CIPA Access point installation and user Guide

Contents

1. Overview

The CIPA Access Point allows your company to send to and receive documents from the e-Delivery network using the AS2 and/or AS4 (Ebms3 AS4 profile) protocols for AP to AP transmissions, with all the benefits of the e-Delivery dynamic discovery infrastructure .

2. Description

The Cipa AP is composed of an AS2 and AS4 endpoint implementation, and the CIPA dispatcher component that provides a common interface for sending messaging regardless of the protocol that will be used to send the message. The dispatcher adds dynamic partner discovery feature, based on Busdox SMP/SML components to open source AS2 and AS4 implementations. The dispatcher also decides on which protocol to use for the transmission based on the configuration from the SMP and thus provides mutli-protocol support.

The CIPA team decided to use the Mendelson AS2 open source solution as its AS2 server and the Domibus open source AS4 implementation for the AS4 part. The current version of the dispatcher integrates by default with these two "default" implementations however the dispatcher has been designed in a neutral way allowing it to integrate with other AS2 or AS4 endpoint implementations.

To make installation easier and because it'll be the common case, this guide assumes the 3 components are deployed on the same web server.

3 Installation

3.1 Getting the software

The Cipa access point software that is the object of this guide can be downloaded on the Cipa e-Delivery Joinup site.

3.2 Make sure your system meets the installation requirements

Services

The GW (Gateway) exposes a number of services, most of which may only be exposed to internal users. To ensure that only the required services are reachable from the outside, the use of a proxy server (i.e. Apache2) is highly recommended.

Software Requirements

A Java runtime environment Version 6 or 7 is required for the deployment of the software. The use of version 7 is highly recommended as version 6 is not supported anymore. For downloads and support please see http://java.com.

As the GW is a Java Servlet application, a servlet container is required. The recommended and supported servlet container is Apache Tomcat (http://tomcat.apache.org/) versions 6 and 7. A GW version bundled with a Tomcat 7 container can be downloaded from the projects repository (see **Error! Reference source not found.**).

Please make sure that the JDK you are using to run the access point uses the Java Cryptography Extension (JCE) Unlimited Strength.

Database Requirements (Domibus only)

Supported databases for the GW are MySQL5 and Oracle 10 + 11. While initial database scripts can be provided for all databases supported by Hibernate 3.4.0 on request, update scripts (in case of changes to the database model) will only be provided for the supported database engines.

You will find the scripts to create the initial database included in this installation package. This guide does not provide detailed instruction on how to set up your database as this is highly dependent on your database software.

The database installation is required only for the AS4 access point.

3.3 General Configuration

Configure your proxy server

The access point exposes a number of different services listed in the table below. Please configure your proxy server in a way that only authorized clients can reach the services intended for them (i.e. if you are using an Apache2 webserver as proxy, use ProxyPass and allow directives for the configuration, see http://httpd.apache.org/docs/2.2/mod/mod_proxy.html). The standard server distributed in the cipa installation package listens on http://localhost:8080 and the context path for the application is /domibus. IF YOU DO NOT SECURE THE ACCESS TO THE ADMINISTRATIVE SERVICES THE CONFIDENTIALITY AND INTEGRITY OF YOUR DATA IS AT RISK!

Service localhost:8080	Allow ONLY connection from	description
/domibus	Server administrators	Axis2 status page
/domibus/services/listServices	Server administrators	Overview of deployed axis2 webservices
/domibus/axis2-admin/	Server administrators	Axis2 administration console, user:admin password:axis
/domibus /services/BackendService	Applications retrieving messages from the as4 gateway	Webservice for communicating with the national systems
/domibus/services/msh	The other AS4 gateways	Webservice for communication with other e-AS4 gateways
/as2/HttpReceiver	Other As2 gateways	The http servlet that receives as2 messages
/cipa-dispatcher/send	Applications sending messages though the access point	Unified rest interface used to send simple xml messages to the gateway. Uses Busdox based dynamic discovery to look up receiver access point and protocol to use for the message transfer.
/cipa-dispatcher/AS2Receiver	Internet public	This is the cipa access point as2 receiver endpoint enabling dynamic discovery and PKI based security. This is the endpoint to publish in the SMP for AS2
/cipa-dispatcher/AS4Receiver	Internet public	This is the cipa access point as4 receiver endpoint enabling

		dynamic discovery and PKI based security. This is the endpoint to publish in the SMP for AS4

SSL configuration

The use of SSL for the communication with other GWs is mandatory. The standard configuration assumes that the SSL connection terminates at the proxy server. If you are unsure of the required configurations, please check the documentation of your proxy server (in case of Apache2 http://httpd.apache.org/docs/2.2/ssl).

