EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

SML

Software Architecture Document

(BDMSL)

Version [2.4]

Status [Final]

Date: 14/10/2022

Document Approver(s):

Approver Name	Role
Bogdan DUMITRIU	Project Manager

Document Reviewers:

Reviewer Name	Role
Joze RIHTARŠIČ	Developer

Summary of Changes:

Version	Date	Created by	Short Description of Changes
1.0	24/07/2015	Adrien FERIAL	Initial Version
1.1	15/10/2015	Adrien FERIAL	Changes after comments from Benoît DEBROUX, Sandro D'ORAZIO and Olivier DERVEAU
1.2	13/01/2016	Adrien FERIAL	Changes Related to ROLE_ADMIN
2.0	01/06/2016	Yves ADAM	Merge with technical design document
2.1	26/07/2016	Flavio SANTOS	Replacing the word CIPA by BDMSL
2.2	26/08/2016	Yves ADAM	Adjust some formatting errors
2.3	29/08/2016	Flavio SANTOS	WebContext for Jboss Added and DNS algorithm for NAPTR replaced by SHA-256 Base32
2.4	19/09/2016	Adrien FERIAL	Added new error code
2.5	27/09/2016	Flavio SANTOS	Data model diagram update
2.6	24/01/2017	Flavio SANTOS	Granting SMP Role to multiple domains
2.7	06/03/2017	Flavio SANTOS	Updating Migration Key Constraint
2.8	07/03/2017	Tiago MIGUEL	Changes related to Is Alive
2.9	24/04/2017	Flavio SANTOS	Added multiple domains configuration
2.10	16/06/2017	Tiago MIGUEL	Data model update regarding subdomains
2.11	18/07/2017	Flavio SANTOS	Added BlueCoat Authentication activation flag. Added regular expression configuration to validate participant id. Added logical address protocol configuration flag
2.12	09/01/2018	Flavio SANTOS	Defining participant DNS records configuration
2.13	27/03/2018	CEF Support	Reuse policy notice added.
2.14	30/04/2018	Flavio SANTOS	Adding private key and domain configurations
2.15	03/07/2018	Jože RIHTARŠIČ	Changed Admin-Pwd hash algorithm from SHA256 to BCrypt Added new configuration parameter for SMP authorization
2.16	25/09/2018	Caroline AEBY	No more standby service
2.17	13/05/2019	Caroline AEBY	Updates for SML 4.0
2.18	20/01/2020	Jože RIHTARŠIČ	Note added regarding deleting participant under migration.
2.19	30/11/2020	Jože RIHTARŠIČ	Added new configuration properties and

			description for managing truststore certificates.
2.2	07/04/2022	Caroline AEBY	No more CEF
2.3	15/09/2022	Jože RIHTARŠIČ Caroline AEBY	Updates for SML 4.2
2.4	14/10/2022	Caroline AEBY	Replaced CEF by Digital Europe Programme in heading

Table of Contents

1. Introduction

1.1. Purpose

SML was initiated by PEPPOL [REF2] The PEPPOL SML specification was submitted as input to the OASIS BDXR TC (Business Document Exchange Technical Committee) with the intent of defining a standardized and federated document transport infrastructure for business document exchange. They resulted into a new committee specification: BDXL (Business Document Metadata Service Location) [REF3].

In WP6 [REF4] , e-SENS defines the Service Location ABB based upon OASIS BDXL specification, compliant with the legacy SML specification.

The eDelivery Business Document Metadata Service Location application (BDMSL) is the sample implementation of the Service Location ABB.

This document is the Software Architecture document of the eDelivery Business Document Metadata Service Location application (BDMSL) sample implementation. It intends to provide detailed information about the project:

- An overview of the solution
- The different layers
- The principles governing its software architecture

1.2. References

#	Document	Contents outline
[REF1]	SML Specification	Defines the profiles for the discovery and management interfaces for the Business Document Exchange Network (BUSDOX) Service Metadata Locator service.
[REF2]	PEPPOL	The OpenPEPPOL Association is responsible for the governance and maintenance of the PEPPOL specifications that enable European businesses to easily deal electronically with any European public sector buyer in their procurement processes.
[REF3]	OASIS Business Document Metadata Service Location Version 1.0 (BDXL)	This specification defines service discovery methods. A method is first specified to query and retrieve a URL for metadata services. Two metadata service types are then defined. Also an auxiliary method pattern for discovering a registration service to enable access to metadata services is described. The methods

#	Document	Contents outline
		defined here are instances of the generic pattern defined within IETF RFCs for Dynamic Delegation Discovery Services (DDDS). This specification then defines DDDS applications for metadata and metadata-registration services.
[REF4]	WP6 from eSENS	Work Package 6, eSENS

2. Overview of the solution

The eDelivery BDMSL is a solution conformant with both the SML specification and BDXL Core Implementation Conformance specification.

2.1. Service Metadata Locator (SML)

The SML is the only centrally operated component in the eDelivery Messaging Infrastructure. The dynamic discovery process begins with the establishment of the Service Metadata relating to the particular gateway to which a sender wants to transmit a message. To find the address of the Service Metadata of a participant, the Service Metadata Locator [REF1] service specification is based on the use of DNS (Domain Name System) lookups.

2.2. Business Document Metadata Service Location (BDXL)

The functionality of SML has been subsumed in a more general technical specification called the Business Document Metadata Service Location (BDXL) [REF3].This specification is based on DNS, like SML, but is based on a different type of DNS resource records called URI-enabled Naming Authority Pointer records (U-NAPTR), which are defined to support Dynamic Delegation Discovery Service (DDDS). The result of a query is a full URI, which can use HTTPS and supports server (and optionally client) authentication.

3. Presentation of the different layers

A multi-layered architecture requires respecting some principles:

- Structure of the java packages: in a project, every layer is represented as a package containing all the Java components of this layer
- Calls between the layers: The calls must respect the hierarchy of the layers and must be performed only by using interfaces as shown in the diagram below:

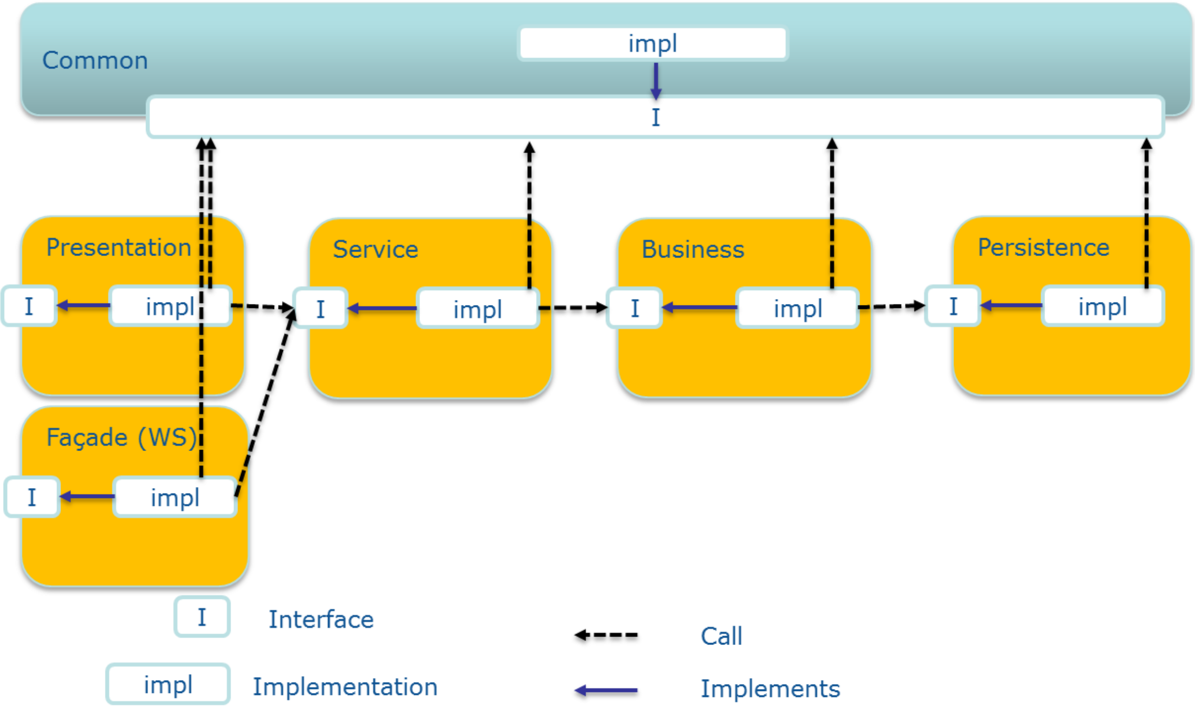


Figure 1 - Inter-layers interactions

3.1. Presentation layer

The presentation layer manages the Graphical User Interface (GUI: pages, graphical components, etc.) and the page flow.

This layer mainly handles:

- The GUI
- Interaction with the user
- The page flow
- User sessions
- Calls to the service layer through a controller

3.1.1. Naming convention

- Java package for controller: eu.europa.ec.bdmsl.presentation.controller
- Implementation: eu.europa.ec.bdmsl.presentation.controller.<PageName>Controller
- Folder for the JSP files : src/main/webapp/WEB-INF/jsp

3.1.2. Dependencies

In this layer, only the following calls are allowed:

- Calls to technical components
- Calls to the service layer
- Calls to the common package

3.1.3. Frameworks and patterns

The presentation layer relies on the Spring MVC (Model-View-Controller) framework.

The views are represented by JSP files. These files are bound to controllers. The models are built from calls to the service layer. They are then returned to the views. The presentation layer includes the controllers.

For instance, this is the `ListDNSController` implementation class:

```
package eu.europa.ec.bdmsl.presentation.controller;

[...]

@Controller
public class ListDNSController {

    @Autowired
    private ILoggingService loggingService;

    @Value("${dnsClient.enabled}")
    private String dnsEnabled;

    @Value("${dnsClient.server}")
    private String dnsServer;

    // Path to the service
    @RequestMapping("/listDNS")
    public String listDNS(Model model) {
        loggingService.debug("Calling listDNS...");
        [...]
        // We can add any object in the model and retrieve them in the views
        model.addAttribute("dnsEnabled", dnsEnabled);
        // bound to listDNS.jsp file in the src/main/webapp/WEB-INF/jsp folder
        return "listDNS";
    }
}
```

In the `listDNS.jsp` file, the model can be accessed like this:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <head>
        <title>BDMSL Service</title>
    </head>
    <body>
        <h1>ListDNS</h1>
        <c:choose>
            <!-- Access to the model -->
            <c:when test="${dnsEnabled}">
                [...]
            </c:when>
            <c:otherwise>
                <ul>
                    <li>The DNS client is disabled.</li>
                </ul>
            </c:otherwise>
        </c:choose>
    </body>
</html>
```

3.2. Service layer

The service layer is the most important layer of the application as it coordinates the calls to the business rules.

The service layer handles the transaction management. It creates the transaction and instantiates the technical objects required (sessions, connection, etc.). The transactions manage the commit/rollback depending on the errors raised by the different layers that it calls (service, business, and persistence).

The transaction management is handled by Spring through the use of the annotation `@Transactional`. Spring transparently encapsulates the calls to the different services and create if necessary transactions if the configuration requires it.

By default, the transactions are in "read-only" mode (attribute `read-only = true`) at the class level.

