EUROPEAN COMMISSION

DIRECTORATE-GENERAL INFORMATICS

CEF eDelivery

Quick Start Guide

CONTENTS

INTRODUCTION

CEF e-Delivery provides a set of components to exchange messages over the internet using B2B protocols. See the document concerning the ***Introduction to the Connecting Europe Facility eDelivery building block*** also included in this package for more information.

In this particular *static* deployment context, the full set of components (e.g. dynamic discovery, connector) is not required. Business Registers cannot communicate directly with one another. Business Registers must always communicate through the European Central Platform using B2B, AS4 protocol (no need for AS2).

Therefore, this specific release *cipa-edelivery-distribution-3.2.0-as4-jboss* provides only an AS4 gateway (CEF e-Delivery component called *domibus*) running on a JBoss application server and using a MySQL database for data persistence.

PURPOSE OF THIS GUIDE

In this document, you will find instructions to cover the deployment scenario as illustrated in the figure below. In other words, we will guide you to setup two JBoss standalone instances connected on two separate machines to exchange B2B documents securely over AS4 by:

- Deploying and configuring both JBoss instances (A and B)

- Configuring processing mode files for both AS4 gateways

- Using the provided AS4 gateways certificates

- Setup the instances A and B for running test cases (see Testing section)
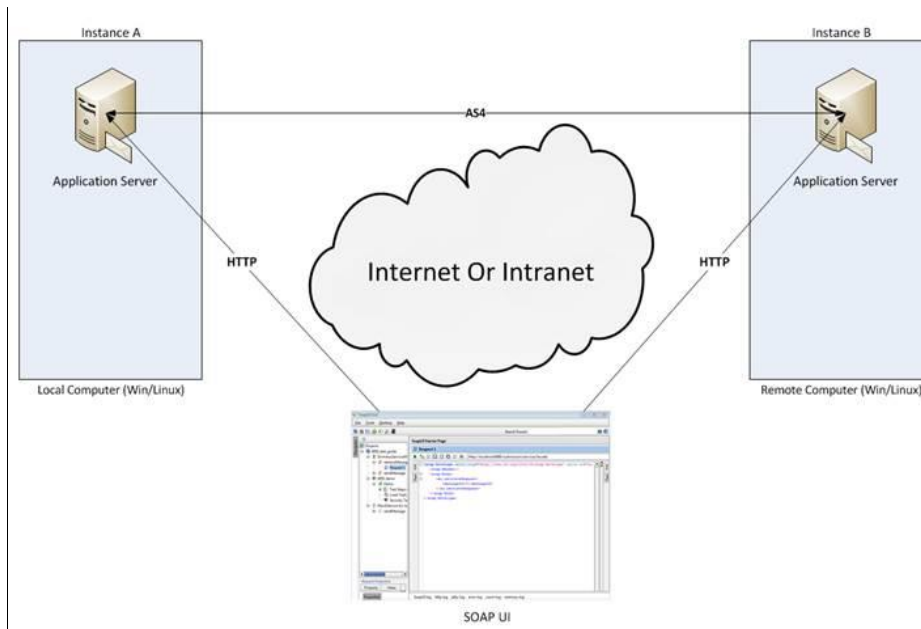
Installation on two different machines

Remarks:

o *The same procedure can be extended to a third (or more) instance.*

o *This guide does not cover the preliminary network configuration allowing communication between separate networks (i.e. infrastructure firewall/Proxy setup).*

PREREQUISITES

- Java runtime environment (JRE), version 7:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html

- JCE Unlimited Strength Policy files, for JRE7:
  http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html
  Copy the jar files from the extracted zip to <JRE_HOME>\lib\security.

- MySQL database server listening on the default port 3306:
  http://dev.mysql.com/downloads/windows/installer/5.6.html

Please install the above software on your host machine. For further information and installation details, we kindly advise you to refer to the manufacturers' websites.

Remark:

> *Please ensure that environment variable JAVA_HOME is set to JRE7 but also that the path for JRE7 and My SQL are set to their respective bin directory.*
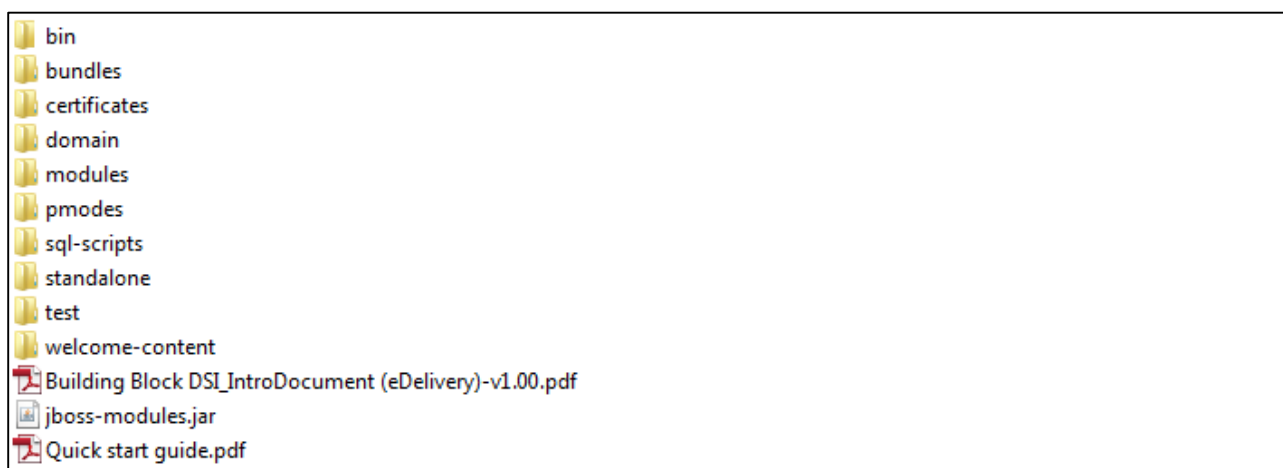
CONFIGURE YOUR ENVIRONMENT

1.1. Package Overview

Download the CIPA eDelivery Distribution from the shared drive:

https://joinup.ec.europa.eu/nexus/content/repositories/releases/eu/europa/ec/cipa/cef-edelivery-distribution/3.2.0-alpha-1/cef-edelivery-distribution-3.2.0-alpha-1-as4-jboss.zip

This package has the following structure and includes 3.2.0 Release Notes:

Package content

- **<CEF-eDelivery path>/bin** contains the executable batch file (windows) and shell script (Linux) required to launch the JBoss instance.

- **<CEF-eDelivery path>/certificates** contains a keystore (including private keys of instance A and instance B) and a trust store (including public keys of *Instance A* and *Instance B*) that can be used by both instances. For this test release, each instance uses self-signed certificates. Please refer to Annex 5 for more information about AS4 security.

