attribute, or with the processing mode of the MSH, or with the normative requirements of the ebMS specification.
EBMS_0004	Other	failure	Content	
EBMS_0005	ConnectionFailure	failure	Communication	The MSH is experiencing temporary or permanent failure in trying to open a transport connection with a remote MSH.
EBMS_0006	EmptyMessagePartitionChannel	warning	Communication	There is no message available for pulling from this MPC at this moment.
EBMS_0007	MimeInconsistency	failure	Unpackaging	The use of MIME is not consistent with the required usage in this specification.
EBMS_0008	FeatureNotSupported	failure	Unpackaging	Although the message document is well formed and schema valid, the presence or absence of some element/ attribute is not consistent with the capability of the MSH, with respect to supported features.
EBMS_0009	InvalidHeader	failure	Unpackaging	The ebMS header is either not well formed as an XML document, or does

Error Code	Short Description	Recommended Severity	Category Value	Description or Semantics
				not conform to the ebMS packaging rules.
EBMS_0010	ProcessingModeMismatch	failure	Processing	The ebMS header or another header (e.g. reliability, security) expected by the MSH is not compatible with the expected content, based on the associated P-Mode.
EBMS_0011	ExternalPayloadError	failure	Content	The MSH is unable to resolve an external payload reference (i.e. a Part that is not contained within the ebMS Message, as identified by a PartInfo/href URI).
EBMS_0101	FailedAuthentication	failure	Processing	The signature in the Security header intended for the "ebms" SOAP actor, could not be validated by the Security module.
EBMS_0102	FailedDecryption	failure	Processing	The encrypted data reference the Security header intended for the "ebms" SOAP actor could not be decrypted by the Security Module.
EBMS_0103	PolicyNoncompliance	failure	Processing	The processor determined that the message's security methods, parameters, scope or other security policy-level requirements or agreements were not satisfied.
EBMS_0201	DysfunctionalReliability	failure	Processing	Some reliability function as implemented by the Reliability module, is not operational, or the reliability state associated with this message sequence is not valid.
EBMS_0202	DeliveryFailure	failure	Communication	Although the message was sent under Guaranteed delivery requirement, the Reliability module could not get assurance that the message was properly delivered, despite resending efforts.
EBMS_0301	MissingReceipt	failure	Communication	A Receipt has not been received for a message that was previously sent by the MSH generating this error
EBMS_0302	InvalidReceipt	failure	Communication	A Receipt has been received for a message that was previously sent by the MSH generating this error, but the content does not match the message content (e.g. some part has not been acknowledged, or the digest associated does not match the signature digest, for NRR).
EBMS_0303	DecompressionFailure	failure	Communication	An error occurred during the decompression
EBMS_0020	RoutingFailure	failure	Processing	An Intermediary MSH was unable to route an ebMS message and stopped processing the message.
EBMS_0021	MPCCapacityExceeded	failure	Processing	An entry in the routing function is matched that assigns the message to an MPC for pulling, but the intermediary MSH is unable to store the message with this MPC
EBMS_0022	MessagePersistenceTimeout	failure	Processing	An intermediary MSH has assigned the message to an MPC for pulling and has successfully stored it. However, the intermediary set a limit on the time it