```
...
@Transactional(readOnly = true)
public class ManageServiceMetadataServiceImpl extends AbstractServiceImpl
implements IManageServiceMetadataService {
...
    @Transactional(readOnly = false, rollbackFor = Exception.class)
    public void create(final ServiceMetadataPublisherBO smpBO) throws
BusinessException, TechnicalException {
     ...
    }

    public ServiceMetadataPublisherValue read(ServiceMetadataPublisherBO
messagePartBO) throws BusinessException, TechnicalException {
     ...
   }
}
```

The service layer performs different calls to the service/business layers.

The objects from the service layer are POJO that implement the singleton pattern. The objects from the different layers are injected by Spring by setters.

3.2.1. Naming convention

Naming convention for:

- Package: eu.europa.ec.bdmsl.service
- Interface: eu.europa.ec.bdmsl.service.I<InterfaceName>Service
- Implementation package: eu.europa.ec.bdmsl.service.impl

- Implementation: eu.europa.ec.bdmsl.service.impl.<InterfaceName>ServiceImpl

3.2.2. Dependencies

In this layer, only the following calls are allowed:

- Calls to technical components
- Calls to the business layer
- Calls to the common package

3.2.3. Frameworks and patterns

The service layer uses these frameworks:

- Spring: transaction management, exception handling, dependency injection

3.2.4. Development of the service layer

3.2.4.1. Interface

```
public interface IManageServiceMetadataService {

  /**
   * Retrieves the Service Metadata Publisher record for the service metadata.
   * @param serviceMetadataPublisherID the unique ID
   *           of the Service Metadata Publisher for which the record is required
   * @return ServiceMetadataPublisherBO the service metadata publisher record.
   * @throws TechnicalException Technical exception.
   * @throws BusinessException Business exception.
   */
  ServiceMetadataPublisherBO read(String serviceMetadataPublisherID)
    throws TechnicalException, BusinessException;
  ...
}
```

3.2.4.2. Implementation classes

The implementation classes extend the parent-class «`AbstractServiceImpl`» and implement their dedicated interface (here «`IManageServiceMetadataService`»).

```
@Transactional(readOnly = true)
public class ManageServiceMetadataServiceImpl extends AbstractServiceImpl
implements IManageServiceMetadataService {

  private IManageServiceMetadataServiceBusiness
manageServiceMetadataServiceBusiness;

  /*
   * (non-Javadoc)
   *
   * @see eu.europa.ec.bdmsl.service.IManageServiceMetadataService#read (String)
   */
  @Override
  @Transactional(readOnly = true)
  public ServiceMetadataPublisherBO read(String serviceMetadataPublisherID)
    throws TechnicalException, BusinessException {
    ServiceMetadataPublisherBO smpBO =
manageServiceMetadataServiceBusiness.read(serviceMetadataPublisherID);
```

```
      return smpValue;
   }
   ...
}
```

The parent-class «AbstractServiceImpl» contains all common attributes and methods of the implementation classes of the service layer (logging service, etc.).

3.2.4.3. Transaction configuration

The transaction management is managed by Spring.

The JDBC connections to the database are open by Spring if the processing of the service requires access to the database (though the call to the business layer).

The configuration of the transaction management is made by annotations. These annotations define the rollback policy when certain types of exception may be raised. Indeed, if a non-critical error is raised, it could be useful to perform a commit anyway.

The annotations for the transaction management are set in the service class implementations with @Transactional. This way, we scan specify which interface and methods are executed in a transactional context.

There are two types of transaction modes:

- **read-only** is used by default: can perform read actions but cannot write anything in the database.
- **read-write** is used for CUD methods (Create, Update, Delete).

Propagation attributes manage the opening of the transactions. In this project, the attribute REQUIRED is used. This attribute, which is used by default, means that the method must be executed in a transaction context. If the transaction does not exist at the time of the call, a new one is created.

Thus, only one transaction is allowed for a call to a method in the service layer. If the method calls itself other services, the transaction will be propagated.

By default, Spring performs a rollback when a runtime exception is thrown. This behaviour can be modified with the attribute rollbackFor by passing a list of exceptions for which a rollback will be performed. In the BDMSL component, we rollback for any type of exception (checked and runtime), so we set the following value: rollbackFor = Exception.class.

3.3. Web service package

This chapter describes the use of the Apache CXF framework in the web service package to expose SOAP and REST web services.

A web service is a service that can be remotely invoked by another system.

The services of the eDelivery BDMSL application are declared in Spring and implement the Singleton pattern.

In order to connect the classes exposed by CXF to the service class managed by Spring, we use an additional package that plays the role of **Façade**: `eu.europa.ec.bdmsl.ws`

The façades define the same interfaces as the services they are linked to. They have the same methods as the Java implementation classes of the services managed by Spring. The façades implement strictly the interface they reference and serve as a transition with the external systems, taking into consideration matters like database connection, transaction, security, etc. The façade also performs the conversion of the objects from JAXB to BO and vice-versa.

The reference to the underlying service is injected in the façade with Spring. For each method of the interface implemented by the façade, we invoke the same method as on the service implementation class:

```
[...]
public class ManageParticipantIdentifierWSImpl extends AbstractWSImpl implements
IManageParticipantIdentifierWS {

    @Autowired
    private IManageParticipantIdentifierService
manageParticipantIdentifierService;

    @Autowired
    private MapperFactory mapperFactory;

    [...]

    @Override
    @WebResult(name = "ParticipantIdentifierPage", targetNamespace =
"http://busdox.org/serviceMetadata/locator/1.0/", partName = "messagePart")
    @WebMethod(operationName = "List", action =
"http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:listIn")
    public ParticipantIdentifierPageType list(@WebParam(partName =
"pageRequestType", name = "PageRequest", targetNamespace =
"http://busdox.org/serviceMetadata/locator/1.0/") PageRequestType
pageRequestType) throws NotFoundFault, InternalErrorFault, UnauthorizedFault,
BadRequestFault {
        ParticipantIdentifierPageType result = null;
        try {
            [...]
            // convert input from JAXB to BO
            PageRequestBO pageRequestBO =
mapperFactory.getMapperFacade().map(pageRequestType, PageRequestBO.class);

            // call the service layer
            ParticipantListBO resultParticipantBOList =
manageParticipantIdentifierService.list(pageRequestBO);
            loggingService.businessLog(LogEvents.BUS_PARTICIPANT_LIST,
pageRequestBO.getSmpId());

            // convert output from BO to JAXB
            result = mapperFactory.getMapperFacade().map(resultParticipantBOList,
ParticipantIdentifierPageType.class);
        } catch (Exception exc) {
            [...]
            handleException(exc);
        }
        return result;
    }
    ...
}
```

3.3.1. Dependencies

In this package, only the following calls are allowed:

- Calls to technical components
- Calls to the service layer
- Calls to the common package

3.3.2. SOAP web services

This section describes the general principles governing SOAP web services.

3.3.2.1. Naming convention

- Package: eu.europa.ec.bdmsl.ws.soap
- Interface: eu.europa.ec.bdmsl.ws.soap.I<InterfaceName>WS
- Implementation package: eu.europa.ec.bdmsl.ws.soap.impl
- Implementation: eu.europa.ec.bdmsl.ws.soap.impl.<InterfaceName>WSImpl

3.3.2.2. Exposing a web service

As from Java 5, the JSR 181, implemented in Apache CXF, allows declaring a Java class as a SOAP web service.

To declare a façade as a web service class, the use of the annotations `@WebService`, `@SOAPBinding` et `@BindingType` is required as follow:

```
@WebService(serviceName = "ManageServiceMetadataService", portName =
"ManageServiceMetadataServicePort", targetNamespace =
Constants.MANAGE_METADATA_SERVICE_NS)
@SOAPBinding(style = SOAPBinding.Style.DOCUMENT, use = SOAPBinding.Use.LITERAL)
@BindingType(javax.xml.ws.soap.SOAPBinding.SOAP11HTTP_BINDING)
public class ManageServiceMetadataWSImpl {
...
}
```

NB: these annotations are provided by the JSR 181 API and must be set at both the implementation class and interfaces level.

To avoid the exposition of some methods like getter/setters, we use the annotation `@WebMethod(exclude=true)`.

3.3.2.3. Declaration of the exposed services

The endpoint and the implementing classes are defined in the `cxf-servlet.xml` file. This is how we can expose the service `bdmslservice`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://cxf.apache.org/jaxws
        http://cxf.apache.org/schemas/jaxws.xsd">
```

```
    [...]

 <jaxws:endpoint
   id="bdmslService"
   implementor="eu.europa.ec.bdmsl.ws.soap.impl.BDMSLServiceWSImpl"
   address="/bdmslservice">
 </jaxws:endpoint>

</beans>
```

3.3.2.4. WSDL

The exposed web services are defined in a contract interface file named WSDL. In the eDelivery BDMSL application, we use a **WSDL-first approach**. It means that we first design the WSDL and generate Java code from the WSDL.

The classes are generated through the use of a maven plugin defined in the `pom.xml` file. To generate the classes, the following command line must be run. Among other operations, this command will automatically call the `wsdl2java` goal from the `cxf-codegen-plugin` plugin:

```
mvn package
```

For the SML compliance, the WSDL files are defined in the SML specification. In the eDelivery BDMSL application, they are:

- ManageBusinessIdentifierService-1.0.wsdl
- ManageServiceMetadataService-1.0.wsdl
- BDMSLService-1.0.wsdl

These files are located in the `src/main/webapp/WEB-INF/wsdl` directory.

3.3.2.5. Binding

The use of JAXB configuration allows customizing the generated sources like package or class names, and also allowing the definition of adapters between XML and Java types (marshalling/unmarshalling).

These binding files are stored as `xjb` files in the `src/main/webapp/WEB-INF/wsdl` directory.

3.3.2.6. Mapping JAXB objects / BO

JAXB objects are generated from the WSDL. Inside the different layers of the application, we only use Business Objects (BO). We don't directly use JAXB generated objects in the business logic because this would tightly couple the business logic to the WSDL. A schema change in a WSDL (such as for version update) typically leads to a different package structure for classes generated from that WSDL via JAXB. To mitigate this risk, we use different Java objects: the business objects (BO). For more information on the mapping of JAXB objects to BO, see the chapter ***8 Object mapping.***

3.3.2.7. Frameworks and patterns

- Apache CXF: Exposing SOAP/REST web services. Only in the web module service.
- Spring: dependency injection

3.3.3. Services specifications

This paragraph provides implementation details of the BDMSL.

There are 3 interfaces described in this paragraph:

Interface	Description
ManageServiceMetadataService-1.0.wsdl	Defined in the PEPPOL SML specification [REF1], in Appendix B: WSDLs.
ManageBusinessIdentifierService-1.0.wsdl	Defined in the PEPPOL SML specification [REF1], in Appendix B: WSDLs.
BDMSLService-1.0.wsdl	Contains services not covered by any specification from OASIS or PEPPOL, used by the SMP user.
BDMSLAdminService-1.0.wsdl	Contains administration services for managing the SML instance.

3.3.3.1. ManageService Metadata Service

3.3.3.1.1. WSDL file

- ManageServiceMetadataService-1.0.wsdl

3.3.3.1.2. Operation Create()

3.3.3.1.2.1. Pre-requisites

- The user has a valid certificate[1]
- The role associated to the certificate is ROLE_SMP
- The SMP doesn't already exists in the system

3.3.3.1.2.2. Description

Establishes a Service Metadata Publisher metadata record, containing the metadata about the Service Metadata Publisher (SMP), as outlined in the ServiceMetadataPublisherService data type.

- Input CreateServiceMetadataPublisherService: ServiceMetadataPublisherService - contains the service metadata publisher information, which includes the logical and physical addresses for the SMP (Domain name and IP address). It is assumed that the ServiceMetadataPublisherID has been assigned to the calling user out-of-bands.
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Create operation.
- Fault: badRequestFault - returned if the supplied CreateServiceMetadataPublisherService does not contain consistent data.

[1] In this document, we consider that a certificate is valid if it is not revoked and not expired.

- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason.