- **<CEF-eDelivery path>/standalone** contains:

- o *Configuration* folder where you will find the *standalone.xml* (used to administer your JBOSS)
- o *Log* folder where the logs are stored
- o *Deployments* folder where the two WAR files are stored

- *domibus-app.war* which provides a reference implementation to allow backend(s) to interact with domibus AS4 gateways.

- *domibus.war* which is an ebMS3 gateway based on the e-SENS AS4 profile.

- **<CEF-eDelivery path>/modules/eu/domibus/main** contains domibus configuration files.

- **<CEF-eDelivery path>/pmodes** contains an AS4 processing mode (pMode-configuration.xml) pre-configured to use compression, payload encryption, message signing and non-repudiation. The provided PMode file must be updated for both instances.

- **<CEF-eDelivery path>/sql-scripts** contains the required application SQL code that needs to be executed on the MySQL database.

- **<CEF-eDelivery path>/test** contains a SOAP UI test project.

1.2. JBoss Standalone Instance

As described in the purpose of this guide, we need to configure two instances running on two separate machines. Therefore, the procedure below would need to be applied on both machines *Hostname A* (local machine) and *Hostname B* (remote machine). Please note that an extra step is only required for *Hostname B*.

1. Extract the zip file containing the installation package of the CIPA eDelivery to a location on your physical machine, which we will refer to in this document as your *< eDelivery installation path >*.

2. Open a command prompt and navigate to this directory:

    < eDelivery installation path >\sql-scripts.

3. Execute the following commands in the command prompt :

```
mysql -h localhost -u root --password=root -e "drop schema if exists
edelivery;create schema edelivery;alter database edelivery
charset=utf8; create user edelivery identified by 'edelivery';grant
all on edelivery.* to edelivery;"
```

```
mysql -h localhost -u root --password=root edelivery < create-mysql.sql
```

Remarks:

- o *If you are using Windows, make sure to have mysql.exe added to your PATH variable.*

- o If you are using a different schema, please adapt your commands but also edit the *standalone.xml* file and replace the line ***jdbc:mysql://localhost:3306/edelivery*** with ***jdbc:mysql://localhost:3306/"yourSchemaName"***

4.	Update the default properties of *my-default.ini* (Windows) or *my.cnf* (Linux).

a.	max_allowed_packet property

```
# The maximum size of one packet or any generated or intermediate string, or any parameter sent by the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet = 512M
```

b.	innodb_log_file_size property

```
# # Size of each log file in a log group. You should set the combined size
# of log files to about 25%-100% of your buffer pool size to avoid
# unneeded buffer pool flush activity on log file overwrite. However,# note that larger logfile size will increase the time needed for the recovery process
innodb_log_file_size = 5120M
```

5.	Restart MySQL service:

MSSQLServerADHelper 100		SQL Active...	Stopped	N/A
MySQL56	2708	MySQL56	Running	N/A
napagent		Network A...	Stopped	NetworkSe...

MySQL service

6.	This step is only required for **Hostname B**.

Edit **<CEF-eDelivery path>/modules/eu/europa/ec/cipa/configuration/main/domibus/ domibus-configuration.xml** and replace *instanceA* with *instanceB* as shown below:

```
<!-- The default keystore alias to use, if none is specified. -->
<prop key="org.apache.ws.security.crypto.merlin.keystore.alias">instanceB</prop>
```

7.	You can now start the JBoss standalone instance on your computer.

Execute:

a.	bin/standalone.sh (for Linux)

b.	bin/standalone.bat (for windows)
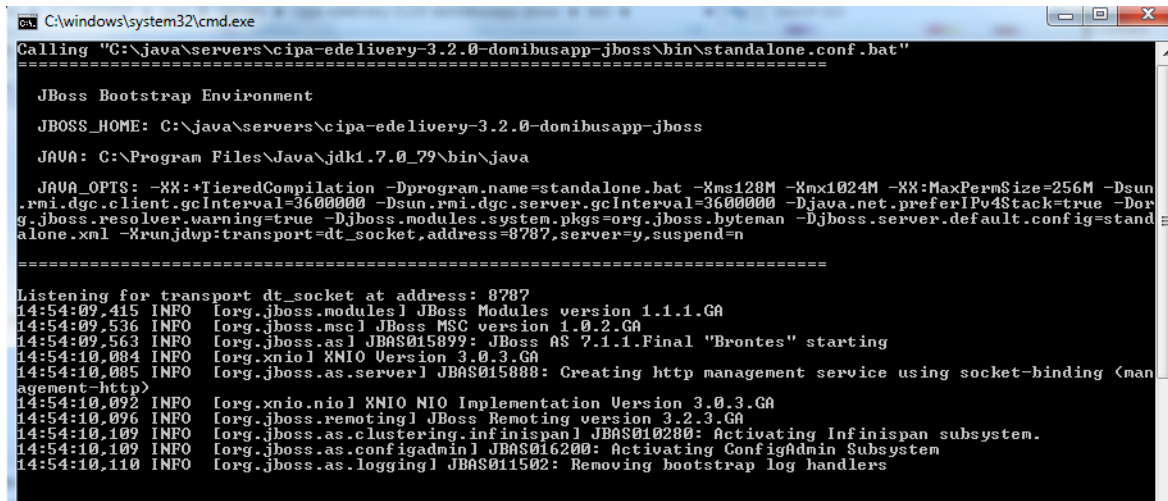
Remark:

