PEPPOL Java Registry Locator Service Developer Guide

Version 1.1DRAFT

2011-01-04

Contents

[1 Document information 2](#_Toc285047494)

[1.1 Document history 2](#_Toc285047495)

[1.2 Editors 2](#_Toc285047496)

[2 Introduction 3](#_Toc285047497)

[3 Prerequisites 3](#_Toc285047498)

[4 Getting and Compiling the Source Code 3](#_Toc285047499)

[5 Management service 3](#_Toc285047500)

[5.1 Deploying the service 3](#_Toc285047501)

[5.2 Configuring the service 4](#_Toc285047502)

[6 Management console client 4](#_Toc285047503)

[6.1 Building and packing 4](#_Toc285047504)

[6.2 Using the client 4](#_Toc285047505)

[7 Management client library 5](#_Toc285047506)

[7.1 Building and packing 5](#_Toc285047507)

[7.2 Using the library 5](#_Toc285047508)

[7.2.1 ManageParticipantIdentifierServiceCaller 5](#_Toc285047509)

[7.2.2 ManageServiceMetadataServiceCaller 6](#_Toc285047510)

[8 SML at BRZ 6](#_Toc285047511)

[8.1 Building 6](#_Toc285047512)

[8.2 Deploying 7](#_Toc285047513)

# Document information

## Document history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Initials | Changes |
| 2009-04-23 | 0.1.0 | CUP | Initial draft |
| 2009-04-30 | 0.2.0 | CUP | Added review comments |
| 2009-09-28 | 0.3.0 | CUP | Updated to match 0.9 specification |
| 2009-10-08 | 0.4.0 | CUP | Updated to match 0.95 specification |
| 2009-10-19 | 0.5.0 | CUP | Updated the host names |
| 2010-01-21 | 0.6.0 | CUP | Updated to match 1.0 specification |
| 2011-01-04 | 1.1.0 | PH | Updated to the current implementation |

## Editors

CUP: Christian Uldall Pedersen (Accenture)

DS: Dennis Søgaard (Accenture)

PH: Philip Helger (Austrian Federal Computing Center)

# Introduction

This document is a developer guide to the client side components of the PEPPOL Java Service Metadata Locator service. Please ensure that you followed the generic introduction in the “PEPPOL Java 00 Developer Guide” document. Afterwards you may continue with this document.

# Prerequisites

Required components for running the SML

* Latest Tomcat 6.x server with Metro 2.0.1 framework
* MySQL 5.x

# Getting and Compiling the Source Code

The Service Metadata Locator project source code is located at:

<https://svn.forge.osor.eu/svn/peppol/java/sml>

The project has the following components:

* Management service (/sml-webapp)
* Management client library (/client-library)
* Management console client(/client-console)

All of the components are added to the repository as Eclipse projects. The easiest way to build the projects is therefore to import the projects into Eclipse. The console client is dependent on the client library project being present as a project on the classpath.

As a prerequisite the projects from the following paths must be installed into your local Maven repository:

* <https://svn.forge.osor.eu/svn/peppol/java/parent-pom>
* <https://svn.forge.osor.eu/svn/peppol/java/busdox/busdox-types>
* <https://svn.forge.osor.eu/svn/peppol/java/busdox/busdox-peppol>

# Management service

The SML management service has been implemented as a SOAP interface with a database backend. It is possible to change the backend, but the following description is based on a database backend. A copy of the database creation SQL can be found in the /etc/database\_backups/ folder.

## Deploying the service

The service can be deployed in two ways:

* Running the service through Eclipse. This can be done by running the test file “eu.peppol.registry.sml.jetty.RunInJettySMLMgr” as a regular Java application.
* Creating a war file by running “mvn install” on the commandline of the ServiceMetadataLocatorManagement project (assuming all prerequisites have been built)

Metro must be installed on the JRE/Tomcat server for the service to compile and run!

## Configuring the service

A file named “config.properties” must be present in the src/main/resources folder containing the following configuration properties:

dataHandler.class=<Data handler class>

dataChangedListener.class=<Data changed listener class>

dnsClient.enabled=<Whether DNS should be enabled or not>

dnsClient.zone=<DNS zone name>

dnsClient.smlzonename=<SML DNS zone name>

dnsClient.server=<DNS server url>

Example:

dataHandler.class=eu.peppol.registry.sml.hibernate.HibernateDataHandler

dataChangedListener.class=eu.peppol.registry.sml.dns.DNSDataChangedListener

dnsClient.enabled=true

dnsClient.zone=peppolcentral.org.

dnsClient.smlzonename=sml.peppolcentral.org.

dnsClient.server=blixdns1

When using the Hibernate data handler one must include a “hibernate.cfg.xml” file in the src/main/resources folder. Example of the most important parameters of the hibernate.cfg.xml configuration:

<property name=*"connection.url"*>jdbc:mysql://localhost/sml</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>peppol</property>

# Management console client

## Building and packing

The easiest way to build the client is by using Eclipse. The project is dependent on the management client library which must be on the classpath.

Either one can launch eu.peppol.registry.sml.management.client.Main directly from Eclipse or one can run “mvn install” on the console to build a jar file.