3.3.3.1.2.3. Technical design

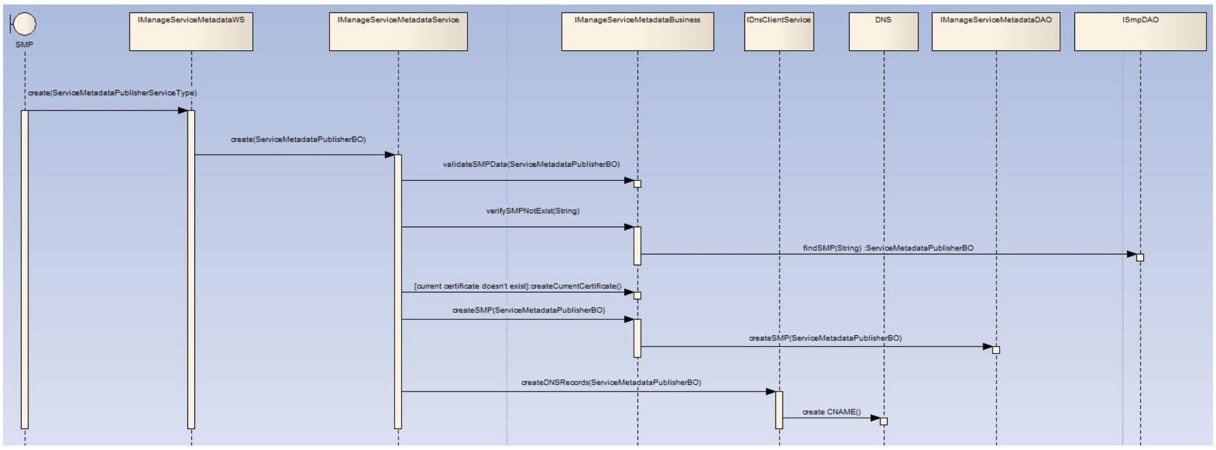

SMP
IManageServiceMetadataWS
IManageServiceMetadataService
IManageServiceMetadataBusiness
IDnsClientService
DNS
IManageServiceMetadataDAO
ISmpDAO
create(ServiceMetadataPublisherServiceType)
create(ServiceMetadataPublisherBO)
validateSMPData(ServiceMetadataPublisherBO)
verifySMPNotExist(String)
findSMP(String) :ServiceMetadataPublisherBO
[current certificate doesn't exist]:createCurrentCertificate()
createSMP(ServiceMetadataPublisherBO)
createSMP(ServiceMetadataPublisherBO)
createDNSRecords(ServiceMetadataPublisherBO)
create CNAME()

Figure 2 - Sequence Diagram - ManageServiceMetadata Create

3.3.3.1.3. Operation Read()

3.3.3.1.3.1. Pre-requisites

- The user has a valid certificate
- The role of the user is ROLE_SMP
- The SMP already exists

3.3.3.1.3.2. Description

Retrieves the Service Metadata Publisher record for the service metadata publisher.

- Input ReadServiceMetadataPublisherService: ServiceMetadataPublisherID - the unique ID of the Service Metadata Publisher for which the record is required
- Output: ServiceMetadataPublisherService - the service metadata publisher record, in the form of a ServiceMetadataPublisherService data type
- Fault: notFoundFault - returned if the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Read operation
- Fault: badRequestFault - returned if the supplied parameter does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.1.3.3. Technical design

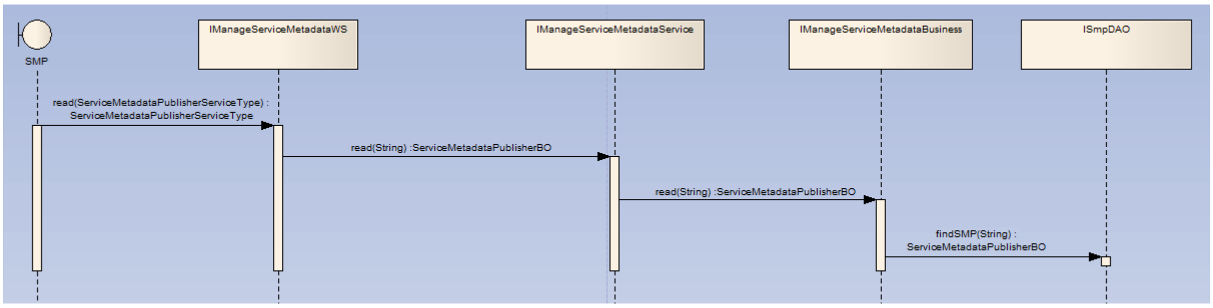

SMP
IManageServiceMetadataWS
IManageServiceMetadataService
IManageServiceMetadataBusiness
ISmpDAO
read(ServiceMetadataPublisherServiceType) :
ServiceMetadataPublisherServiceType
read(String) :ServiceMetadataPublisherBO
read(String) :ServiceMetadataPublisherBO
findSMP(String) :
ServiceMetadataPublisherBO

Figure 3 - Sequence Diagram - ManageServiceMetadata Read

3.3.3.1.4. Operation Update()

3.3.3.1.4.1. Pre-requisites

- The user has a valid certificate
- The role of the user is ROLE_SMP
- The SMP already exists

3.3.3.1.4.2. Description

Updates the Service Metadata Publisher record for the service metadata publisher.

- Input UpdateServiceMetadataPublisheServicer: ServiceMetadataPublisherService - contains the service metadata for the service metadata publisher, which includes the logical and physical addresses for the SMP (Domain name and IP address). If the request's logical address is different from the logical address stored into the database, all participant's NAPTR records under the specified SMP will be updated with the new logical address passed by request.
- Fault: notFoundFault - returned if the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Update operation
- Fault: badRequestFault - returned if the supplied UpdateServiceMetadataPublisheServicer does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.1.4.3. Technical design

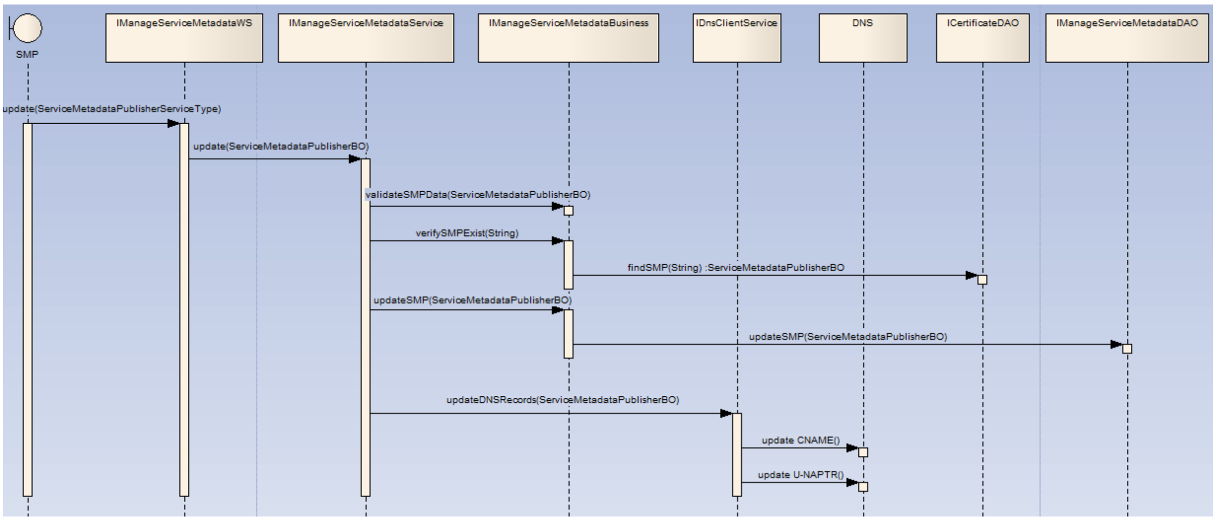

SMP
IManageServiceMetadataWS
IManageServiceMetadataService
IManageServiceMetadataBusiness
IDnsClientService
DNS
ICertificateDAO
IManageServiceMetadataDAO
update(ServiceMetadataPublisherServiceType)
update(ServiceMetadataPublisherBO)
validateSMPData(ServiceMetadataPublisherBO)
verifySMPExist(String)
findSMP(String) :ServiceMetadataPublisherBO
updateSMP(ServiceMetadataPublisherBO)
updateSMP(ServiceMetadataPublisherBO)
updateDNSRecords(ServiceMetadataPublisherBO)
update CNAME()
update U-NAPTR()

Figure 4 - Sequence Diagram - ManageServiceMetadata Update

3.3.3.1.5. Operation Delete()

3.3.3.1.5.1. Pre-requisites

- The user has a valid certificate
- The role of the user is ROLE_SMP
- The SMP already exists

3.3.3.1.5.2. Description

Deletes the Service Metadata Publisher record for the service metadata publisher.

- Input DeleteServiceMetadataPublisherService: ServiceMetadataPublisherID - the unique ID of the Service Metadata Publisher to delete
- Fault: notFoundFault - returned if the identifier of the SMP could not be found

- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Delete operation
- Fault: badRequestFault - returned if the supplied DeleteServiceMetadataPublisherService does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

Implementation note: If the SMP is linked to many participants, then the participants are deleted from the database and the DNS by batch of 300 elements. This is to avoid reaching the limit of the DNS protocol. Indeed, the RFC1035 of the DNS standard states that the messages are bound to 65535 bytes length.

3.3.3.1.5.3. Technical design

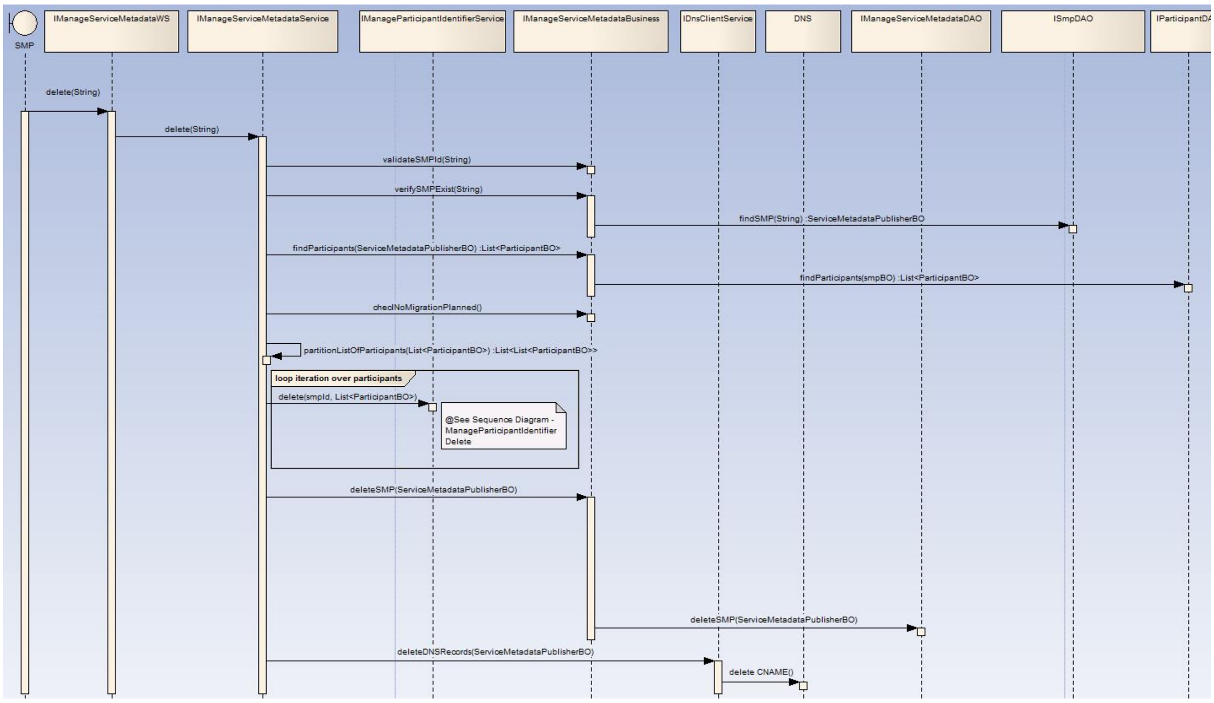

SMP
IManageServiceMetadataWS
IManageServiceMetadataService
IManageParticipantIdentifierService
IManageServiceMetadataBusiness
IDnsClientService
DNS
IManageServiceMetadataDAO
ISmpDAO
IParticipantDA
delete(String)
delete(String)
validateSMPId(String)
verifySMPExist(String)
findSMP(String) :ServiceMetadataPublisherBO
findParticipants(ServiceMetadataPublisherBO) :List<ParticipantBO>
findParticipants(smpBO) :List<ParticipantBO>
checlNoMigrationPlanned()
partitionListOfParticipants(List<ParticipantBO>) :List<List<ParticipantBO>>
loop iteration over participants
delete(smpId, List<ParticipantBO>)
@See Sequence Diagram - ManageParticipantIdentifier Delete
deleteSMP(ServiceMetadataPublisherBO)
deleteSMP(ServiceMetadataPublisherBO)
deleteDNSRecords(ServiceMetadataPublisherBO)
delete CNAME()

Figure 5 - Sequence Diagram - ManageServiceMetadata Delete

3.3.3.2. Manage Participant Identifier

3.3.3.2.1. WSDL file

- ManageBusinessIdentifierService-1.0.wsdl

3.3.3.2.2. Operation Create()

3.3.3.2.2.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The role of the user is ROLE_SMP
- The participants do not exist yet

3.3.3.2.2.2. Description

Creates an entry in the Service Metadata Locator Service for information relating to a specific participant identifier. Regardless of the number of services a recipient exposes, only one record corresponding to the participant identifier is created in the Service Metadata Locator Service by the Service Metadata Publisher which exposes the services for that participant.