In the default configuration, the JVM allocation memory is set to 1024. This parameter has to be reduced if the system cannot reserve enough space during the initializing of the application server (edit the parameter Xmx in the file "bin/standalone.conf.bat"):

```
rem # JVM memory allocation pool parameters - modify as appropriate.
set "JAVA_OPTS=-Xms128M –Xmx1024M -XX:MaxPermSize=256M"
```
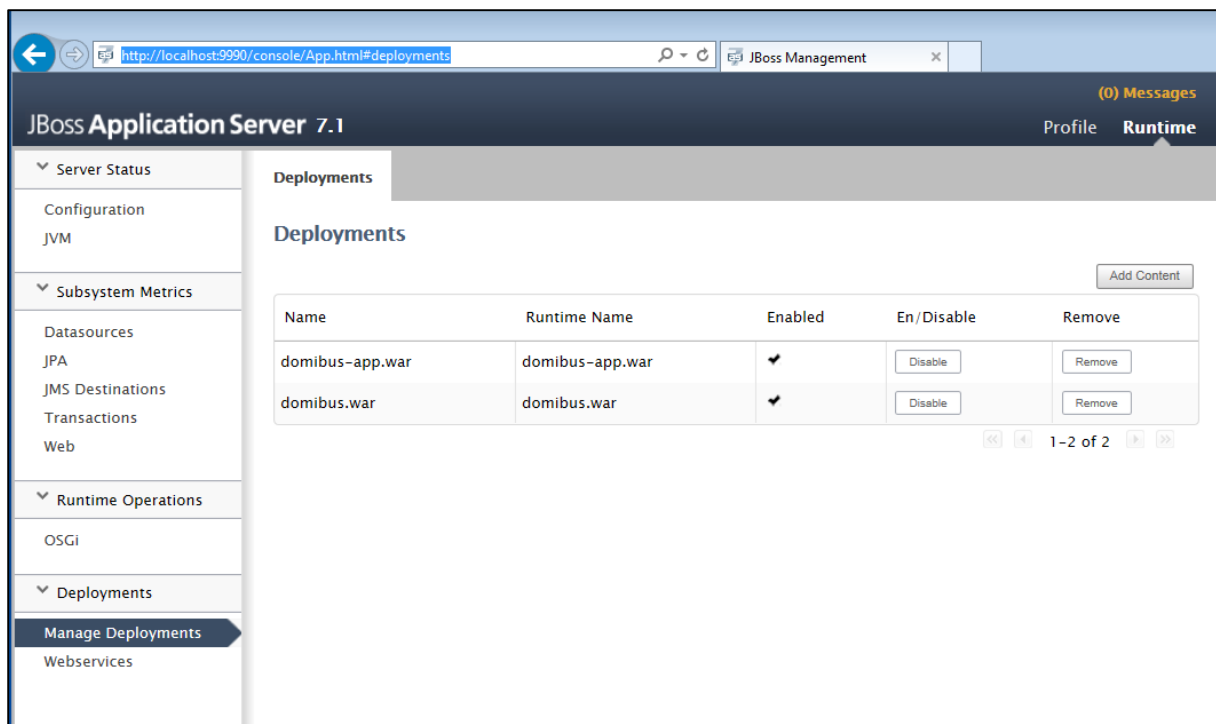
Expected result:

Boss instance up and running

Remarks:

- o *The result of the deployment can be check on the JBOSS admin console (Username: **manager**, Password: **manage**)*

- o *If the application server does not start properly, more details about the encountered errors can be found in the log files. Refer to **<CEF-eDelivery path>/standalone/log/***

8. Once the application server is started, you can ensure that this server is operational by displaying the administration dashboard (*http://localhost:8080/domibus/home*) in your browser as below:

Domibus administration page

Remarks:

- o *To allow the remote application to send a message to this machine, you would need to create a dedicated rule (to allow this port) from your local firewall (cf. annex "[Firewall Settings](#)")*

- o *If you intend to install both instances on the same server, you will need to change instance B ports to avoid conflicts and database schema 'edelivery' before starting the server.*

9. Edit *< eDelivery installation path >\pmodes\pMode-configuration.xml* and replace *UndefinedHostnameA* and *UndefinedHostnameB* with their real names as shown below:
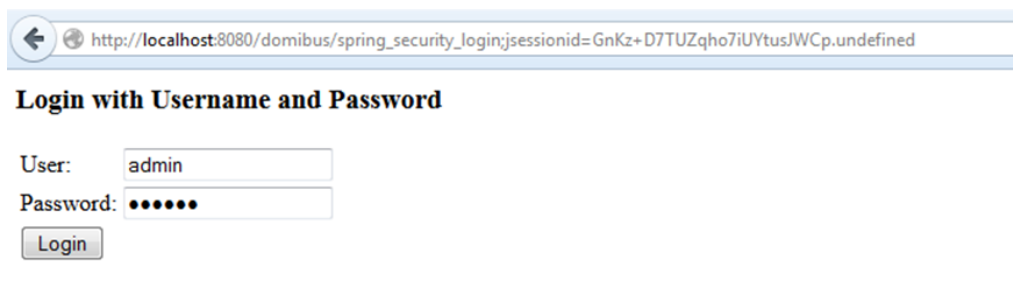
```xml
<parties>
    <partyIdTypes>
        <partyIdType name="exampleType" value="http://www.domibus.eu/exampleType"/>
    </partyIdTypes>
    <party name="instanceA"
            endpoint="http://UndefinedHostnameA:8080/domibus/services/msh"

        <identifier partyId="instanceAId1" partyIdType="exampleType"/>
        <identifier partyId="instanceAId2" partyIdType="exampleType"/>
    </party>
    <party name="instanceB"
            endpoint="http://UndefinedHostnameB:8080/domibus/services/msh"

        <identifier partyId="instanceBId1" partyIdType="exampleType"/>
        <identifier partyId="instanceBId2" partyIdType="exampleType"/>
    </party>
</parties>
```

PMode view
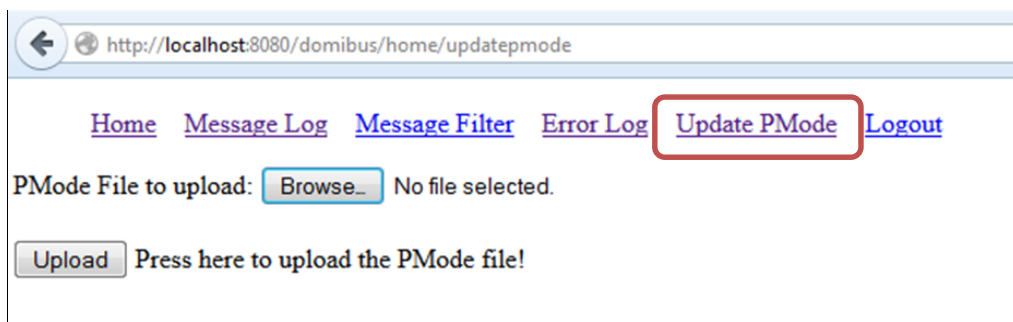
For more details about the provided PMode, please see Annex 4.