Error Code	Short Description	Recommended Severity	Category Value	Description or Semantics
				was prepared to wait for the message to be pulled, and that limit has been reached.
EBMS_0023	MessageExpired	warning	Processing	An MSH has determined that the message is expired and will not attempt to forward or deliver it.
EBMS_0030	BundlingError	failure	Content	The structure of a received bundle is not in accordance with the bundling rules.
EBMS_0031	RelatedMessageFailed	failure	Processing	A message unit in a bundle was not processed because a related message unit in the bundle caused an error.
EBMS_0040	BadFragmentGroup	failure	Content	A fragment is received that relates to a group that was previously rejected.
EBMS_0041	DuplicateMessageSize	failure	Content	A fragment is received but more than one fragment message in a group of fragments specifies a value for this element.
EBMS_0042	DuplicateFragmentCount	failure	Content	A fragment is received but more than one fragment message in a group of fragments specifies a value for this element.
EBMS_0043	DuplicateMessageHeader	failure	Content	A fragment is received but more than one fragment message in a group of fragments specifies a value for this element.
EBMS_0044	DuplicateAction	failure	Content	A fragment is received but more than one fragment message in a group of fragments specifies a value for this element.
EBMS_0045	DuplicateCompressionInfo	failure	Content	A fragment is received but more than one fragment message in a group of fragments specifies a value for a compression element.
EBMS_0046	DuplicateFragment	failure	Content	A fragment is received but a previously received fragment message had the same values for GroupId and FragmentNum
EBMS_0047	BadFragmentStructure	failure	Unpackaging	The href attribute does not reference a valid MIME data part, MIME parts other than the fragment header and a data part are in the message, or the SOAP Body is not empty.
EBMS_0048	BadFragmentNum	failure	Content	An incoming message fragment has a a value greater than the known FragmentCount.
EBMS_0049	BadFragmentCount	failure	Content	A value is set for FragmentCount, but a previously received fragment had a greature value.
EBMS_0050	FragmentSizeExceeded	warning	Unpackaging	The size of the data part in a fragment message is greater than **Pmode[].Splitting.FragmentSize**
EBMS_0051	ReceiveIntervalExceeded	failure	Unpackaging	More time than **Pmode[].Splitting.JoinInterval** has passed since the first fragment was received but not all other fragments are received.
EBMS_0052	BadProperties	warning	Unpackaging	Message properties were present in the fragment SOAP header that were not specified in **Pmode[].Splitting.RoutingProperties**

Error Code	Short Description	Recommended Severity	Category Value	Description or Semantics
EBMS_0053	HeaderMismatch	failure	Unpackaging	The eb3:Message header copied to the fragment header does not match the eb3:Message header in the reassembled source message.
EBMS_0054	OutOfStorageSpace	failure	Unpackaging	Not enough disk space available to store all (expected) fragments of the group.
EBMS_0055	DecompressionError	failure	Processing	An error occurred while decompressing the reassembled message.
EBMS_0060	ResponseUsingAlternateMEP	Warning	Processing	A responding MSH indicates that it applies the alternate MEP binding to the response message.
EBMS_0065	InvalidXML	failure	Content	The XML could not be validated agains the corresponding xsd.

9.1.2. Web service WSDL

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:tns="http://eu.domibus.wsplugin/"
    xmlns:ns1="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/" name="WebServicePlugin"
    targetNamespace="http://eu.domibus.wsplugin/">

  <wsdl:types>

    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="xml.xsd"/>
    </schema>

    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://www.w3.org/2005/05/xmlmime" schemaLocation="xmlmime.xsd"/>
    </schema>

    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://www.w3.org/2003/05/soap-envelope" schemaLocation="envelope.xsd"/>
    </schema>

    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://eu.domibus.wsplugin/" schemaLocation="webservicePlugin-body.xsd"/>
    </schema>

    <schema xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/"
          schemaLocation="webservicePlugin-header.xsd"/>
    </schema>

  </wsdl:types>

  <wsdl:message name="getMessageErrors">
    <wsdl:part element="tns:getErrorsRequest" name="getErrorsRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessageErrorsWithAccessPointRole">
    <wsdl:part element="tns:getErrorsRequestWithAccessPointRole" name="getErrorsRequestWithAccessPointRole">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="retrieveMessage">
    <wsdl:part element="tns:retrieveMessageRequest" name="retrieveMessageRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="RetrieveMessageFault">
    <wsdl:part element="tns:FaultDetail" name="RetrieveMessageFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="markMessageAsDownloaded">
    <wsdl:part element="tns:markMessageAsDownloadedRequest" name="markMessageAsDownloadedRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="MarkMessageAsDownloadedFault">
    <wsdl:part element="tns:FaultDetail" name="MarkMessageAsDownloadedFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="markMessageAsDownloadedResponse">
    <wsdl:part element="tns:markMessageAsDownloadedResponse" name="markMessageAsDownloadedResponse">
    </wsdl:part>
    <wsdl:part element="ns1:Messaging" name="ebMSHeaderInfo">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPendingMessagesFault">
    <wsdl:part element="tns:FaultDetail" name="listPendingMessagesFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPushFailedMessagesFault">
```