Configure your database

If you are performing an initial deployment of the GW software, you need to create a database user and a matching database. Instructions how to do this can be found in the manual of your database software. After that execute the database scripts appropriate for your database engine. Those scripts can be found in the sql-scripts folder of your software package, named domibus_${database_engine}_initial.sql.

Deploy the database driver

Because of licensing reasons the distributed Tomcat server does not contain any database drivers besides an Apache Derby driver for testing purposes. A JDBC compatible database driver has to be deployed to the server (tomcat-7/lib).

3.3 Domibus AS4 Configuration

The Domibus.xml

The file conf/Catalina/localhost/domibus.xml contains most of the configuration data required to run the GW. During runtime those parameters are bound to the JNDI of the servlet container under the path java:comp/env/. If you use a servlet container different than Tomcat make sure those values are provided by the server. Details on how to accomplish this can be found in the documentation of your servlet container.

```
<Context path="/domibus" privileged="false" reloadable="false">

    <!-- DATABASE CONFIGURATION START -->
    <!-- Database configuration, driverClassName, url, username and password
        must be changed -->
    <Resource name="domibusPU"
              auth="Container"
              factory="org.apache.naming.factory.BeanFactory"
              type="com.mchange.v2.c3p0.ComboPooledDataSource"
              driverClass="com.mysql.jdbc.Driver"
              maxPoolSize="50"
              minPoolSize="15"
```

```
                acquireIncrement="3"
                acquireRetryAttempts="0"
                acquireRetryDelay="3000"
                breakAfterAcquireFailure="false"
                maxConnectionAge="60"
                maxIdleTime="30"
                maxIdleTimeExcessConnections="10"
                idleConnectionTestPeriod="15"
                testConnectionOnCheckout="false"
                preferredTestQuery="SELECT 1"
                debugUnreturnedConnectionStackTraces="true"
                autoCommitOnClose="true"
                jdbcUrl="jdbc:mysql://localhost:3306/domibus?user=domibus"
                user="domibus"
                password="domibus"/>

    <!-- Name of the database connection as defined above -->
    <Environment name="domibus.persistence.unit" value="domibusPU"
                type="java.lang.String"/>

    <!-- File where addidtional persistence properties are defined -->
    <Environment name="domibus.persistence.properties"
value="${catalina.home}/conf/domibus/hibernate.properties"
                type="java.lang.String"/>
    <!-- DATABASE CONFIGURATION END -->


    <!-- SECURITY CONFIGURATION START -->
    <!-- File where the security module is configured -->
    <Environment name="domibus.module.security.configFile"
                value="${catalina.home}/conf/domibus/security-config.xml"
type="java.lang.String"/>

    <!-- Folder where security policies are defined -->
    <Environment name="domibus.module.security.policiesFolder"
                value="${catalina.home}/conf/domibus/policies"
type="java.lang.String"/>

    <!-- Timeout in milliseconds for the expiration of the security configuration
        cache -->
    <Environment name="domibus.module.security.config.timeout"
                value="30000" type="java.lang.String"/>

    <!-- Password for the private key used by the gateway -->
    <Environment name="domibus.module.security.config.privatekey.password"
                value="apache" type="java.lang.String"/>
    <!-- SECURITY CONFIGURATION END -->


    <!-- FILE SYSTEM CONFIGURATION START -->
    <!-- Folder where the backend module stores messages -->
    <Environment name="domibus.module.backend.messagesFolder"
```

```
                value="${catalina.home}/store/backend_store"
type="java.lang.String"/>

    <!-- Folder where the ebms3 module stores attachments -->
    <Environment name="domibus.module.ebms3.attachmentFolder"
                value="${catalina.home}/store/attachments"
type="java.lang.String"/>

    <!-- Folder where the PMode configuration files are located -->
    <Environment name="domibus.module.ebms3.PModesDir"
value="${catalina.home}/conf/domibus/pmodes"
                type="java.lang.String"/>

    <!-- File where gateway consumption rules are defined -->
    <Environment name="domibus.module.ebms3.gatewayConfigFile"
                value="${catalina.home}/conf/domibus/gateway.xml"
type="java.lang.String"/>

    <!-- Folder where the ebms3 module stores submitted messages -->
    <Environment name="domibus.module.ebms3.submittedMessagesFolder"
                value="${catalina.home}/store/send" type="java.lang.String"/>

    <!-- Folder where the ebms3 module stores received messages -->
    <Environment name="domibus.module.ebms3.receivedMessagesFolder"
                value="${catalina.home}/store/receive" type="java.lang.String"/>

    <!-- File where worker execution rules are defined -->
    <Environment name="domibus.module.ebms3.workersFile"
                value="${catalina.home}/conf/domibus/workers.xml"
type="java.lang.String"/>

    <!--  support for domibus 1.3 messages -->
    <Environment name="domibus.module.ebms3.enforce.1_3.compatibility"
                value="true" type="java.lang.Boolean"/>
    <!-- FILE SYSTEM CONFIGURATION END -->


    <!-- MISC CONFIGURATION START -->
    <!-- Number of days after which not downloaded messages are deleted -->
    <Environment name="domibus.module.backend.messagesTimeLive"
                value="60" type="java.lang.String"/>

    <!-- Cron expression determining how often the check for the time limit
        defined above is executed -->
    <Environment name="domibus.module.backend.deleteMessagesCron"
                value="0 1/2 * * * ?" type="java.lang.String"/>

    <!-- Hostnames of the server the gateway is running on -->
    <Environment name="domibus.module.ebms3.hostnames" value="localhost,127.0.0.1"
                type="java.lang.String"/>
    <!-- MISC CONFIGURATION END -->
```

```
</Context>
```