10. Upload the PMode file on both instances:

a. To upload a PMode XML file, connect to the administration dashboard using your credentials (by default: login = **admin**; password = **123456**) to http://localhost:8080/domibus/home

Login to administration dashboard

b. Click on the *Upload PMode* tab:

PMode update

c. Select your PMode from "< eDelivery installation path >\pmodes" and click on *Upload*:
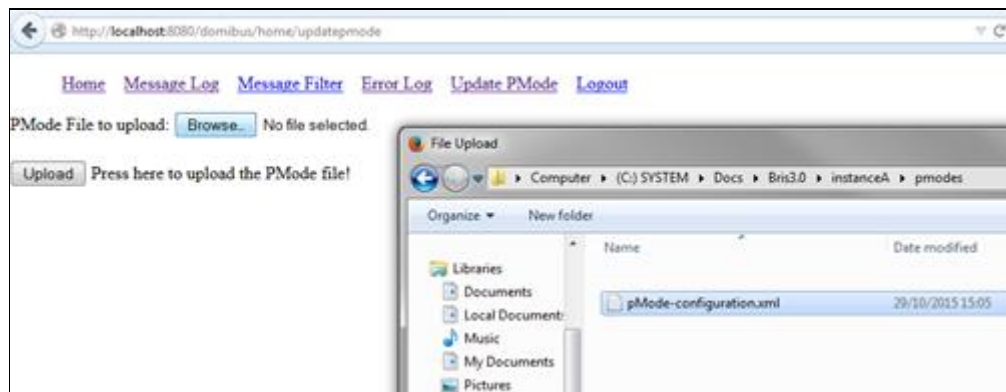
PMode uploading

Now your JBoss instances are running and ready to send or receive messages.
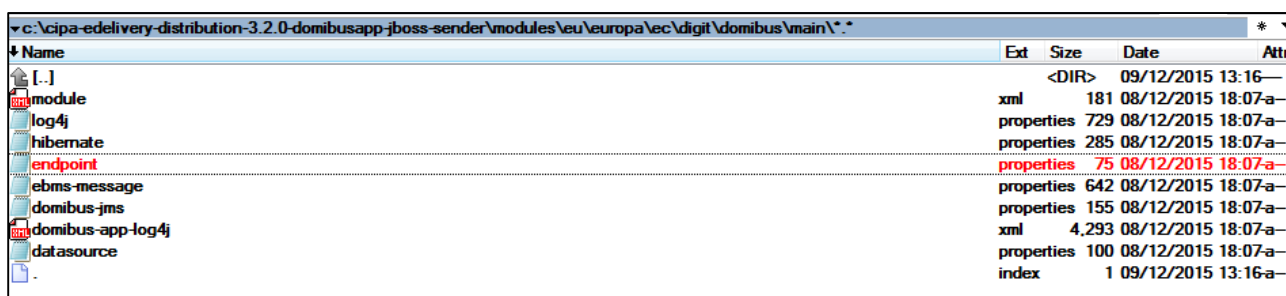
TESTING

As explained in the Release Notes document and to facilitate testing, we have developed a Reference Web Service endpoint to illustrate how participants can connect and interact with the AS4 gateway to send messages.

In addition, each AS4 gateway is pre-configured to push all received messages to a default *Web Service endpoint* (based on the same WSDL as is used in the Reference Web Service). This pre-configured endpoint is defined in the <CEF-eDelivery path>/modules/eu/domibus/main/domibus-configuration.xml.

Remark:

> *Make sure to edit **endpoint.properties** and configure the endpoint.*

| `basic.endpoint.uri` | = | http://localhost:8089/mockBasicDomibus?WSDL |

For full instructions on how to send messages, please see the *Test guide.pdf* in the test folder.

Remark:

> *If you encounter connection timeouts on the test you should increase the Socket Timeout setting of SoapUI. This can be done in File -> Preferences.*

Please refer to the *Known Limitations* section in the *Release Notes* for any restrictions in the default configuration.

ANNEX 1 PARAMETERS

Parameters	Local instance (Instance A)	Remote instance (Instance B)
Host Name	Host Name A	Host Name B
Database	MySQL database	MySQL database
Administrator Page	Username: *admin* Password: *123456* http://localhost:8080/domibus/home	Username: *admin* Password: *123456* http://localhost:8080/domibus/home
Database Schema	edelivery	edelivery
Database connector	Username: *edelivery* Password: *edelivery* dbc:mysql://localhost:3306/edelivery*	Username: *edelivery* Password: *edelivery* jdbc:mysql://localhost: 3306/edelivery
DB username/password	edelivery/edelivery	edelivery/edelivery
PModes XML files	pmodes/pMode-configuration.xml	pmodes/pMode-configuration.xml
Keystore location	certificates/keystore.jks	certificates/keystore.jks
Keystore Alias Name	*instanceA*	*instanceB* to be edited in *domibus-configuration.xml*
JBoss Admin console URL	Username: *manager* Password: *manage* http://localhost:9990/console/index.html	Username: *manager* Password: *manage* http://localhost:9990/console/index.html

* *localhost* represents the server name that hosts the database and the application server for their respective instance.

ANNEX 2 FIREWALL SETTINGS

The firewall settings may prevent you from exchanging messages between your local and remote JBoss instances.

To test the status of a port, run the command *telnet <server_ip> <port>*

The following ports must be opened on both machines A and B (TCP protocol):

•   8080 (HTTP port)

•   5445 (JMS messaging)

•   5455 (JMS messaging throughput)

•   3306 (MySQL port)

•   9990 (JBoss admin console)