```
    <wsdl:part element="tns:FaultDetail" name="listPushFailedMessagesFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="rePushFailedMessagesFault">
    <wsdl:part element="tns:FaultDetail" name="rePushFailedMessagesFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessageErrorsFault">
    <wsdl:part element="tns:FaultDetail" name="getMessageErrorsFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="retrieveMessageResponse">
    <wsdl:part element="tns:retrieveMessageResponse" name="retrieveMessageResponse">
    </wsdl:part>
    <wsdl:part element="ns1:Messaging" name="ebMSHeaderInfo">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPendingMessagesResponse">
    <wsdl:part element="tns:listPendingMessagesResponse" name="listPendingMessagesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPushFailedMessagesResponse">
    <wsdl:part element="tns:listPushFailedMessagesResponse" name="listPushFailedMessagesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="rePushFailedMessagesResponse">
  </wsdl:message>
  <wsdl:message name="getStatusResponse">
    <wsdl:part element="tns:getStatusResponse" name="getStatusResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPendingMessages">
    <wsdl:part element="tns:listPendingMessagesRequest" name="listPendingMessagesRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listPushFailedMessages">
    <wsdl:part element="tns:listPushFailedMessagesRequest" name="listPushFailedMessagesRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="rePushFailedMessages">
    <wsdl:part element="tns:rePushFailedMessagesRequest" name="rePushFailedMessagesRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getStatus">
    <wsdl:part element="tns:statusRequest" name="statusRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getStatusWithAccessPointRole">
    <wsdl:part element="tns:statusRequestWithAccessPointRole" name="statusRequestWithAccessPointRole">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="StatusFault">
    <wsdl:part element="tns:FaultDetail" name="StatusFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="submitMessage">
    <wsdl:part element="tns:submitRequest" name="submitRequest">
    </wsdl:part>
    <wsdl:part element="ns1:Messaging" name="ebMSHeaderInfo">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="submitMessageResponse">
    <wsdl:part element="tns:submitResponse" name="submitResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SubmitMessageFault">
    <wsdl:part element="tns:FaultDetail" name="SubmitMessageFault">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMessageErrorsResponse">
    <wsdl:part element="tns:getMessageErrorsResponse" name="getMessageErrorsResponse">
    </wsdl:part>
```