If you are using the provided tomcat instance, editing the “DATABASE CONFIGURATION” and “SECURITY CONFIGURATION” sections of this file is sufficient. Otherwise you will also have to edit the “FILE SYSTEM” section and create the respective folders and copy the defined configuration files to the specified locations. Relative paths in this configuration file are always relative to the folder your server is started from.

DATABASE CONFIGURATION section of the provided tomcat gives you an example on how to configure a datasource to connect your access point to a MySql database. You should adapt this according to the database you are going to use with your access point. Although it is possible to use derby as database this should be limited for internal testing.

the hibernate.properties

The hibernate.properties file is located in the conf/domibus folder and defines additional database properties. The only property that has to be changed is “hibernate.dialect”. Valid configuration properties are listed at http://docs.jboss.org/hibernate/orm/3.5/javadocs/org/hibernate/dialect/package-summary.html. If you are using an Oracle 11g database please use the org.hibernate.dialect.Oracle10gDialect.

```
hibernate.dialect= org.hibernate.dialect.MySQLDialect

hibernate.show_sql=false

hibernate.format_sql=true

hibernate.hbm2ddl.auto=none
```

Please contact your local database administrator for the configuration of the other properties. The default values should be fine, but your local policy may differ (i.e. on the number of allowed pooled connections).

PModes and public certificates

Pmodes is the AS4/EBMS3 way of defining partners with whom you can exchange messages. In the Domibus AS4 implementation, Pmodes are defined in xml files. All the pmode file must be placed in the conf\domibus\pmodes folder of the provided tomcat instance.

With the Cipa access point, Pmodes are created at runtime using data coming from the SMP. All Podes created that way by the access point can be found in the following file: cipa.dynamic.pmode.xml.

All three components must share the same truststore. That store is the certificates.p12 file the can be found under the /bin folder of the provided tomcat instance. The Password of the truststore is test.