This is how you can open a port on the Windows Firewall

1.   Click on **Start** then on **Control Panel**

2.   Click on **Windows Firewall** and then click on **Advanced Settings**.

3.   Right click on **Inbound Rules** then on **New Rule**:
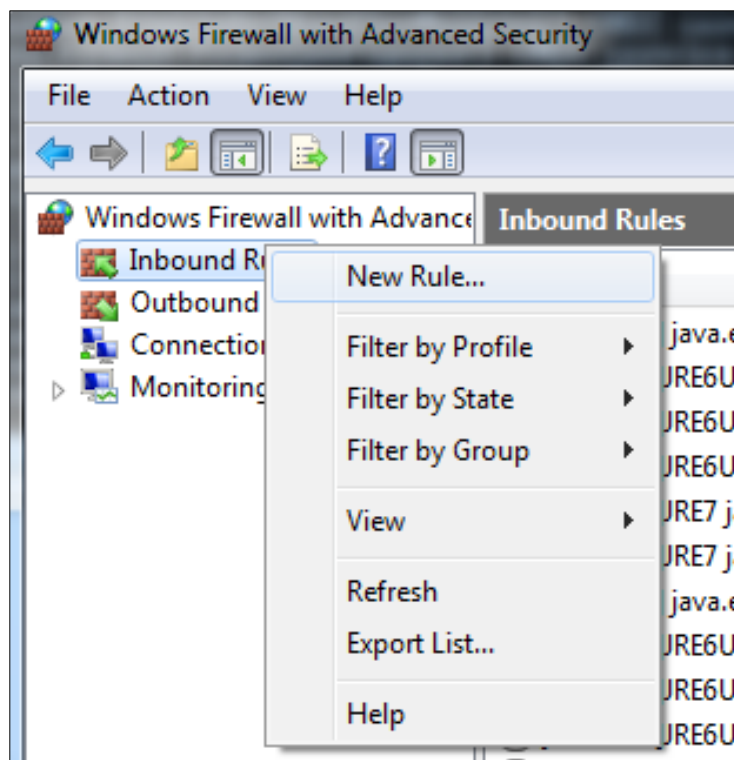
4.   Select *Port* and click on *Next*:
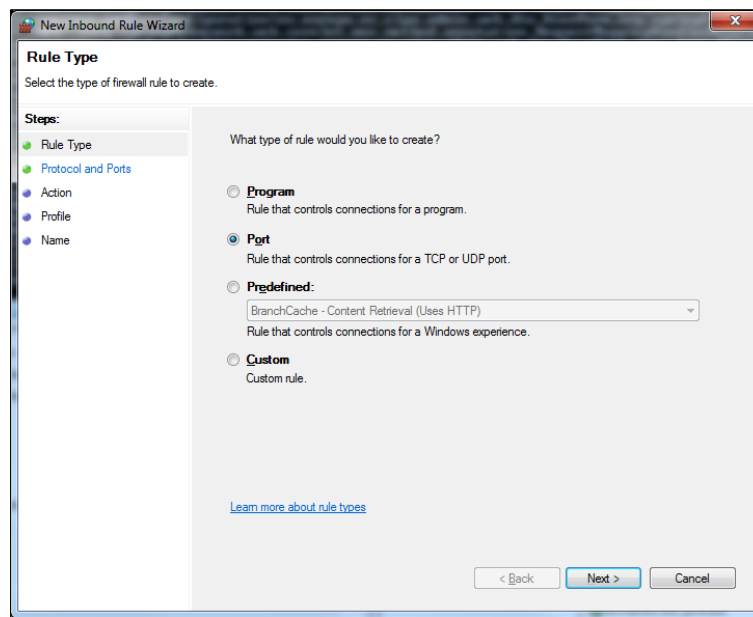
5.   Enter a specific local port (e.g. 8080) and click on *Next*:
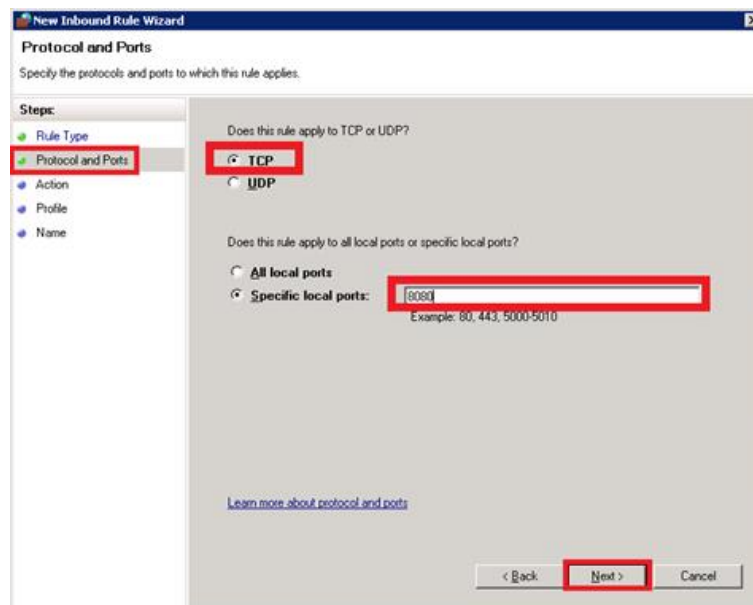
6. Click on *Next*:
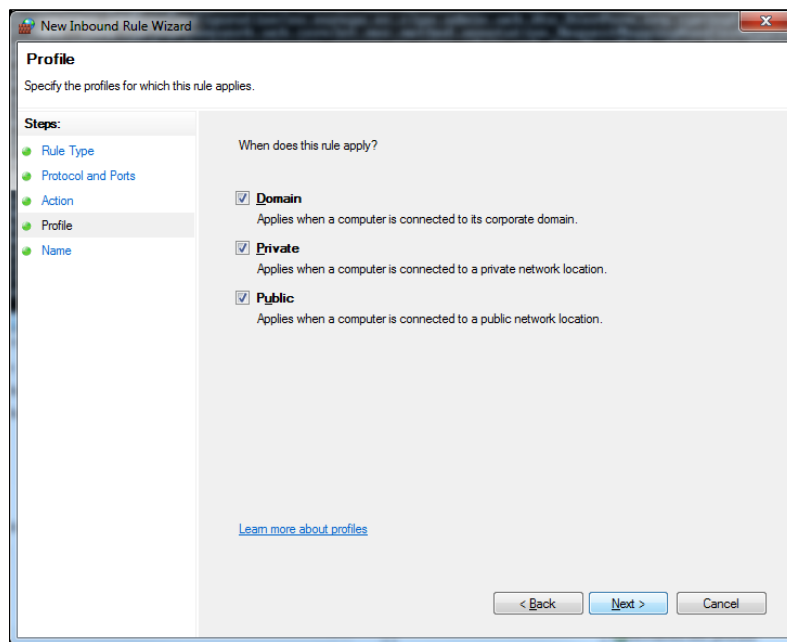
7. Name the rule and click on *Finish*:
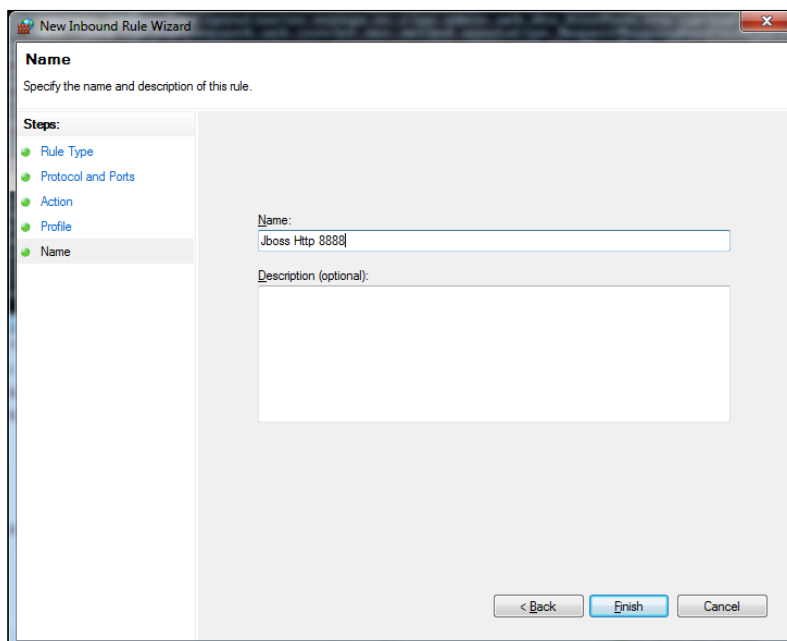
ANNEX 3 PROCESSING MODE

Processing modes (PModes) describe how messages are exchanged between AS4 partners (*Instance A* and *Instance B*). These files contain the identifiers of each AS4 gateway (identified as *parties* in the PMode file below).