```
  </wsdl:message>
  <wsdl:portType name="WebServicePluginInterface">
    <wsdl:operation name="submitMessage">
      <wsdl:input message="tns:submitMessage" name="submitMessage">
      </wsdl:input>
      <wsdl:output message="tns:submitMessageResponse" name="submitMessageResponse">
      </wsdl:output>
      <wsdl:fault message="tns:SubmitMessageFault" name="SubmitMessageFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStatus">
      <wsdl:input message="tns:getStatus" name="getStatus">
      </wsdl:input>
      <wsdl:output message="tns:getStatusResponse" name="getStatusResponse">
      </wsdl:output>
      <wsdl:fault message="tns:StatusFault" name="StatusFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStatusWithAccessPointRole">
      <wsdl:input message="tns:getStatusWithAccessPointRole" name="getStatusWithAccessPointRole">
      </wsdl:input>
      <wsdl:output message="tns:getStatusResponse" name="getStatusResponse">
      </wsdl:output>
      <wsdl:fault message="tns:StatusFault" name="StatusFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="listPendingMessages">
      <wsdl:input message="tns:listPendingMessages" name="listPendingMessages">
      </wsdl:input>
      <wsdl:output message="tns:listPendingMessagesResponse" name="listPendingMessagesResponse">
      </wsdl:output>
      <wsdl:fault message="tns:listPendingMessagesFault" name="listPendingMessagesFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="listPushFailedMessages">
      <wsdl:input message="tns:listPushFailedMessages" name="listPushFailedMessages">
      </wsdl:input>
      <wsdl:output message="tns:listPushFailedMessagesResponse" name="listPushFailedMessagesResponse">
      </wsdl:output>
      <wsdl:fault message="tns:listPushFailedMessagesFault" name="listPushFailedMessagesFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="rePushFailedMessages">
      <wsdl:input message="tns:rePushFailedMessages" name="rePushFailedMessages">
      </wsdl:input>
      <wsdl:output message="tns:rePushFailedMessagesResponse" name="rePushFailedMessagesResponse">
      </wsdl:output>
      <wsdl:fault message="tns:rePushFailedMessagesFault" name="rePushFailedMessagesFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMessageErrors">
      <wsdl:input message="tns:getMessageErrors" name="getMessageErrors">
      </wsdl:input>
      <wsdl:output message="tns:getMessageErrorsResponse" name="getMessageErrorsResponse">
      </wsdl:output>
      <wsdl:fault message="tns:getMessageErrorsFault" name="getMessageErrorsFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMessageErrorsWithAccessPointRole">
      <wsdl:input message="tns:getMessageErrorsWithAccessPointRole" name="getMessageErrorsWithAccessPointRole">
      </wsdl:input>
      <wsdl:output message="tns:getMessageErrorsResponse" name="getMessageErrorsResponse">
      </wsdl:output>
      <wsdl:fault message="tns:getMessageErrorsFault" name="getMessageErrorsFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="retrieveMessage">
      <wsdl:input message="tns:retrieveMessage" name="retrieveMessage">
      </wsdl:input>
      <wsdl:output message="tns:retrieveMessageResponse" name="retrieveMessageResponse">
      </wsdl:output>
      <wsdl:fault message="tns:RetrieveMessageFault" name="RetrieveMessageFault">
```

```
        </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="markMessageAsDownloaded">
        <wsdl:input message="tns:markMessageAsDownloaded" name="markMessageAsDownloaded">
        </wsdl:input>
        <wsdl:output message="tns:markMessageAsDownloadedResponse" name="markMessageAsDownloadedResponse">
        </wsdl:output>
        <wsdl:fault message="tns:MarkMessageAsDownloadedFault" name="MarkMessageAsDownloadedFault">
        </wsdl:fault>
    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="WebServicePlugin_SoapBinding" type="tns:WebServicePluginInterface">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="submitMessage">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="submitMessage">
        <soap12:header message="tns:submitMessage" part="ebMSHeaderInfo" use="literal"/>
        <soap12:body parts="submitRequest" use="literal"/>
      </wsdl:input>
      <wsdl:output name="submitMessageResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SubmitMessageFault">
        <soap12:fault name="SubmitMessageFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStatus">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getStatus">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getStatusResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="StatusFault">
        <soap12:fault name="StatusFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="getStatusWithAccessPointRole">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getStatusWithAccessPointRole">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getStatusResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="StatusFault">
        <soap12:fault name="StatusFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMessageErrors">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getMessageErrors">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getMessageErrorsResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="getMessageErrorsFault">
        <soap12:fault name="getMessageErrorsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMessageErrorsWithAccessPointRole">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getMessageErrorsWithAccessPointRole">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getMessageErrorsResponse">
        <soap12:body use="literal"/>
```