The certificates.p12 must contain the root certificates you trust if you are using dynamic discovery and dynamic partner creation based on PKI. By default the Peppol root certificates are installed in the truststore.

Private key

Create a keystore containing only your private key for signing messages. This key should differ from the key you are using for SSL encryption. The keystore can be created using the standard Java "keytool" command. Documentation for this tool is available at http://docs.oracle.com/javase/1.4.2/docs/tooldocs/windows/keytool.html. Copy this keystore to the location configured in the security-config.xml file (default: conf/domibus/keys/keystore.jks).

security-config.xml

This file contains the keystore configuration and security agreements between participating countries. Please do edit contents inside the <tns:keystores> element only. This file can be found in the conf\domibus folder of the tomcat server.

```
      <tns:privateKeystore>
         <tns:localAlias>APP_1000000002</tns:localAlias>
         <tns:storepwd>Europa2011</tns:storepwd>
         <tns:keypwd></tns:keypwd>
         <tns:file>accesspoint.jks</tns:file>
         <tns:storeType>jks</tns:storeType>
      </tns:privateKeystore>

      <tns:publicKeystore>
         <tns:storepwd>test</tns:storepwd>
         <tns:file>certificates.p12</tns:file>
         <tns:storeType>pkcs12</tns:storeType>
      </tns:publicKeystore>
   </tns:keystores>
```

key	description
tns:localAlias	The alias of your private key
tns:storepwd	The password of your keystore file
tns:keypwd	The password of your private key
tns:file	The location and name of your keystore file
tns:storeType	The type of the keystore file. Use jks keystores only

3.4 Mendelson AS2 Configuration

The only thing needed to configure mendelson is to add your accesspoint keypair in the ***certificates.p12*** file. Due to technical limitations, Mendelson cannot use two different files as Truststore and Keystore and must use the certificates.p12 file.

The certificates.p12 must contain the root certificates you trust if you are using dynamic discovery and dynamic partner creation based on PKI. By default the Peppol root certificates are installed in the truststore.

conf\keystore is the SSL truststore Mendelson uses when sending to partners through SSL. It's already configured with several common root CAs, if you ever face SSL certificate problems when sending to a partner, you'll have to add your partner's root CA certificate here. This store's password is 'test'.

3.5 Dispatcher AS4 Configuration

The configuration of the dispatcher is the last step before you can start using the access point. The configuration of the dispatcher is centralised in the dispatcher.properties that you can find on the CIPAConf folder of the provided tomcat instance. The file contains the following properties :

Property Name	Description
as2_endpoint_url	URL of your AS2 endpoint servlet receiving messages. In our case, Mendelson's http receiver
as4_endpoint_url	URL of your AS4 endpoint servlet receiving messages. In our case, the domibus msh webservice endpoint
as4_pmodeFilePath	The pas to the Pmode file that is dynamically updated by the dispatcher
as4_pmodereload_url	Domibus servlet url to reload the pmodes before sendin the message
as4_default_security_name	The security policy that is used for the dynamic pmode creation. By default messages are signed.
ebms_wsdl_path	The url to the domibus backend service wsdl.
server_mode	debug or production, among other things the main difference is on debug mode you can use self-signed SSL certificates
ssl_truststore , ssl_truststore_password	only used when your AS2 endpoint uses SSL instead fo plain HTTP. Because most of the times the dispatcher and the AS2 endpoint will be deployed on the same server, you won't need SSL and these two properties will be ignored.
db_driver_name , etc..	properties configuring Mendelson's HSQLDB where it stores metadata about the partners and message state. You don't need to change these values.
partner_interface_implementation_class , send_interface_implementation_class	the classes implementing the AS2 interfaces. If you decide to use an AS2 endpoint other than Mendelson, then you'll need to implement IAS2EndpointDBInterface and