InstanceAId1, instanceAId2, instanceBId1, instanceBId2 represent the clients' backend identifiers connected to their associated AS4 gateway (which are *Instance A* and *Instance B* respectively). Therefore, adding, modifying or deleting a participant implies modifying the corresponding PMode files.

In a production environment, you will have an XML file for each instance generated by a plugin (external plugin for Eclipse). This XML file is updated every time a participant is added or modified. For testing purposes, you simply edit the PMode file dedicated to *Instance B*.

Here is an example of the content of a PMode XML file:

Remarks:

- o *In this setup we have allowed each party (InstanceA or InstanceB) to initiate the process. If only InstanceA is supposed to send messages, we need to put only InstanceA in <initiatorParties> and InstanceB in <responderParties>.*

- o *SSL mutual authentication is only required if we use HTTPS for the endpoint. In this case the <CEF-eDelivery path>/modules/eu/domibus/main/clientauthentication.xml file is mandatory.*

- o *The parameter maxSize represents the maximum size allowed for a message and its value can be edited according to the user's need.*

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

<db:configuration xmlns:db="http://domibus.eu/configuration"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://domibus.eu/configuration
file:/C:/development/git-repos/domibus/Domibus-MSH/domibus-
configuration.xsd" party="instanceA">

    <mpcs>

        <mpc name="defaultMpc"

                qualifiedName="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultMPC"

                enabled="true"

                default="true"

                retention_downloaded="0"

                retention_undownloaded="60"/>

    </mpcs>

    <businessProcesses>

        <roles>
```

```xml
                    <role name="default"
                                value="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/defaultRole"/>
                    <role name="exampleMessageProducer"
                                value="exampleMessageProducer"/>
                    <role name="exampleMessageReceiver"
                                value="exampleMessageReceiver"/>
            </roles>
            <parties>
                    <partyIdTypes>
                            <partyIdType name="exampleType"
value="http://www.domibus.eu/exampleType"/>
                    </partyIdTypes>
                    <party name="instanceA"
                                endpoint="http://UndefinedHostnameA:8080/domi
bus/services/msh"
                                allowChunking="false"
>
                            <identifier partyId="instanceAId1"
partyIdType="exampleType"/>
                            <identifier partyId="instanceAId2"
partyIdType="exampleType"/>
                    </party>
                    <party name="instanceB"
                                endpoint="http://UndefinedHostnameB:8080/domi
bus/services/msh"
                                allowChunking="false"
>
                            <identifier partyId="instanceBId1"
partyIdType="exampleType"/>
                            <identifier partyId="instanceBId2"
partyIdType="exampleType"/>
                    </party>
            </parties>
            <meps>
                    <mep name="oneway" value="http://docs.oasis-
open.org/ebxml-msg/ebms/v3.0/ns/core/200704/oneWay"/>
                    <mep name="twoway" value="http://docs.oasis-
open.org/ebxml-msg/ebms/v3.0/ns/core/200704/twoWay"/>
                    <binding name="push" value="http://docs.oasis-
open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push"/>
                    <binding name="pushAndPush" value="http://docs.oasis-
open.org/ebxml-msg/ebms/v3.0/ns/core/200704/push-and-push"/>
            </meps>
```

```xml
<properties>
        <property name="originalSenderProperty"
                  key="originalSender"
                  datatype="string"
                  required="true"/>
        <property name="finalRecipientProperty"
                  key="finalRecipient"
                  datatype="string"
                  required="true"/>
        <propertySet name="listPropertySet">
            <propertyRef property="finalRecipientProperty"/>
            <propertyRef property="originalSenderProperty"/>
        </propertySet>
</properties>
<payloadProfiles>
        <payload name="businessContentPayload"
                  cid="payload"
                  required="true"
                  mimeType="text/xml"
                  inBody="false"/>
        <payload name="businessContentAttachment"
                  cid="attachment"
                  required="false"
                  mimeType="application/octet-stream"
                  inBody="false"/>
        <payloadProfile name="messageProfile"
                  maxSize="40894464">
            <attachment name="businessContentPayload"/>
            <attachment name="businessContentAttachment"/>
        </payloadProfile>
</payloadProfiles>
<securities>
        <security name="signAndEncrypt"
                  policy="signEncrypt.xml"
                  signatureMethod="RSA_SHA256" />
</securities>
<errorHandlings>
        <errorHandling name="errorHandling"
                  errorAsResponse="true"
                  businessErrorNotifyProducer="true"
```

```xml
                                    businessErrorNotifyConsumer="true"
                                    deliveryFailureNotifyProducer="true"/>

            </errorHandlings>

            <agreements>

                    <agreement name="exampleAgreement"
value="http://domibus.eu/agreement" type=""/>

            </agreements>

            <services>

                    <service name="as4Service" value="AS4"
type="exampleService"/>

            </services>

            <actions>

                    <action name="sendBasicMessage" value="submit"/>

            </actions>

            <as4>

                    <receptionAwareness
name="exampleReceptionAwarenessRetryThreeDuplicateDetectionTrue"
retry="1;6;CONSTANT" duplicateDetection="true"/>

                    <reliability
name="exampleReliabilityNonrepudiationTrueReplypatternResponse"
nonRepudiation="false" replyPattern="response"/>

            </as4>

            <legConfigurations>

                    <legConfiguration name="examplePushLegsendBasicMessage"

                                    service="as4Service"

                                    action="sendBasicMessage"

                                    defaultMpc="defaultMpc"

                                    reliability="exampleReliabilityNonrepudiation
TrueReplypatternResponse"

                                    security="signAndEncrypt"

                                    receptionAwareness="exampleReceptionAwareness
RetryThreeDuplicateDetectionTrue"

                                    payloadProfile="messageProfile"

                                    errorHandling="errorHandling"