```
        </wsdl:output>
        <wsdl:fault name="getMessageErrorsFault">
          <soap12:fault name="getMessageErrorsFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="listPendingMessages">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="listPendingMessages">
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="listPendingMessagesResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="listPendingMessagesFault">
          <soap12:fault name="listPendingMessagesFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="listPushFailedMessages">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="listPushFailedMessages">
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="listPushFailedMessagesResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="listPushFailedMessagesFault">
          <soap12:fault name="listPushFailedMessagesFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="rePushFailedMessages">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="rePushFailedMessages">
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="rePushFailedMessagesResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="rePushFailedMessagesFault">
          <soap12:fault name="rePushFailedMessagesFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="retrieveMessage">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="retrieveMessage">
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="retrieveMessageResponse">
          <soap12:header message="tns:retrieveMessageResponse" part="ebMSHeaderInfo" use="literal"/>
          <soap12:body parts="retrieveMessageResponse" use="literal"/>
        </wsdl:output>
        <wsdl:fault name="RetrieveMessageFault">
          <soap12:fault name="RetrieveMessageFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="markMessageAsDownloaded">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="markMessageAsDownloaded">
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="markMessageAsDownloadedResponse">
          <soap12:header message="tns:markMessageAsDownloadedResponse" part="ebMSHeaderInfo" use="literal"/>
          <soap12:body parts="markMessageAsDownloadedResponse" use="literal"/>
        </wsdl:output>
        <wsdl:fault name="MarkMessageAsDownloadedFault">
          <soap12:fault name="MarkMessageAsDownloadedFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="WebServicePlugin">
      <wsdl:port binding="tns:WebServicePlugin_SoapBinding" name="WEBSERVICEPLUGIN_PORT">
        <soap12:address location="http://localhost:8080/domibus/services/wsplugin"/>
      </wsdl:port>
```