- Input CreateParticipantIdentifier: ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP which holds its data
- Fault: notFoundFault - returned if the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Create operation
- Fault: badRequestFault - returned if the supplied CreateParticipantIdentifier does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.2.3. Technical Design

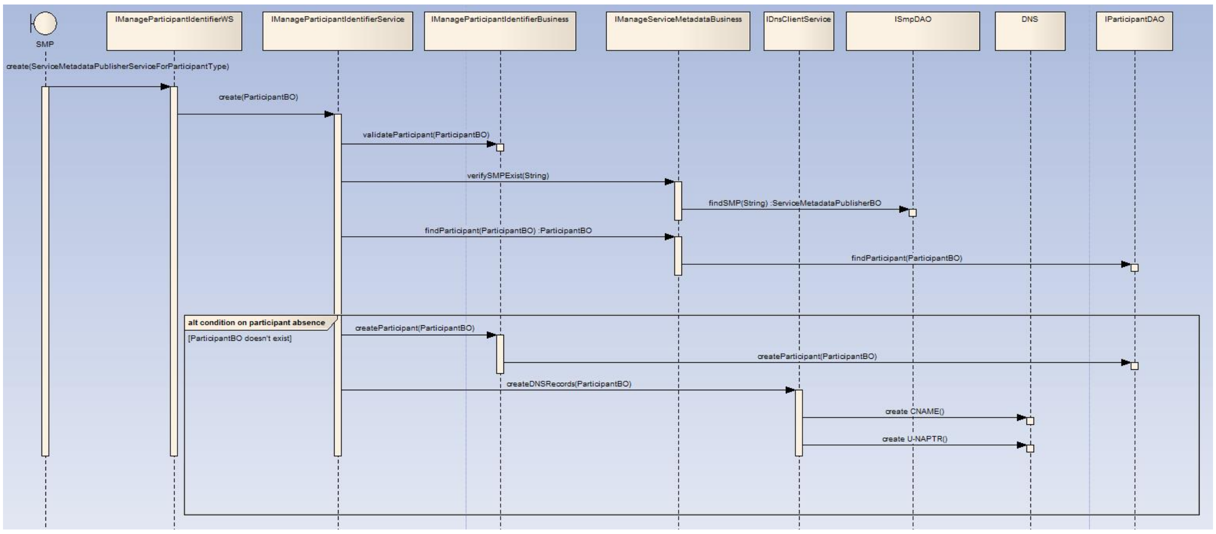

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageParticipantIdentifierBusiness
IManageServiceMetadataBusiness
IDnsClientService
ISmpDAO
DNS
IParticipantDAO
create(ServiceMetadataPublisherServiceForParticipantType)
create(ParticipantBO)
validateParticipant(ParticipantBO)
verifySMPExist(String)
findSMP(String) :ServiceMetadataPublisherBO
findParticipant(ParticipantBO) :ParticipantBO
findParticipant(ParticipantBO)
alt condition on participant absence
[ParticipantBO doesn't exist]
createParticipant(ParticipantBO)
createParticipant(ParticipantBO)
createDNSRecords(ParticipantBO)
create CNAME()
create U-NAPTR()

Figure 6 - Sequence Diagram - ManageParticipantIdentifier Create

3.3.3.2.3. Operation CreateList()

3.3.3.2.3.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The participants do not exist yet

- The role of the user is ROLE_SMP
- The number of participants in the list is less than 100

3.3.3.2.3.2. Description

Creates a set of entries in the Service Metadata Locator Service for information relating to a list of participant identifiers. Regardless of the number of services a recipient exposes, only one record corresponding to each participant identifier is created in the Service Metadata Locator Service by the Service Metadata Publisher which exposes the services for that participant.

- Input CreateList: ParticipantIdentifierPage - contains the list of Participant Identifiers for the participants which are added to the Service Metadata Locator Service. The NextPageIdentifier element is absent.
- Fault: notFoundFault - returned if the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the CreateList operation
- Fault: badRequestFault - returned if:
- The supplied CreateList does not contain consistent data
- The number of participants in the list is greater than 100
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.3.3. Technical Design

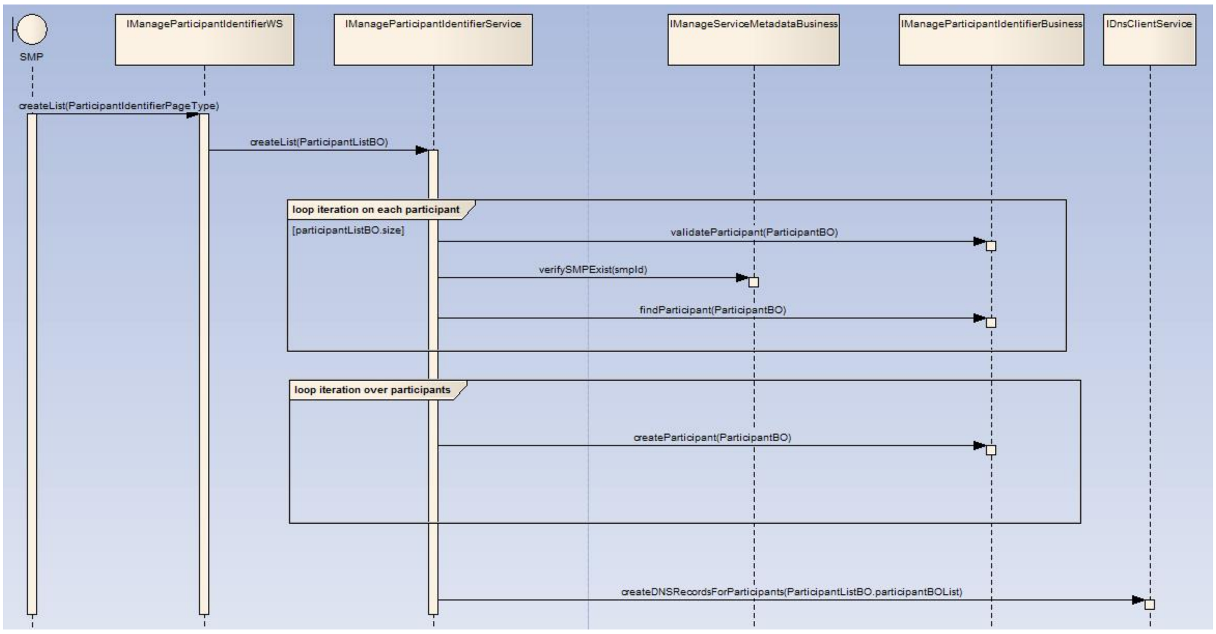

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageServiceMetadataBusiness
IManageParticipantIdentifierBusiness
IDnsClientService
createList(ParticipantIdentifierPageType)
createList(ParticipantListBO)
loop iteration on each participant
[participantListBO.size]
validateParticipant(ParticipantBO)
verifySMPExist(smpId)
findParticipant(ParticipantBO)
loop iteration over participants
createParticipant(ParticipantBO)
createDNSRecordsForParticipants(ParticipantListBO.participantBOList)

Figure 7 - Sequence Diagram - ManageParticipantIdentifier CreateList

3.3.3.2.4. Operation Delete()

3.3.3.2.4.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The role of the user is ROLE_SMP
- The participants already exist
- The participant is not under migration process and does not have an active migration key

3.3.3.2.4.2. Description

Deletes the information that the SML Service holds for a specific Participant Identifier.

- Input DeleteParticipantIdentifier: ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP that publishes its metadata
- Fault: notFoundFault - returned if the participant identifier or the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Delete operation or if the migration to another SMP is in progress
- Fault: badRequestFault - returned if the supplied DeleteParticipantIdentifier does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.4.3. Technical Design

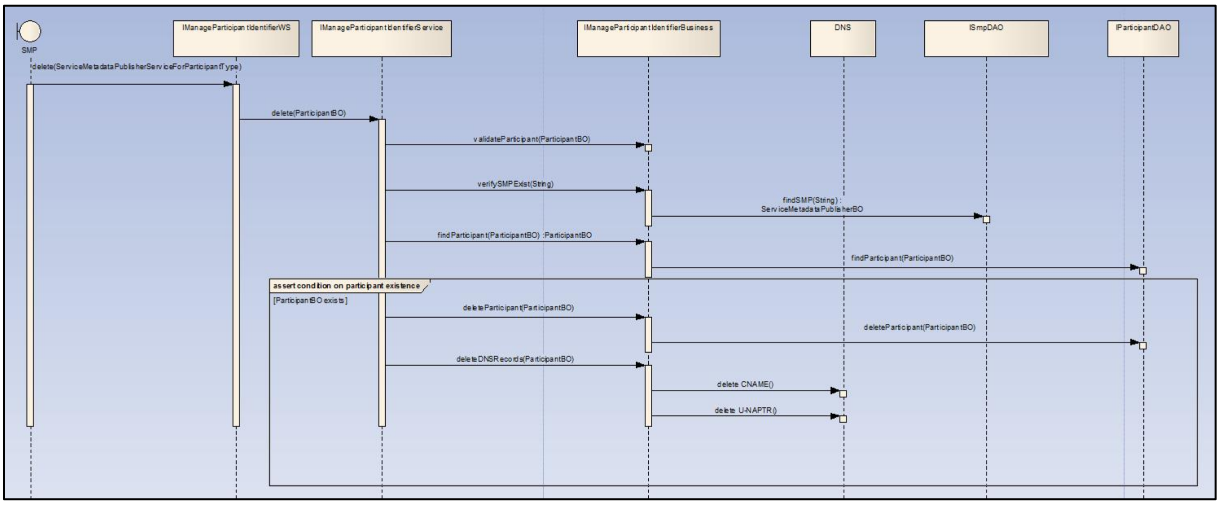

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageParticipantIdentifierBusiness
DNS
ISmpDAO
IParticipantDAO
delete(ServiceMetadataPublisherServiceForParticipantType)
delete(ParticipantBO)
validateParticipant(ParticipantBO)
verifySMPExist(String)
findSMP(String) : ServiceMetadataPublisherBO
findParticipant(ParticipantBO) :ParticipantBO
findParticipant(ParticipantBO)
assert condition on participant existence
[ParticipantBO exists]
deleteParticipant(ParticipantBO)
deleteParticipant(ParticipantBO)
deleteDNSRecords(ParticipantBO)
delete CNAME()
delete U-NAPTR()

Figure 8 - Sequence Diagram - ManageParticipantIdentifier Delete

3.3.3.2.5. Operation DeleteList()

3.3.3.2.5.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The participants already exist
- The participants are not under migration process
- The role of the user is ROLE_SMP
- The number of participants in the list is less than 100

3.3.3.2.5.2. Description

Deletes the information that the SML Service holds for a list of Participant Identifiers.