                                    compressPayloads="true">

                                    </legConfiguration>

            </legConfigurations>

<process name="as4exampleProcess"

            agreement="exampleAgreement"

            mep="oneway"

            binding="push"

            initiatorRole="exampleMessageProducer"

            responderRole="exampleMessageReceiver">
```

```xml
        <initiatorParties>
                <initiatorParty name="instanceA"/>
                <initiatorParty name="instanceB"/>
        </initiatorParties>
        <responderParties>
                <responderParty name="instanceA"/>
                <responderParty name="instanceB"/>
        </responderParties>
        <legs>
                <leg name="examplePushLegsendBasicMessage"/>
        </legs>
</process>
        </businessProcesses>
</db:configuration>
```

ANNEX 4 DOMIBUS PCONF TO EBMS3 PMODE MAPPING

The following table provides additional information concerning the Domibus configuration files.

Domibus pconf	EbMS3 Specification [ebMS3CORE] [AS4-Profile]	Description
MPCs	-	Container which defines the different MPCs (Message Partition Channels).
MPC	„PMode[1].BusinessInfo.MPC: The value of this parameter is the identifier of the MPC (Message Partition Channel) to which the message is assigned. It maps to the attribute **Messaging / UserMessage**	Message Partition Channel allows the partition of the flow of messages from a *Sending MSH* to a *Receiving MSH* into several flows, each of which is controlled separately. An MPC also allows merging flows from several *Sending MSHs* into a unique flow that will be treated as such by a *Receiving MSH*. The value of this parameter is the identifier of the MPC to which the message is assigned.
MessageRetentionDownloaded	-	Retention interval for messages already delivered to the backend.
MessageRetentionUnDownloaded	-	Retention interval for messages not yet delivered to the backend.
Parties	-	Container which defines the different PartyIdTypes, Party and Endpoint.
PartyIdTypes	maps to the attribute **Messaging/UserMessage/ PartyInfo**	Message Unit bundling happens when the Messaging element contains multiple child elements or Units (either User Message Units or Signal Message Units).
Party ID	maps to the element **Messaging/UserMessage/ PartyInfo**	The ebCore Party ID type can simply be used as an identifier format and therefore as a convention for values to be used in configuration and – as such – does not require any specific solution building block.

Endpoint	maps to **PMode[1].Protocol.Address**	The endpoint is a party attribute that contains the link to the MSH.
		The value of this parameter represents the address (endpoint URL) of the *Receiver MSH* (or *Receiver Party*) to which Messages under this PMode leg are to be sent. Note that a URL generally determines the transport protocol (e.g. if the endpoint is an email address, then the transport protocol must be SMTP; if the address scheme is "http", then the transport protocol must be HTTP).
AS4	-	Container
Reliability [@Nonrepudiation] [@ReplyPattern]	Nonrepudiation maps to **PMode[1].Security.SendReceipt.NonRepudiation**	PMode[1].Security.SendReceipt.NonRepudiation : value = 'true' (to be used for non-repudiation of receipt), value = 'false' (to be used simply for reception awareness).
	ReplyPattern maps to **PMode[1].Security.SendReceipt.ReplyPattern**	PMode[1].Security.SendReceipt.ReplyPattern: value = 'Response' (sending receipts on the HTTP response or back-channel).
		PMode[1].Security.SendReceipt.ReplyPattern: value = 'Callback' (sending receipts use a separate connection.)
ReceptionAwareness [@retryTimeout] [@retryCount] [@strategy] [@duplicateDetection]	retryTimeout maps to **PMode[1].ReceptionAwareness.Retry=true**	These parameters are stored in a composite string.
	PMode[1].ReceptionAwareness.Retry.Parameters retryCount maps to	• *retryTimeout* defines timeout in seconds.
	PMode[1].ReceptionAwareness.Retry.Parameters	• *retryCount* is the total number of retries.
	strategy maps to **PMode[1].ReceptionAwareness.Retry.Parameters**	• *strategy* defines the frequency of retries. The only *strategy* available as of now is *CONSTANT*.
	duplicateDetection maps to **PMode[1].ReceptionAwareness.DuplicateDetection**	• *duplicateDetection* allows to check duplicates when receiving twice the same message. The only *duplicateDetection* available as of now is *TRUE*.
Securities	-	Container
Security	-	Container

Policy	PMode[1].Security.* NOT including PMode[1].Security.X509.Signature.Algorithm	The parameter in the pconf file defines the name of a WS-SecurityPolicy file.
SignatureMethod	PMode[1].Security.X509.Signature.Algorithm	This parameter is not supported by WS-SecurityPolicy and therefore it is defined separately.
BusinessProcessConfiguration	-	Container
Agreements	maps to eb:Messaging/ UserMessage/ CollaborationInfo/ AgreementRef	This OPTIONAL element occurs zero times or once. The *AgreementRef* element is a string that identifies the entity or artifact governing the exchange of messages between the parties.
Actions	-	Container
Action	maps to **Messaging/ UserMessage/ CollaborationInfo/Action**	This REQUIRED element occurs once. The element is a string identifying an operation or an activity within a Service that may support several of these
Services	-	Container
ServiceTypes Type	maps to **Messaging/ UserMessage/ CollaborationInfo/ Service[@type]**	This REQUIRED element occurs once. It is a string identifying the service that acts on the message and it is specified by the designer of the service.
MEP [@Legs]	-	An ebMS MEP defines a typical choreography of ebMS User Messages which are all related through the use of the referencing feature (RefToMessageId). Each message of an MEP instance refers to a previous message of the same instance, unless it is the first one to occur. Messages are associated with a label (e.g. *request*, *reply*) that precisely identifies their direction between the parties involved and their role in the choreography.
Bindings	-	Container

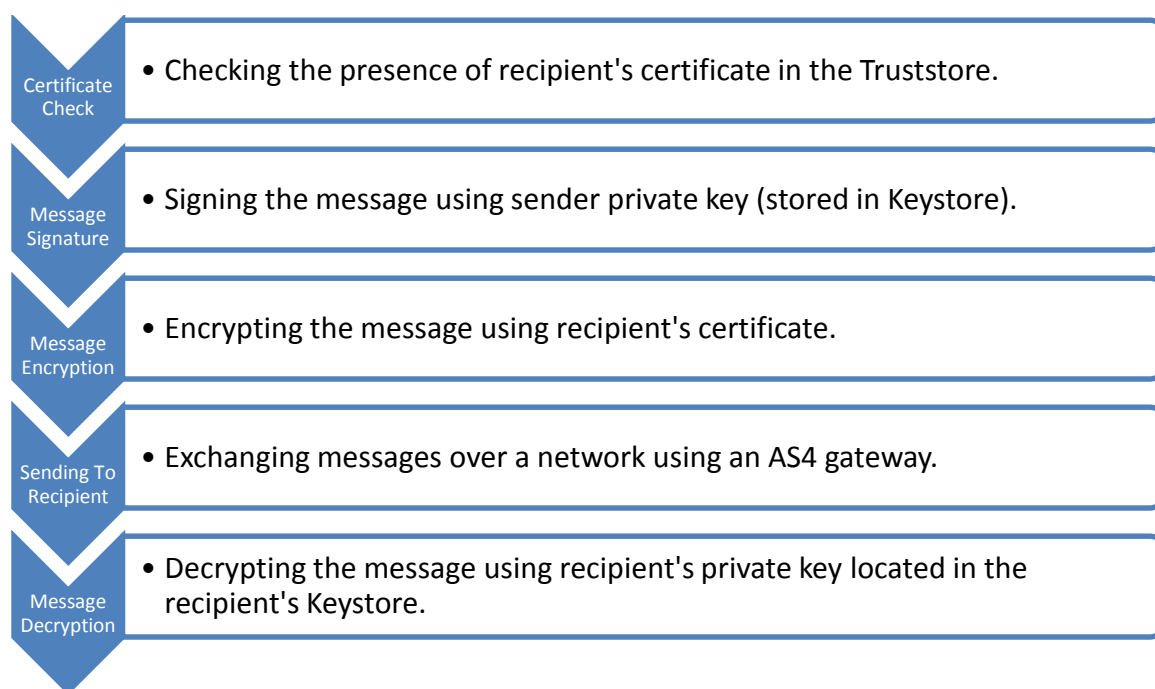|		