```
    </wsdl:service>
</wsdl:definitions>
```

9.2. Backend interface message standard

9.2.1. *Backend WSDL*

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
      xmlns:tns="eu.domibus"
      name="backendService"
      targetNamespace="eu.domibus">

    <wsdl:types>

      <schema xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="eu.domibus" schemaLocation="BackendService.xsd"/>
      </schema>
    </wsdl:types>

    <wsdl:message name="submitMessage">
      <wsdl:part element="tns:submitMessage" name="submitMessage"/>
    </wsdl:message>
    <wsdl:message name="submitMessageResponse"/>
    <wsdl:message name="submitMessageFault">
      <wsdl:part element="tns:BackendFaultDetail" name="submitMessageFault"/>
    </wsdl:message>

    <wsdl:message name="sendSuccess">
      <wsdl:part element="tns:sendSuccess" name="sendSuccess"/>
    </wsdl:message>
    <wsdl:message name="sendSuccessResponse"/>
    <wsdl:message name="sendSuccessFault">
      <wsdl:part element="tns:BackendFaultDetail" name="sendSuccessFault"/>
    </wsdl:message>

    <wsdl:message name="sendFailure">
      <wsdl:part element="tns:sendFailure" name="sendFailure"/>
    </wsdl:message>
    <wsdl:message name="sendFailureResponse"/>
    <wsdl:message name="sendFailureFault">
      <wsdl:part element="tns:BackendFaultDetail" name="sendFailureFault"/>
    </wsdl:message>
    <wsdl:message name="receiveSuccess">
      <wsdl:part element="tns:receiveSuccess" name="receiveSuccess"/>
    </wsdl:message>
    <wsdl:message name="receiveSuccessResponse"/>
    <wsdl:message name="receiveSuccessFault">
      <wsdl:part element="tns:BackendFaultDetail" name="receiveSuccessFault"/>
    </wsdl:message>

    <wsdl:message name="receiveFailure">
      <wsdl:part element="tns:receiveFailure" name="receiveFailure"/>
    </wsdl:message>
    <wsdl:message name="receiveFailureResponse"/>
    <wsdl:message name="receiveFailureFault">
      <wsdl:part element="tns:BackendFaultDetail" name="receiveFailureFault"/>
    </wsdl:message>

    <wsdl:message name="delete">
      <wsdl:part element="tns:delete" name="delete"/>
    </wsdl:message>
    <wsdl:message name="deleteResponse"/>
    <wsdl:message name="deleteFault">
```

```
    <wsdl:part element="tns:BackendFaultDetail" name="deleteFault"/>
  </wsdl:message>

  <wsdl:message name="deleteBatch">
    <wsdl:part element="tns:deleteBatch" name="deleteBatch"/>
  </wsdl:message>
  <wsdl:message name="deleteBatchResponse"/>
  <wsdl:message name="deleteBatchFault">
    <wsdl:part element="tns:BackendFaultDetail" name="deleteBatchFault"/>
  </wsdl:message>

  <wsdl:message name="messageStatusChange">
    <wsdl:part element="tns:messageStatusChange" name="messageStatusChange"/>
  </wsdl:message>
  <wsdl:message name="messageStatusChangeResponse"/>
  <wsdl:message name="messageStatusChangeFault">
    <wsdl:part element="tns:BackendFaultDetail" name="messageStatusChangeFault"/>
  </wsdl:message>

  <wsdl:portType name="BackendInterface">
    <wsdl:operation name="submitMessage">
      <wsdl:input message="tns:submitMessage" name="submitMessage">
      </wsdl:input>
      <wsdl:output message="tns:submitMessageResponse" name="submitMessageResponse">
      </wsdl:output>
      <wsdl:fault message="tns:submitMessageFault" name="submitMessageFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="sendSuccess">
      <wsdl:input message="tns:sendSuccess" name="sendSuccess">
      </wsdl:input>
      <wsdl:output message="tns:sendSuccessResponse" name="sendSuccessResponse">
      </wsdl:output>
      <wsdl:fault message="tns:sendSuccessFault" name="sendSuccessFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="sendFailure">
      <wsdl:input message="tns:sendFailure" name="sendFailure">
      </wsdl:input>
      <wsdl:output message="tns:sendFailureResponse" name="sendFailureResponse">
      </wsdl:output>
      <wsdl:fault message="tns:sendFailureFault" name="sendFailureFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="receiveSuccess">
      <wsdl:input message="tns:receiveSuccess" name="receiveSuccess">
      </wsdl:input>
      <wsdl:output message="tns:receiveSuccessResponse" name="receiveSuccessResponse">
      </wsdl:output>
      <wsdl:fault message="tns:receiveSuccessFault" name="receiveSuccessFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="receiveFailure">
      <wsdl:input message="tns:receiveFailure" name="receiveFailure">
      </wsdl:input>
      <wsdl:output message="tns:receiveFailureResponse" name="receiveFailureResponse">
      </wsdl:output>
      <wsdl:fault message="tns:receiveFailureFault" name="receiveFailureFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="delete">
      <wsdl:input message="tns:delete" name="delete">
      </wsdl:input>
      <wsdl:output message="tns:deleteResponse" name="deleteResponse">
      </wsdl:output>
      <wsdl:fault message="tns:deleteFault" name="deleteFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteBatch">
      <wsdl:input message="tns:deleteBatch" name="deleteBatch">
      </wsdl:input>
      <wsdl:output message="tns:deleteBatchResponse" name="deleteBatchResponse">
```

```
      </wsdl:output>
      <wsdl:fault message="tns:deleteBatchFault" name="deleteBatchFault">
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="messageStatusChange">
      <wsdl:input message="tns:messageStatusChange" name="messageStatusChange">
      </wsdl:input>
      <wsdl:output message="tns:messageStatusChangeResponse" name="messageStatusChangeResponse">
      </wsdl:output>
      <wsdl:fault message="tns:messageStatusChangeFault" name="messageStatusChangeFault">
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="BackendServiceSoapBinding" type="tns:BackendInterface">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="submitMessage">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="submitMessage">
        <soap12:body parts="submitMessage" use="literal"/>
      </wsdl:input>
      <wsdl:output name="submitMessageResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="submitMessageFault">