- Input DeleteList: ParticipantIdentifier - contains the list of Participant Identifiers for the participants which are removed from the Service Metadata Locator Service. The NextPageIdentifier element is absent.
- Fault: notFoundFault - returned if one or more participant identifiers or the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the DeleteList operation or if one of the participants is under migration process
- Fault: badRequestFault - returned if:
- The supplied DeleteList does not contain consistent data
- The number of participants in the list is greater than 100

- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.5.3. Technical Design

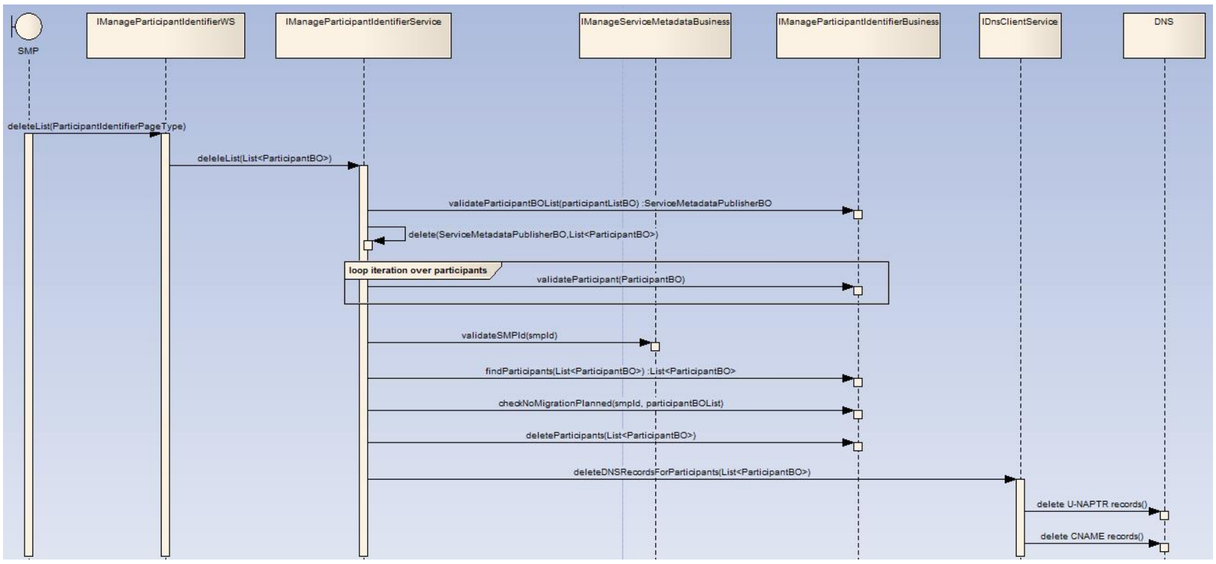

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageServiceMetadataBusiness
IManageParticipantIdentifierBusiness
IDnsClientService
DNS
deleteList(ParticipantIdentifierPageType)
deleleList(List<ParticipantBO>)
validateParticipantBOList(participantListBO) :ServiceMetadataPublisherBO
delete(ServiceMetadataPublisherBO,List<ParticipantBO>)
loop iteration over participants
validateParticipant(ParticipantBO)
validateSMPId(smpId)
findParticipants(List<ParticipantBO>) :List<ParticipantBO>
checkNoMigrationPlanned(smpId, participantBOList)
deleteParticipants(List<ParticipantBO>)
deleteDNSRecordsForParticipants(List<ParticipantBO>)
delete U-NAPTR records()
delete CNAME records()

Figure 9 - Sequence Diagram - ManageParticipantIdentifier DeleteList

3.3.3.2.6. Operation PrepareToMigrate()

3.3.3.2.6.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The role of the user is ROLE_SMP
- The participants already exist

3.3.3.2.6.2. Description

Prepares a Participant Identifier for migration to a new Service Metadata Publisher. This operation is called by the Service Metadata Publisher which currently publishes the metadata for the Participant Identifier. The Service Metadata Publisher supplies a Migration Code which is used to control the migration process. The Migration Code must be passed (out of band) to the Service Metadata Publisher which is taking over the publishing of the metadata for the Participant Identifier and which MUST be used on the invocation of the Migrate() operation. This operation can only be invoked by the Service Metadata Publisher which currently publishes the metadata for the specified Participant Identifier.

- Input PrepareMigrationRecord: MigrationRecordType - contains the Migration Key and the Participant Identifier which is about to be migrated from one Service Metadata Publisher to another.
- Fault: notFoundFault - returned if the participant identifier or the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the PrepareToMigrate operation
- Fault: badRequestFault - returned if the supplied PrepateMigrationRecord does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.6.3. Technical Design

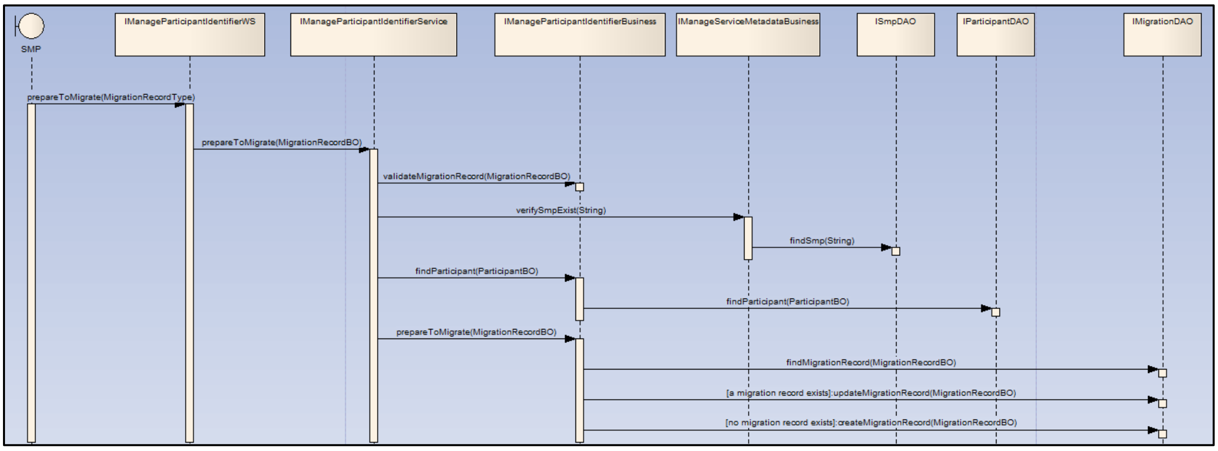

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageParticipantIdentifierBusiness
IManageServiceMetadataBusiness
ISmpDAO
IParticipantDAO
IMigrationDAO
prepareToMigrate(MigrationRecordType)
prepareToMigrate(MigrationRecordBO)
validateMigrationRecord(MigrationRecordBO)
verifySmpExist(String)
findSmp(String)
findParticipant(ParticipantBO)
findParticipant(ParticipantBO)
prepareToMigrate(MigrationRecordBO)
findMigrationRecord(MigrationRecordBO)
[a migration record exists]:updateMigrationRecord(MigrationRecordBO)
[no migration record exists]:createMigrationRecord(MigrationRecordBO)

Figure 10 - Sequence Diagram - ManageParticipantIdentifier PrepareToMigrate

3.3.3.2.7. Operation Migrate()

3.3.3.2.7.1. Pre-requisites

- The user has a valid certificate
- The SMP already exists
- The participants already exist
- The role of the user is ROLE_SMP
- The prepareToMigrate service has been called for this participant

3.3.3.2.7.2. Description

Migrates a Participant Identifier already held by the Service Metadata Locator Service to target a new Service Metadata Publisher. This operation is called by the Service Metadata Publisher which is taking over the publishing for the Participant Identifier. The operation requires the new Service Metadata Publisher to provide a migration code which was originally obtained from the old Service Metadata Publisher. The PrepareToMigrate operation MUST have been previously invoked for the supplied Participant Identifier, using the same MigrationCode, otherwise the Migrate() operation fails. Following the successful invocation of this operation, the lookup of the metadata for the service endpoints relating to a particular Participant Identifier will resolve (via DNS) to the new Service Metadata Publisher.

- Input CompleteMigrationRecord: MigrationRecordType - contains the Migration Key and the Participant Identifier which is to be migrated from one Service Metadata Publisher to another.
- Fault: notFoundFault - returned if the migration key or the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the Migrate operation
- Fault: badRequestFault - returned if the supplied CompleteMigrationRecord does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

3.3.3.2.7.3. Technical Design

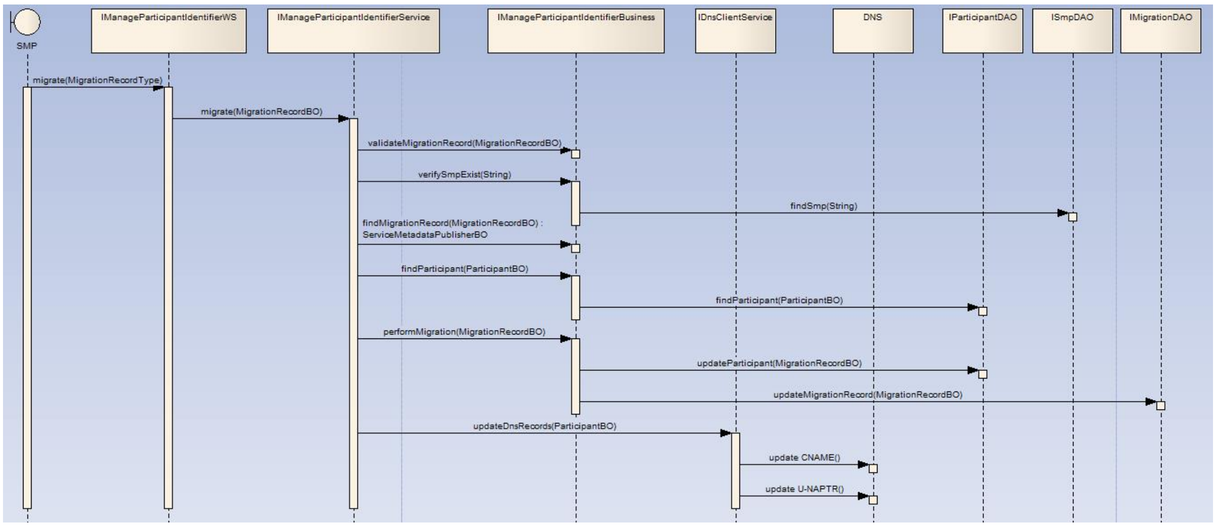

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageParticipantIdentifierBusiness
IDnsClientService
DNS
IParticipantDAO
ISmpDAO
IMigrationDAO
migrate(MigrationRecordType)
migrate(MigrationRecordBO)
validateMigrationRecord(MigrationRecordBO)
verifySmpExist(String)
findSmp(String)
findMigrationRecord(MigrationRecordBO) : ServiceMetadataPublisherBO
findParticipant(ParticipantBO)
findParticipant(ParticipantBO)
performMigration(MigrationRecordBO)
updateParticipant(MigrationRecordBO)
updateMigrationRecord(MigrationRecordBO)
updateDnsRecords(ParticipantBO)
update CNAME()
update U-NAPTR()

Figure 11 - Sequence Diagram - ManageParticipantIdentifier Migrate

3.3.3.2.8. Operation List()

3.3.3.2.8.1. Pre-requisites

- The user has a valid certificate
- The role of the user is ROLE_SMP
- The SMP already exists

3.3.3.2.8.2. Description

List() is used to retrieve a list of all participant identifiers associated with a single Service Metadata Publisher, for synchronization purposes. Since this list may be large, it is returned as pages of data, with each page being linked from the previous page.