Binding	-	The previous definition of ebMS MEP is quite abstract and ignores any binding consideration to the transport protocol. This is intentional, so that application level MEPs can be mapped to ebMS MEPs independently from the transport protocol to be used.
Roles	-	Container
Role	maps to **PMode.Initiator.Role or PMode.Responder.Role** depending on where this is used. In ebMS3 message this defines the content of the following element: • For Initiator: **Messaging/UserMessage/PartyInfo/From/Role** • For Responder: **Messaging/UserMessage/PartyInfo/To/Role**	The required role element occurs once, and identifies the authorized role (*fromAuthorizedRole* or *toAuthorizedRole*) of the Party sending the message (when present as a child of the *From* element), or receiving the message (when present as a child of the *To* element). The value of the role element is a non-empty string, with a default value of *http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/defaultRole* Other possible values are subject to partner agreement.
Processes	-	Container
PayloadProfiles	-	Container
Payloads	-	Container

Payload	maps to **PMode[1].BusinessInfo.PayloadProfile**	This parameter allows specifying some constraint or profile on the payload. It specifies a list of payload parts.
		A payload part is a data structure that consists of five properties:
		1. **name** (or Content-ID) that is the **part identifier**, and can be used as an index in the notation PayloadProfile;
		2. **MIME data type** (text/xml, application/pdf, etc.);
		3. **name of the applicable XML Schema file** if the MIME data type is text/xml;
		4. **maximum size in kilobytes**;
		5. **Boolean** string indicating whether the part is **expected** or **optional**, within the User message.
		The message payload(s) must match this profile.
ErrorHandlings	-	Container
ErrorHandling	-	Container
ErrorAsResponse	maps to **PMode[1].ErrorHandling.Report.AsResponse**	This Boolean parameter indicates (if *true*) that errors generated from receiving a message in error are sent over the back-channel of the underlying protocol associated with the message in error. If *false*, such errors are not sent over the back-channel.
ProcessErrorNotifyProducer	maps to **PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer**	This Boolean parameter indicates whether (if *true*) the Producer (application/party) of a User Message matching this PMode should be notified when an error occurs in the Sending MSH, during processing of the *User Message to be sent*.

ProcessErrorNotifyConsumer	maps to **PMode[1].ErrorHandling.Report.ProcessErrorNotifyProducer**	This Boolean parameter indicates whether (if *true*) the Consumer (application/party) of a User Message matching this PMode should be notified when an error occurs in the Receiving MSH, during processing of the *received User message*.
DeliveryFailureNotifyProducer	maps to **PMode[1].ErrorHandling.Report.DeliveryFailuresNotifyProducer**	When sending a message with this reliability requirement (*Submit* invocation), one of the two following outcomes shall occur: - The Receiving MSH successfully delivers (*Deliver* invocation) the message to the Consumer. - The Sending MSH notifies (*Notify* invocation) the Producer of a delivery failure.
Legs	-	Container
Leg	-	Because messages in the same MEP may be subject to different requirements - e.g. the reliability, security and error reporting of a response may not be the same as for a request – the PMode will be divided into *legs*. Each user message label in an ebMS MEP is associated with a PMode leg. Each PMode leg has a full set of parameters for the six categories above (except for *General Parameters*), even though in many cases parameters will have the same value across the MEP legs. Signal messages that implement transport channel bindings (such as PullRequest) are also controlled by the same categories of parameters, except for *BusinessInfo group*.
Process	-	In *Process* everything is plugged together.

Domibus pconf to ebMS3 mapping

ANNEX 5 INTRODUCTION TO AS4 SECURITY

To secure the exchanges between instances A and B (*Instance A* is sending a message to *Instance B* in this example), it is necessary to set up each instance's *keystore* and *trustore* accordingly. The diagram below shows a brief explanation of the main steps of this process:

Certificate Check
- Checking the presence of recipient's certificate in the Truststore.

Message Signature
- Signing the message using sender private key (stored in Keystore).

Message Encryption
- Encrypting the message using recipient's certificate.

Sending To Recipient
- Exchanging messages over a network using an AS4 gateway.

Message Decryption
- Decrypting the message using recipient's private key located in the recipient's Keystore.

In order to exchange B2B messages and documents between *instances* A and B, it is necessary to check the following:

For *Instance A*	For *Instance B*
Check that *Instance B* certificate (public key of B) is in trustore.jks of A, if not add it.	Check that *Instance* A certificate (public key of A) is in trustore.jks of B, if not add it.
Check that the name of B private key is in the keystore.jks, if not add it.	Check that the name of A private key is in the keystore.jks, if not add it.
In *domibus-configuration.xml*: the keystore alias should be *instanceA*, you may edit the keystore password (by default *test*), and the path to keystore.jks (if you change it).	In *domibus-configuration.xml* edit: the alias property to *InstanceB*, the keystore password (by default *test*) if you need to, and the path to keystore.jks (if you change it).

In a production environment, each participant would need a certificate delivered by a certification authority and remote exchanges between business partners would be managed by each partner's PMode (that should be uploaded on each instance).

For testing purposes, this package provides a pre-configured PMode XML file, a *trustore.jks* file and a *keystore.jks* file to be added to *instances* A and B as described in <u>Annex 3</u> .

Remark:

> *It is necessary to open the required ports when Instance A or Instance B are behind a local firewall. e.g. port 8080 is not opened by default in Windows; we would need to create a dedicated rule on Windows firewall to open TCP 8080 port. See annex "<u>Firewall Settings</u>".*