	IAS2EndpointSendInterface from the cipa-dispatcher code, and update these values.
keystore_path, keystore_password	details about the keystore your AS2 endpoint will use to store your partners' certificates.Must be using the certificates.p12 provided truststore
keystore_ap_ca_alias	alias of the AP CA certificate inside aforementioned keystore. This certificate is used to perform the pki base check. By default the peppol access poin PKI.
keystore_ap_alias	The alias of your access point certificate. You should use the Common Name of the certificate as Alias.
ocsp_responder_url	URL where the dispatcher will check for revoked certificates, in case *ocsp_validation_activated* is set to true
smp_mode	the mode in which the dispatcher will determine the receiver's metadata: ▪ NO_SMP: the dispatcher will try to retrieve the receiver's metadata from the internal AS2 endpoint's database. ▪ DIRECT_SMP: the dispatcher will call the url on *smp_url* to retrieve participants' metadata. ▪ PRODUCTION: the dispatcher will use the OpenPeppol infrastructure (DNS discovery + SMP call) to retrieve the participants' metadata.
start_endpoint , as2_endpoint, ebms_endpoint	the higher the values, the more preference that protocol will have in case you are sending a message to a participant able to communicate in both protocols. If value is set to 0, that protocol will not be enabled in the dispatcher.
temp_folder_path	the dispatcher automatically creates temp message files every time before sending, this property sets the path for a folder storing them.
cache_max_number_entries	the dispatcher keeps an internal cache with participants' metadata not to have to call the SMP every time. This value sets the maximum number of entries on that cache.
cache_expire_entry_after_hours	number of hours after which the cache will consider an entry expired.

If you are using the provided tomcat instance you just need to update the following properties keystore_ap_ca_alias (if you want to trust antother root), keystore_ap_alias and the smp related properties.

4 How to use it

4.1 SMP installation

After you've finalized your AP configuration, you are ready to make a first test. But to properly test your dispatcher's functionalities you'll first need an accessible SMP with at least one participant registered on it. If you don't have any SMP installed please follow these steps:

1. Create the SMP database. You need to set up a MySql server and execute the following script on it to create the database
2. Update the config.properties file that you will find in the SMP folder of this installation package to the values relevant for your environment and copy it to the CIPAConf folder of the tomcat included in this distribution. Here is a description of the values you must set in there :
    a. egServiceRegistrationHook.id: the ID your SMP will use to identify itself.
    b. regServiceRegistrationHook.regLocatorUrl: the URL of the SML this SMP will communicate with.
    c. regServiceRegistrationHook.keystore.classpath:
    d. regServiceRegistrationHook.keystore.password:
    e. xmldsig.keystore.classpath: path to the keystore where the keypair used to sign responses is located. A relative path can be given for the trustore, in that case the base directory will be ${webser_base_dir}/keystores/ , so if xmldsig.keystore.classpath=keystore1.jks the equivalent absolute path will be ${webser_base_dir}/keystores/keystore1.jks .
    f. xmldsig.keystore.password: password for the keystore
    g. xmldsig.keystore.key.alias: alias of the keypair in the keystore used to sign responses.
    h. xmldsig.keystore.key.password: password for the alias.
    i. jdbc.driver: JDBC driver for the SMP DB.
    j. jdbc.url: URL of the SMP DB.
    k. jdbc.user: user to log in the SMP DB.
    l. jdbc.password: password to log in the SMP DB.
    m. target-database: type of database used.
    n. jdbc.read-connections.max: Maximum number of connections the SMP can keep open on the DB.
3. Copy the cipa-smp-full-webapp.war file that you will find in the SMP folder of this installation in the tomcat-7\webapps
4. Start or restart the server to deploy the smp.(you can check that the smp is up and running using the following url : http://localhost:8080/cipa-smp-full-webapp/web/index.html)

When you have an SMP up and running you can start configuring participants you need to test as2 and AS4. Please follow these steps:

1. Download the SMP test project for SoapUI. Import this project into the SoapUI test tool (you can download it from the SOAP UI website).
2. In SoapUI, double click on the SMP node marked in blue, and under the Service Endpoints tab modify the url to point to your SMP.
3. Register a participant into your SMP. First step is put a ServiceGroup, go to 'Non-Core SMP Services' node and double click the test 'Put ServiceGroup'. Change the participant's data at your convenience, making sure the url parameters (participant scheme and participant id in this case) match the data you insert in the XML. Run the test.
4. Create the service metadata. Double click on 'Put SignedServiceMetadata' test. If you changed the values for the service group, you'll have to make the same changes here. The xml field 'EndpointReference' is the url where messages will be sent for this participant (must point to the specific receiver servlet on your access point /cipa-dispatcher/AS2Receiver for as2 or /cipa-dispatcher/AS4Receiver) for AS4. The attribute transportProfile in the Endpoint tag is the protocol this participant will be registered with. Possible values as specified in the "Peppol Policy for Identifiers" document are busdox-transport-as2-ver1p0 for AS2 and ebms3-as4 for the ASA protocol. Run the test.
5. Trick : You can use one access point for the testing : simply use your ap URL as endpoint reference to send the message to yourself.

When the participant has been correctly registered on your SMP, now you can test the dispatcher.

4.2 Testing the dispatcher

Now we are going to send our first AS2 or AS4 message through the dispatcher:

1. make sure that dispatcher.properties file inside the CIPAConf folder of your tomcat has the values for *smp-mode* set to direct_smp and *smp_url* pointing to the SMP you registered your participant.
2. Your access point will be listening formessages to send on /cipa-dispatcher/send.()rest webservice
3. Download the AS2/AS4 SoapUI project and import it into SoapUI.
4. Double click the AS2 node marked with a blue icon and under Service Endpoints tab change the url if needed.
5. Double click on the 'Send AS2 Message' test. Here you'll have to make modifications accordingly to the participant you created on your SMP, so that Receiver.Identifier , DocumentID.InstanceIdentifier and ProcessID.InstanceIdentifier have the right same values you set when you registered its metadata.
6. Run the test. If everything has been correctly set up, it will call the dispatcher, that will extract the receiver's values from the SBDH header and call your SMP to retrieve the participant's metadata. Once the dispatcher knows the receiver's certificate and url where to send, it updates Mendelson or Domibus with that data and forwards the message to it. The message should arrive to your AS2 or AS4 receiver endpoint few seconds afterwards. The use of AS2 or AS4 depends on what you configured for the participant in the SMP.

When sending to the dispatcher, you can get the following error messages due to bad SBDH documents or misconfiguration:

- *An error occurred while treating the SBDH request*: your SBDH document is malformed. You'll receive information about what's missing in the error response.

- *The necessary field 'X' could not be found in the SBDH header*: one of ReceiverIdentifier, SenderIdentifier, DocumentIdentifier or ProcessIdentifier weren't included on your SBDH document, or they weren't at the right xml location.
- *There's no activated protocol in this Access Point able to communicate with the receiver*: the protocols you want to activate on your dispatcher are specified in conf.properties inside cipa-dispatcher.war. If the receiver of your message doesn't use any of your activated protocols, there's no way your endpoint can send to it.

If you want to test message reception on your new AP, you'll have to register the following urls on an SMP:

- host:port/cipa-dispatcher/AS2Receiver for AS2 protocol
- host:port/cipa-start-server/AS4Receiver for AS4 protocol.

Message reception is automatic and there's no need for any action from your side. If a message signed with a valid AP certificate gets to your AP you'll successfully receive the message, and it'll be stored in your server (by default inside /bin/messages/ for AS2 messages and inside the domibus database for AS4). More information about Mendelson AS2 endpoint can be found at http://community.mendelson-e-c.com/forum/as2

To retrieve a message from the Domibus gateway you can use the As4 retrieve test suite that is part of the SoapUI test project you downloaded earlier. There are two services that you can call on your accesspoint to retrieve messages:

- listPendingMessages that returns a list of messages that are available for download in your gateway.
- downloadMessage service that allows you to download a message. You must update the <messageID></messageID> element to specify the id of a message that is available for download (retrieved with the list pending message). The message will be returned in the bodyload element of the response in a base64 encoded string.

If you want to learn more about how the AS2 protocol works, please refer to http://en.wikipedia.org/wiki/AS2 and http://www.ietf.org/rfc/rfc4130.txt .