- Input Page: PageRequest - contains a PageRequest containing the ServiceMetadataPublisherID of the SMP and (if required) an identifier representing the next page of data to retrieve. If the NextPageIdentifier is absent, the first page is returned.
- Output: ParticipantIdentifierPage - a page of Participant Identifier entries associated with the Service Metadata Publisher, also containing a <Page/> element containing the identifier that represents the next page, if any.
- Fault: notFoundFault - returned if the next page or the identifier of the SMP could not be found
- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the List operation
- Fault: badRequestFault - returned if the supplied NextPage does not contain consistent data
- Fault: internalErrorFault - returned if the SML service is unable to process the request for any reason

Note that the underlying data may be updated between one invocation of List() and a subsequent invocation of List(), so that a set of retrieved pages of participant identifiers may not represent a consistent set of data.

3.3.3.2.8.3. Technical Design

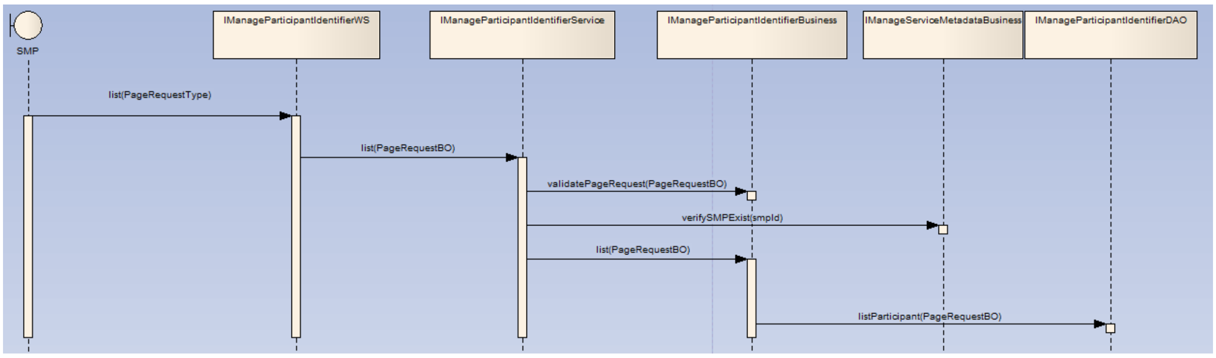

SMP
IManageParticipantIdentifierWS
IManageParticipantIdentifierService
IManageParticipantIdentifierBusiness
IManageServiceMetadataBusiness
IManageParticipantIdentifierDAO
list(PageRequestType)
list(PageRequestBO)
validatePageRequest(PageRequestBO)
verifySMPExist(smpId)
list(PageRequestBO)
listParticipant(PageRequestBO)

Figure 12 - Sequence Diagram - ManageParticipantIdentifier List

3.3.3.3. BDMSLService interface

This interface describes non-core services that are not defined in the SML or BDX specifications. The services are used by SMP and Monitor users.

3.3.3.3.1. WSDL file

- BDMSLService-1.0.wsdl

3.3.3.3.2. Operation PrepareChangeCertificate()

3.3.3.3.2.1. Pre-requisites

- The current certificate of the user is valid
- The role of the user is ROLE_SMP
- The user has the new certificate for the SMP(s)

3.3.3.3.2.2. Description

This operation allows an SMP to prepare a change of its certificate. It is typically called when an SMP has a certificate that is about to expire and already has the new one. This operation MUST be called while the certificate that is already registered in the BDMSL is still valid. If the migrationDate is not empty, then the new certificate MUST be valid at the date provided in the migrationDate element. If the migrationDate element is empty, then the "Valid From" date is extracted from the certificate and is used as the migrationDate. In this case, the "Not Before" date of the certificate must be in the future.

- Fault: unauthorizedFault - returned if the caller is not authorized to invoke the PrepareChangeCertificate operation
- Fault: badRequestFault - returned if
    - The supplied request does not contain consistent data
    - The new certificate is not valid at the date provided in the migrationDate element
    - The migrationDate is not in the future.
    - The migrationDate is not provided and the "Not Before" date of the new certificate is not in the future
    - The migrationDate is not provided and the "Valid From" is in the past
- Fault: internalErrorFault - returned if the BDMSL service is unable to process the request for any reason

3.3.3.3.2.3. Technical design

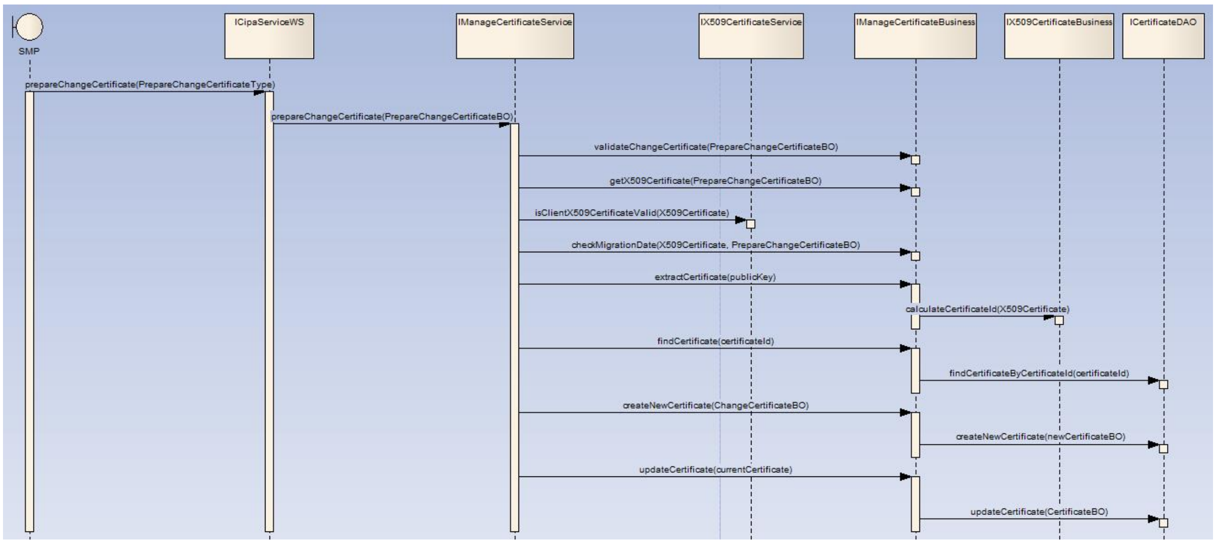

SMP
ICipaServiceWS
IManageCertificateService
IX509CertificateService
IManageCertificateBusiness
IX509CertificateBusiness
ICertificateDAO
prepareChangeCertificate(PrepareChangeCertificateType)
prepareChangeCertificate(PrepareChangeCertificateBO)
validateChangeCertificate(PrepareChangeCertificateBO)
getX509Certificate(PrepareChangeCertificateBO)
isClientX509CertificateValid(X509Certificate)
checkMigrationDate(X509Certificate, PrepareChangeCertificateBO)
extractCertificate(publicKey)
calculateCertificateId(X509Certificate)
findCertificate(certificateId)
findCertificateByCertificateId(certificateId)
createNewCertificate(ChangeCertificateBO)
createNewCertificate(newCertificateBO)
updateCertificate(currentCertificate)
updateCertificate(CertificateBO)

Figure 13 - Sequence Diagram - BDMSLService PrepareChangeCertificate()

3.3.3.3.2.4. Notes

A nightly job performs an analysis to actually perform the change of certificates. The algorithm is as following:

```
List<Certificate> certificates = findCertificateWithPessimisticLock()
for each certificate in certificates do
  if [certificate.new_cert_migration_date <= today] then
    for each allowed_wildcard in bdmsl.allowed_wildcard do
      allowed_wildcard.fk_certificate_id = certificate.new_cert_id
    end for
    for each smp in bdmsl.smp do
      smp.fk_certificate_id = certificate.new_cert_id
    end for
    delete certificate
  else if [certificate.new_cert_migration_date < today] then
    warn "The migration of the certificate couldn't be perform in time"
  end if
end for
```

The scheduling of the job can be configured by setting the value of the property `certificateChangeCronExpression`.

In order to avoid the job to be performed multiple times on a clustered environment, it is necessary to use a pessimistic lock when finding the certificates. The job must run in a single transaction and the lock is released at the end of the transaction.

3.3.3.3.3. Operation IsAlive()

3.3.3.3.3.1. Pre-requisites

- The certificate is valid
- The user has the role ROLE_MONITOR, ROLE_SMP or ROLE_ADMIN

3.3.3.3.3.2. Description

This service has only a monitoring purpose. It can be called to check if the application is up and running.

This service checks if the database and the DNS are accessible by trying to read from the database and to write to and read from DNS.

- Input : none
- Output : none. HTTP 200 OK expected
- Fault: internalErrorFault - returned if the BDMSL service is unable to process the request for any reason

3.3.3.3.4. Operation CreateParticipantIdentifier()

3.3.3.3.4.1. Pre-requisites

- The certificate is valid
- The SMP already exists
- The participant doesn't already exists

3.3.3.3.4.2. Description

This service has the same behaviour as the `Create()` operation in the `ManageParticipantIdentifier` interface but it has one additional and optional input: the `serviceName` element. In the `Create`() operation, the service name is "`Meta:SMP`" by default. In the `CreateParticipantIdentifier()` operation, this service name can be customized.

- serviceName: the name of the service for the NAPTR record

3.3.3.3.4.3. Technical Design

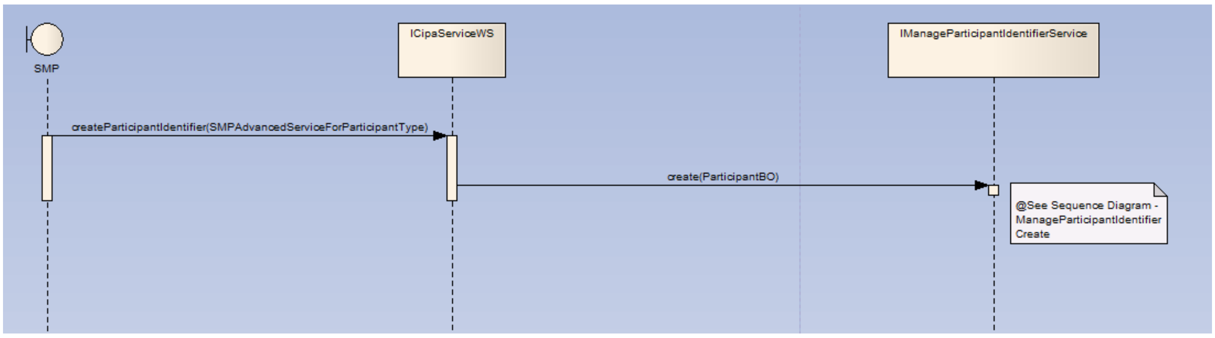

SMP
ICipaServiceWS
IManageParticipantIdentifierService
createParticipantIdentifier(SMPAdvancedServiceForParticipantType)
create(ParticipantBO)
@See Sequence Diagram - ManageParticipantIdentifier Create

Figure 14 - Sequence Diagram - BDMSLService CreateParticipantIdentifier()

Note: the flow for the `create` method of `ManageParticipantIdentifierServiceImpl` can be found here: 3.3.3.2.2.3 Technical Design

3.3.3.4. BDMSLAdminService interface

This interface describes non-core services that are not defined in the SML or BDX specifications. Services are restricted only for role ROLE_ADMIN and it is advised to use the only behind Proxy so that they are not exposed to Internet (They should be used only on intranet).