        <soap12:fault name="submitMessageFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="sendSuccess">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="sendSuccess">
        <soap12:body parts="sendSuccess" use="literal"/>
      </wsdl:input>
      <wsdl:output name="sendSuccessResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="sendSuccessFault">
        <soap12:fault name="sendSuccessFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="sendFailure">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="sendFailure">
        <soap12:body parts="sendFailure" use="literal"/>
      </wsdl:input>
      <wsdl:output name="sendFailureResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="sendFailureFault">
        <soap12:fault name="sendFailureFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="receiveSuccess">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="receiveSuccess">
        <soap12:body parts="receiveSuccess" use="literal"/>
      </wsdl:input>
      <wsdl:output name="receiveSuccessResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="receiveSuccessFault">
        <soap12:fault name="receiveSuccessFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="receiveFailure">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="receiveFailure">
        <soap12:body parts="receiveFailure" use="literal"/>
      </wsdl:input>
      <wsdl:output name="receiveFailureResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
```

```
        <wsdl:fault name="receiveFailureFault">
          <soap12:fault name="receiveFailureFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="delete">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="delete">
          <soap12:body parts="delete" use="literal"/>
        </wsdl:input>
        <wsdl:output name="deleteResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="deleteFault">
          <soap12:fault name="deleteFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="deleteBatch">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="deleteBatch">
          <soap12:body parts="deleteBatch" use="literal"/>
        </wsdl:input>
        <wsdl:output name="deleteBatchResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="deleteBatchFault">
          <soap12:fault name="deleteBatchFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="messageStatusChange">
        <soap12:operation soapAction="" style="document"/>
        <wsdl:input name="messageStatusChange">
          <soap12:body parts="messageStatusChange" use="literal"/>
        </wsdl:input>
        <wsdl:output name="messageStatusChangeResponse">
          <soap12:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="messageStatusChangeFault">
          <soap12:fault name="messageStatusChangeFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="BackendService">
      <wsdl:port binding="tns:BackendServiceSoapBinding" name="BACKEND_PORT">
        <soap12:address location="http://localhost:8080/backend"/>
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

9.2.2. Backend .xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
        xmlns:tns="eu.domibus"
        targetNamespace="eu.domibus">

    <xsd:simpleType name="max255-non-empty-string">
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
        <xsd:maxLength value="255"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="messageStatus">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="READY_TO_PULL"/>
        <xsd:enumeration value="SEND_ENQUEUED"/>
        <xsd:enumeration value="WAITING_FOR_RECEIPT"/>
```

```
      <xsd:enumeration value="ACKNOWLEDGED"/>
      <xsd:enumeration value="SEND_FAILURE"/>
      <xsd:enumeration value="NOT_FOUND"/>
      <xsd:enumeration value="WAITING_FOR_RETRY"/>
      <xsd:enumeration value="RECEIVED"/>
      <xsd:enumeration value="DELETED"/>
      <xsd:enumeration value="DOWNLOADED"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="LargePayloadType">
    <xsd:sequence>
      <xsd:element name="value" type="xsd:base64Binary" xmime:expectedContentTypes="application/octet-stream"/>
    </xsd:sequence>
    <xsd:attribute name="payloadId" type="xsd:token"/>
    <xsd:attribute name="contentType" type="xsd:string"/>
    <xsd:attribute name="mimeType" type="xsd:string"/>
    <xsd:attribute name="payloadName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:element name="submitMessage">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
        <xsd:element name="finalRecipient" type="tns:max255-non-empty-string"/>
        <xsd:element name="originalSender" type="tns:max255-non-empty-string"/>
        <xsd:element minOccurs="0" name="bodyload" type="tns:LargePayloadType"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="payload" type="tns:LargePayloadType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="BackendFaultDetail">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="code" type="xsd:string"/>
        <xsd:element name="message" nillable="true" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="sendSuccess">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="sendFailure">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="receiveSuccess">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="receiveFailure">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="delete">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="deleteBatch">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="messageIds" nillable="true" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="messageStatusChange">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="messageID" type="tns:max255-non-empty-string"/>
        <xsd:element name="messageStatus" type="tns:messageStatus"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

10. List of Figures

List of Tables

11. CONTACT INFORMATION

eDelivery Support Team

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

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