## Using the client

As shown in the previous section the client is executed by specifying a series of parameters. The parameters are specified in the following order:

-u user -p password -h host -i id action type params

* user: The username on the web service.
* password: The password on the web service.
* host: The hostname of the web service.
* id: The publisher id of the SMP.
* action: The action wanted. Possibly values: [create,update,delete,read,list, preparetomigrate,migrate].
* type: The type the action should be used on. Possibly values: [identifier,metadata].
* params: Additional parameters specific to the chosen action and type.

An example of the usage (in a single line):

-u User1 -p Test1234 –i SMP-ID1 -h http://infra.peppol.at/ServiceMetadataLocatorManagement/ read metadata SMP-ID1

# Management client library

## Building and packing

The easiest way to build the library is using Eclipse. A jar can be created by running “mvn install” on the console.

## Using the library

A management client can be created using the metadata management and business identifier management library. The library has two main classes called “ManageParticipantIdentifierServiceCaller” and “ManageServiceMetadataServiceCaller”. Error handling has been omitted in the following examples to keep the code more readable.

The methods, constructors and so on are documented using javadoc.

### ManageParticipantIdentifierServiceCaller

This class contains methods for managing the participant identifiers of a given user. The constants used in the following code examples are:

* *MANAGEPARTICIPANTIDENTIFIERENDPOINT:* The URL of the SML endpoint. Example: <http://infra.peppol.at/ServiceMetadataLocatorManagement/manageparticipantidentifier>
* *SMLUSERNAME:* The username used on the SML. Example: User1
* *SMLPASSWORD:* The password used on the SML. Example: Test1234

The class can be constructed in the following way:

ManageParticipantIdentifierServiceCaller biClient = new ManageParticipantIdentifierServiceCaller(MANAGEPARTICIPANTIDENTIFIERENDPOINT, SMLUSERNAME, SMLPASSWORD);

Example showing how to create a new participant identifier:

ManageParticipantIdentifierServiceCaller biClient = new ManageParticipantIdentifierServiceCaller(MANAGEPARTICIPANTIDENTIFIERENDPOINT, SMLUSERNAME, SMLPASSWORD);

ParticipantIdentifierType businessIdentifierCreate = new ParticipantIdentifierType();

businessIdentifierCreate.setValue(“0010:5798000999999”);

businessIdentifierCreate.setScheme(“iso6523-actorid-upis”);

ServiceMetadataPublisherServiceForParticipantType serviceMetadataPublisherServiceForBusiness = new ServiceMetadataPublisherServiceForParticipantType();

serviceMetadataPublisherServiceForBusiness.setParticipantIdentifier(businessIdentifierCreate);

serviceMetadataPublisherServiceForBusiness.setServiceMetadataPublisherID(“SMP-ID1”);

biClient.create(serviceMetadataPublisherServiceForBusiness);

### ManageServiceMetadataServiceCaller

This class contains methods for managing the service metadata of a given user. The constants used in the following code examples are:

* *MANAGESERVICEMETADATAENDPOINT:* The URL of the SML endpoint. Example: <http://infra.peppol.at/ServiceMetadataLocatorManagement/manageservicemetadata>
* *SMLUSERNAME:* The username used on the SML. Example: User1
* *SMLPASSWORD:* The password used on the SML. Example: Test1234

The class can be constructed in the following way:

ManageServiceMetadataServiceCaller client = new ManageServiceMetadataServiceCaller(MANAGESERVICEMETADATAENDPOINT, SMLUSERNAME, SMLPASSWORD);

Example showing how to create the service metadata of a specific user:

ManageServiceMetadataServiceCaller client = new ManageServiceMetadataServiceCaller(MANAGESERVICEMETADATAENDPOINT, SMLUSERNAME, SMLPASSWORD);

ServiceMetadataPublisherServiceType serviceMetadataCreate = new ServiceMetadataPublisherServiceType();

serviceMetadataCreate.setServiceMetadataPublisherID(“SMP-ID1”);

PublisherEndpointType endpoint = new PublisherEndpointType();

endpoint.setLogicalAddress(“http://domain.com/);

endpoint.setPhysicalAddress(“127.0.0.1”);

serviceMetadataCreate.setPublisherEndpoint(endpoint);

client.create(serviceMetadataCreate);

# SML at BRZ

## Building

To build a new version of the SML perform the following steps in the correct order:

1. Delete the folder “java\sml\sml-webapp\target”
2. Open a command-line and call the following shell script:  
   sml-build.cmd install -Dhttp.proxyHost=172.30.9.12 -Dhttp.proxyPort=8080
3. Copy all files from the directory “java\sml\sml-webapp\target\sml-webapp-x.y.z-SNAPSHOT” into the Tomcat context directory (where x.y.z denotes the latest version number)

## Deploying

After deploying the web application onto the Tomcat server, the following things need to be done:

* Adopt the Hibernate database configuration in the file “WEB-INF/classes/hibernate.cfg.xml” (username, password, url)
* Adopt the SML configuration file “WEB-INF/classes/config.properties” and ensure that DNS is enabled. Example for the development SML:

dataHandler.class=eu.peppol.registry.sml.hibernate.HibernateDataHandler

dataChangedListener.class=eu.peppol.registry.sml.dns.DNSDataChangedListener

dnsClient.enabled=true

dnsClient.zone=peppolcentral.org.

dnsClient.smlzonename=smj.peppolcentral.org.

dnsClient.server=blixdns1

After changing these files, you may need to reload the web application in the Tomcat manager.