3.3.3.4.1. Operation ClearCache()

3.3.3.4.1.1. Pre-requisites

- The certificate is a valid certificate
- The user has the role ROLE_SMP or ROLE_ADMIN

3.3.3.4.1.2. Description

The application manages in-memory caches in order to enhance performances. This service can be called to clear all the caches managed by the application. The in-memory caches are used for:

- The list of trusted aliases and their corresponding domains, because these data are not supposed to be changed frequently
- The content of the Certificate Revocation List, in order to avoid the cost of downloading each time the CRLM for each certificate
- Input : none
- Output : none. HTTP 200 OK expected
- Fault: internalErrorFault - returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.1.3. Technical design

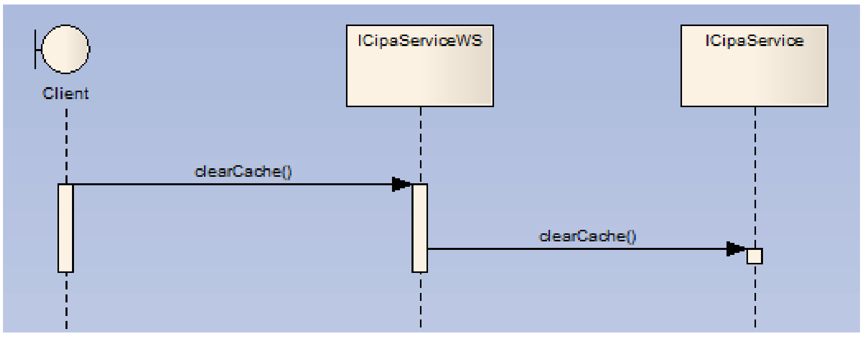


Figure 15 - Sequence Diagram – BDMSLService ClearCache()

3.3.3.4.2. Operation ChangeCertificate()

3.3.3.4.2.1. Pre-requisites

- The user credentials are valid
- The user has the role ROLE_ADMIN
- The user has the new certificate for the SMP

3.3.3.4.2.2. Description

This operation allows the admin team to change the SMP's certificate. It is called by the admin team in case the SMP's certificate has expired and the new one needs to be applied. The new certificate MUST be valid at the date time the request is sent.

- Input : SMP id, Certificate public key
- Output : none. HTTP 200 OK expected
- Fault: unauthorizedFault - returned if:
  - The caller is not authorized to invoke the ChangeCertificate operation (The user doesn't have the ROLE_ADMIN role)
  - The public key already exists
- Fault: badRequestFault - returned if
  - The supplied request does not contain consistent data
  - Invalid public key
  - The new certificate is not valid at the moment the request is sent
  - The SMP id is unknown
- Fault: internalErrorFault - returned if the BDMSL service is unable to process the request for some reason

3.3.3.4.2.3. Technical design

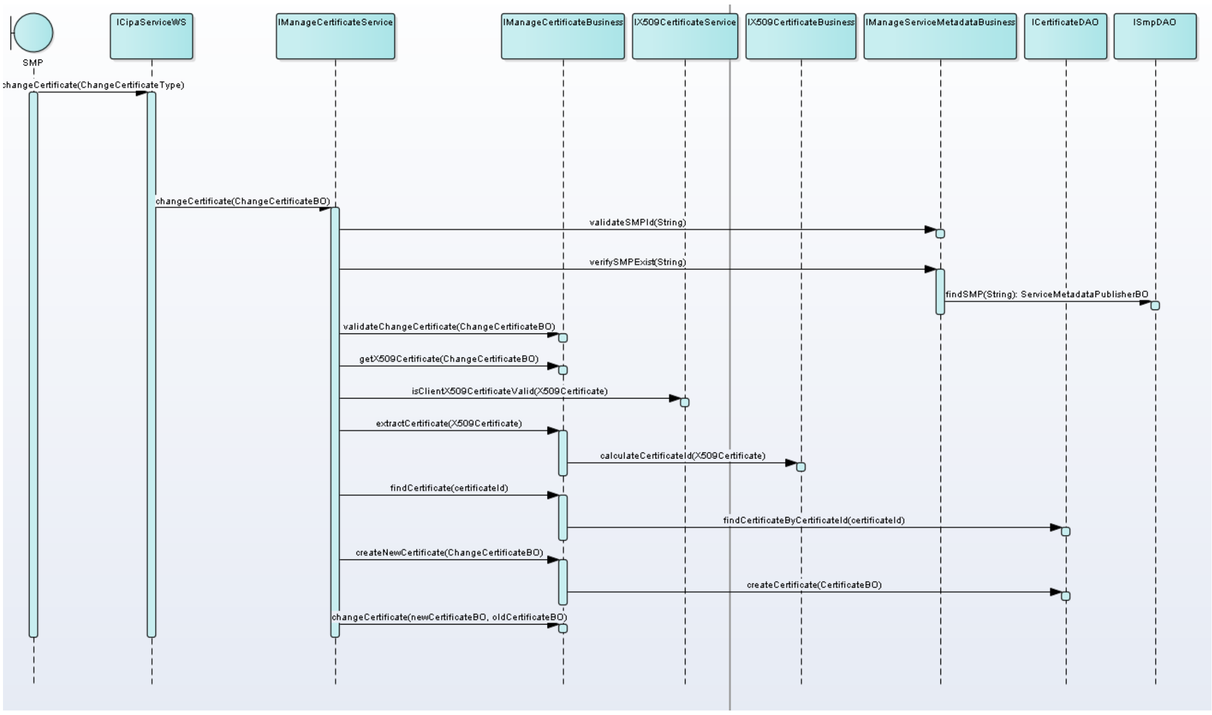

SMP
ICipaServiceWS
IManageCertificateService
IManageCertificateBusiness
IX509CertificateService
IX509CertificateBusiness
IManageServiceMetadataBusiness
ICertificateDAO
ISmpDAO
changeCertificate(ChangeCertificateType)
changeCertificate(ChangeCertificateBO)
validateSMPId(String)
verifySMPExist(String)
findSMP(String): ServiceMetadataPublisherBO
validateChangeCertificate(ChangeCertificateBO)
getX509Certificate(ChangeCertificateBO)
isClientX509CertificateValid(X509Certificate)
extractCertificate(X509Certificate)
calculateCertificateId(X509Certificate)
findCertificate(certificateId)
findCertificateByCertificateId(certificateId)
createNewCertificate(ChangeCertificateBO)
createCertificate(CertificateBO)
changeCertificate(newCertificateBO, oldCertificateBO)

Figure 16 - Sequence Diagram - BDMSLService ChangeCertificate()

3.3.3.4.3. Operation SetProperty()

3.3.3.4.3.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.3.2. Description

This operation allows the admin team to change BDMSL property in the database such as: passwords, DNS url, etc. The new property is taken into account when the cron task refreshes the properties simultaneously on all nodes in the cluster. Crontab properties are only refreshed with the restart of the BDMSL server.

- Input : Property name, Property value, optionally: Property description
- Output : Property data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: badRequestFault - returned if
    - The supplied request does not contain consistent data
    - Invalid propertyName
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.4. Operation GetProperty()

3.3.3.4.4.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.4.2. Description

This operation allows the admin team to retrieve BDMSL property from the database such as: DNS url, smtp configuration, etc.

- Input : Property name
- Output: Property data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid propertyName
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.5. Operation DeleteProperty()

3.3.3.4.5.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.5.2. Description

This operation allows the admin team to delete BDMSL non mandatory properties from database.

- Input : Property name,
- Output : Property data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid propertyName
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.6. Operation CreateSubDomain()

3.3.3.4.6.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.6.2. Description

This operation allows the admin team to create new BDMSL SubDomain. When creating subdomain the DNS types, SMP url scheme restriction, Participant regular expression must be defined.

- Input : SubDomain name, DNS Zone name, SubDomain DNS RecordTypes, Allowed SubDomain SMPs URL schema, Participant regular expression, expression for client certificate subject validation, list of allowed certificate policy OIDs, max count of participants which can be registered on the domain. Max. count of participants which can be registered by one SMP.
- Output : Subdomain data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid SubDomain data
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.7. Operation UpdateSubDomain()

3.3.3.4.7.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.7.2. Description

This operation allows the admin team to update the BDMSL SubDomain properties. In case of changing DNS Record Type and with DNS integration ON - the records are not updated automatically. Records must be updated manually using operations: AddDNSRecord, DeleteDNSRecord.

- Input: SubDomain name + Optional: SubDomain DNS RecordTypes, Allowed SubDomain SMPs URL schema, Participant regular expression, expression for client certificate subject validation, list of allowed certificate policy OIDs, max count of participants which can be registered on the domain. Max. count of participants which can be registered by one SMP.
- Output: Subdomain data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
    - Invalid SubDomain data
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.8. Operation GetSubDomain()

3.3.3.4.8.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.8.2. Description

This operation allows the admin team to read BDMSL SubDomain properties.

- Input : SubDomain name.
- Output : Subdomain data stored to BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
    - Invalid SubDomain data
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.9. Operation DeleteSubDomain()

3.3.3.4.9.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.9.2. Description

This operation allows the admin team to delete empty BDMSL SubDomain.

- Input : SubDomain name.
- Output : Deleted Subdomain data from BDMSL database
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate

- Fault: BadRequestFault - returned if
  - Subdomain has registered participants
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.10. Operation AddSubDomainCertificate()

3.3.3.4.10.1. Pre-requisites

- Valid Certificate
- The user has the role ROLE_ADMIN

3.3.3.4.10.2. Description

This operation allows the admin team to add new Domain certificate to BDMSL SubDomain. Certificate can be flagged as RootPKI certificate and/or as Admin certificate. Admin certificate can be only the certificate which is not flagged as RootPKI certificate.

- Input : SubDomain name, Certificate, Boolean value for: is certificate root PKI value and is certificate Admin Certificate.
- Output : Registered new Domain certificate data.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid Subdomain Certificate data
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.11. Operation UpdateSubDomainCertificate()

3.3.3.4.11.1. Pre-requisites

- The certificate is already added to database
- The user has the role ROLE_ADMIN

3.3.3.4.11.2. Description

This operation allows the admin team to update SubDomain certificate data. Admin can set or clear CRL distribution point, IsAdmin flag and SubDomain name.

- Input :Certificate Identifier + Optional: SubDomain name, Certificate CRL distribution URL, boolean value for is certificate Admin Certificate Boolean value.
- Output : Updated Domain certificate data.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid Subdomain Certificate data
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.12. Operation ListSubDomainCertificate()

3.3.3.4.12.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.12.2. Description

This operation allows the admin team to search for domain certificate by partial certificate DN and by the Subdomain.

- Input: Partial certificate identifier and/or domain name.
- Output: List of registered Domain certificate data which match the search criteria.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.13. Operation AddDNSRecord()

3.3.3.4.13.1. Pre-requisites

- BDMSL is integrated with DNS server.
- The user has the role ROLE_ADMIN

3.3.3.4.13.2. Description

This operation allows the admin team to add new record to DNS server for DNS RecordType: A, CNAME and NAPTR.

- Input: DNS Record name, DNS record Type, DNS record Value and service name in case of NAPTR record type.
- Output: Inserted DNS record.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - Invalid DNS name
  - Invalid DNS value
  - Invalid DNS record type
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason.

3.3.3.4.14. Operation DeleteDNSRecord()

3.3.3.4.14.1. Pre-requisites

- BDMSL is integrated with DNS server
- The user has the role ROLE_ADMIN

3.3.3.4.14.2. Description

This operation allows the admin team to remove records from DNS server. Varoius DNS records can have the same name. The ‘DeleteDNSRecord’ operation removes all DNS records with the same name. The deletion is done from the DNS server even if the DNS record does not exist in the database.

- Input: DNS Record name.
- Output: List of deleted DNS records.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
    - Invalid DNS record name
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason.

3.3.3.4.15. Operation AddTruststoreCertificate()

3.3.3.4.15.1. Pre-requisites

- Valid Certificate
- The user has the role ROLE_ADMIN

3.3.3.4.15.2. Description

This operation allows the admin team to add certificate to the truststore. Service is needed for adding a complete certificate chain to the truststore.

- Input : Base64 encoded X509Certificate + Optional: alias.
- Output : Inserted X509Certificate with truststore alias.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
    - Invalid Certificate data
    - Certificate already exists in truststore
    - Certificate with given alias already exists in truststore
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.16. Operation GetTruststoreCertificate()

3.3.3.4.16.1. Pre-requisites

- The certificate with given alias is already added to the truststore
- The user has the role ROLE_ADMIN

3.3.3.4.16.2. Description

This operation allows the admin team to retrieve the certificate from the truststore by the alias.

- Input : truststore alias.
- Output : X509Certificate data.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
    - alias is not given in parameter
- Fault NotFoundFault - returned if
    - alias is not present in truststore
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.17. Operation DeleteTruststoreCertificate()

3.3.3.4.17.1. Pre-requisites

- The certificate with given alias is in the truststore
- The user has the role ROLE_ADMIN

3.3.3.4.17.2. Description

This operation allows the admin team to remove the certificate from the truststore by the alias.

- Input : truststore alias.
- Output : deleted X509Certificate data.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if
  - alias is not given in parameter
- Fault NotFoundFault - returned if
  - alias is not present in truststore
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.18. Operation ListTruststoreCertificateAliases()

3.3.3.4.18.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.18.2. Description

This operation allows the admin team to retrieve all aliases for the certificates registered in the truststore.

- Input: Optional: partial certificate alias.
- Output: List of registered aliases, which match the search criteria or all aliases if input value is empty.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.19. Operation GenerateInconsistencyReport ()

3.3.3.4.19.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.19.2. Description

This operation allows the admin team to trigger the generation of the inconsistency report by demand. The report is generated asynchronously, and in large DNS zones/tables can take several 10 minutes.

- Input: email for receiving the report.
- Output: email with the report.

- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if email is invalid
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.3.3.4.20. Operation GenerateReport ()

3.3.3.4.20.1. Pre-requisites

- The user has the role ROLE_ADMIN

3.3.3.4.20.2. Description

This operation allows the admin team to trigger generation of any of the reports supported by the BDMSL.

- Input: email for receiving the report. Report type
- Output: email with the report.
- Fault: UnauthorizedFault : Returned if the certificate provided is not a ADMIN certificate
- Fault: BadRequestFault - returned if email or report type are invalid
- Fault: InternalFaultError : Returned if the BDMSL service is unable to process the request for any reason

3.4. Business layer

The business layer manipulates only business objects and defines the business rules.

Business objects are POJO and implement the Singleton pattern. They are defined in the common package described in section 3.6.3 Business Objects (BO).

3.4.1. Naming convention

- Package: eu.europa.ec.bdmsl.business
- Interface: eu.europa.ec.bdmsl.business.I<InterfaceName>Business
- Implementation package: eu.europa.ec.bdmsl.business.impl
- Implementation: eu.europa.ec.bdmsl.business.impl.<InterfaceName>BusinessImpl

3.4.2. Dependencies

In this layer, only the following calls are allowed:

- Calls to technical components
- Calls to the persistence layer
- Calls to the common package

3.4.3. Frameworks

This layer handles most of the business logic processing. Therefore, there is no technical aspect. The only framework used is Spring for the dependency injection.

3.4.4. Development of the business layer

3.4.4.1. Interface

```
public interface IManageServiceMetadataBusiness {

  /**
   * Retrieves the Service Metadata Publisher record for the service metadata.
   * @param serviceMetadataPublisherID the unique ID
   *          of the Service Metadata Publisher for which the record is required
   * @return ServiceMetadataPublisherBO the service metadata publisher record.
   * @throws TechnicalException Technical exception.
   * @throws BusinessException Business exception.
   */
  ServiceMetadataPublisherBO read(String serviceMetadataPublisherID);
}
```

3.4.4.2. Implementation classes

The implementation classes extend the parent-class «AbstractBusinessImpl» and implement their dedicated interface (here «IManageServiceMetadataBusiness»). The AbstractBusinessImpl class only contains the logging service but may be completed with new services if new common requirements are identified for *BusinessImpl classes in future versions.

```
public  class  ManageServiceMetadataBusinessImpl  extends  AbstractBusinessImpl
implements IManageServiceMetadataBusiness {

  private IManageServiceMetadataDAO manageServiceMetadataDAO;

  /*
   * (non-Javadoc)
   *
   * @see eu.europa.ec.bdmsl.service.IManageServiceMetadataBusiness#read (String)
   */
  @Override
  public ServiceMetadataPublisherBO read(String serviceMetadataPublisherID)
    throws TechnicalException, BusinessException {
    ServiceMetadataPublisherBO smpBO =
manageServiceMetadataServiceBusiness.read(serviceMetadataPublisherID);
    return smpBO;
  }
  ...
}
```

3.5. Data Access layer

This layer access to the data persisted in the database. The objects of this layer are POJO that implement the Singleton and DAO pattern.

3.5.1. Naming convention

- Package: eu.europa.ec.bdmsl.dao
- Interface: eu.europa.ec.bdmsl.dao.I<InterfaceName>DAO
- Implementation package: eu.europa.ec.bdmsl.dao.impl
- Implementation: eu.europa.ec.bdmsl.dao.impl.<InterfaceName>DAOImpl
- Package Entity Object: eu.europa.ec.bdmsl.dao.entity
- Entity object: eu.europa.ec.bdmsl.dao.entity.<ObjectName>Entity

3.5.2. Dependencies

In this layer, only the following calls are allowed:

- Calls to technical components
- Calls to the common package

3.5.3. Frameworks

This layer is the only one to use the JPA framework because it is the only one that actually accesses to the database.

The configuration is managed by Spring.

3.5.4. Development of the data access layer

3.5.4.1. Interface

```
public interface ISmpDAO {

  /**
   * Retrieves the Service Metadata Publisher record for the service metadata.
   * @param serviceMetadataPublisherID the unique ID
   *           of the Service Metadata Publisher for which the record is required
   * @return ServiceMetadataPublisherBO the service metadata publisher
   */
  ServiceMetadataPublisherBO  findSMP(String  serviceMetadataPublisherID)  throws
TechnicalException;

}
```

3.5.4.2. Implementation classes

The implementation classes extend the parent-class «AbstractDAOImpl» and implement their dedicated interface (here «IManageServiceMetadataDAO»).

```
public class ManageServiceMetadataDAOImpl extends AbstractDAOImpl implements
ISmpDAO {

  /**
   * @see eu.europa.ec.bdmsl.dao.ISmpDAO#findSMP(String)
   */
  @Override
  public ServiceMetadataPublisherBO findSMP(String serviceMetadataPublisherID)
    throws TechnicalException {
            ServiceMetadataPublisherBO resultBO = null;
        SmpEntity resultSmpEntity = getEntityManager().find(SmpEntity.class, id);
        if (resultSmpEntity != null) {
            resultBO = mapperFactory.getMapperFacade().map(resultSmpEntity,
ServiceMetadataPublisherBO.class);
        } else {
            loggingService.debug("No SMP found for id " + id);
        }

        return resultBO;
   }
...
}
```

3.5.4.3. Mapping BO / entity

The data access layer internally uses JPA entities to perform the Object/Relational mapping with the database. However, the methods exposed in the interfaces only expose Business Objects because the Business objects are the only ones that can be used between the layers. For more information on the mapping BO/Entities, see the chapter ***8 Object mapping.***

3.6. Common package

This package is particular because it can be called without restriction by all the layers of the application: it is transversal.

This common package provides:

- Business and technical Exceptions.
- Business objects (BOs) to be used in every layer
- Constants, error codes, utility classes
- Enums

3.6.1. Naming convention

- Package: eu.europa.ec.bdmsl.common
- Package Business Object: eu.europa.ec.bdmsl.common.bo
- Business Object: eu.europa.ec.bdmsl.common.<ObjectName>BO

3.6.2. Dependencies

In this layer, only the following calls are allowed:

- Calls to technical components

3.6.3. Business Objects (BO)

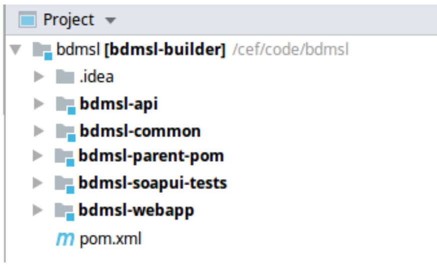
Business objects (BO) are developed in the common package because they are transversal to all layers and are used in the service, business and persistence layer. They are POJO with no dependency to any framework or database. They can walk through the layers. We use BO because they are linked to the domain, and hide the implementation choices made for façade and for the persistence. Thus, they are not directly linked to any database model, or any web service interface.

Each BO extends the abstract class `AbstractBusinessObject` provided by the common library. This class implements `java.io.Serializable` and overrides `equals`, `hashCode` and `toString` as abstract methods.

Each BO must define a `serialVersionUID` and implement the 3 previous methods.

4. Software architecture

There are 5 different maven projects:



```
Project
▼ bdmsl [bdmsl-builder] /cef/code/bdmsl
    ▶ .idea
    ▶ bdmsl-api
    ▶ bdmsl-common
    ▶ bdmsl-parent-pom
    ▶ bdmsl-soapui-tests
    ▶ bdmsl-webapp
      pom.xml
```

Figure 16 - Project structure

In this chapter, we describe the content and the role of each project.

4.1. Library "bdmsl-api"

Project name : bdmsl-api

It is a SOAP web service client (stub) that is generated from the WSDLs of the main project. The api is a Maven project and the output is packaged as a `jar` file.

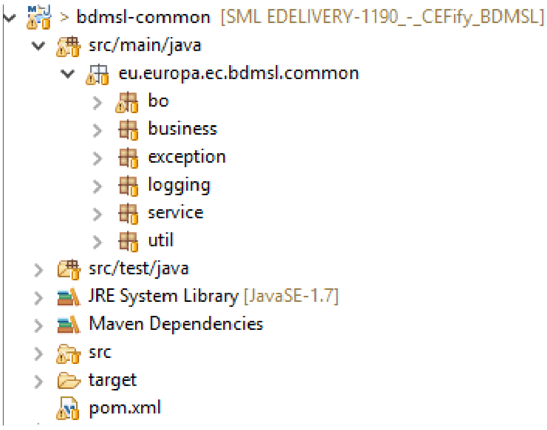
The WSDL files contain all the methods that are exposed, the objects and the exceptions.

The projects that call the web services of the eDelivery BDMSL application can use this web service client.

4.2. Library "bdmsl-common"

Project name : bdmsl-common

Packages:



```
> bdmsl-common [SML EDELIVERY-1190_-_CEFify_BDMSL]
    src/main/java
        eu.europa.ec.bdmsl.common
            > bo
            > business
            > exception
            > logging
            > service
            > util
  > src/test/java
  > JRE System Library [JavaSE-1.7]
  > Maven Dependencies
  > src
  > target
    pom.xml
```

Figure 17 - Packages of the bdmsl-common project

This library is used by all the modules of the eDelivery BDMSL solution. It provides services like:

- Cryptography
- Constants
- Configuration manager
- Utils (dates, encoding, etc.)
- Logging
- Abstract/parent classes common to all eDelivery BDMSL modules

4.3. bdmsl-webapp

Project name : bdmsl-webapp

This is a Maven project that contains all the services, business logic and persistence code of the application. It produces a `war` file that can be deployed in the supported application servers and servlet containers.

4.4. Parent pom

Project name: bdmsl-parent-pom

It's the parent pom of all the Maven module. It contains the version for the dependencies, default configuration of plugins, etc.

4.5. Maven configuration

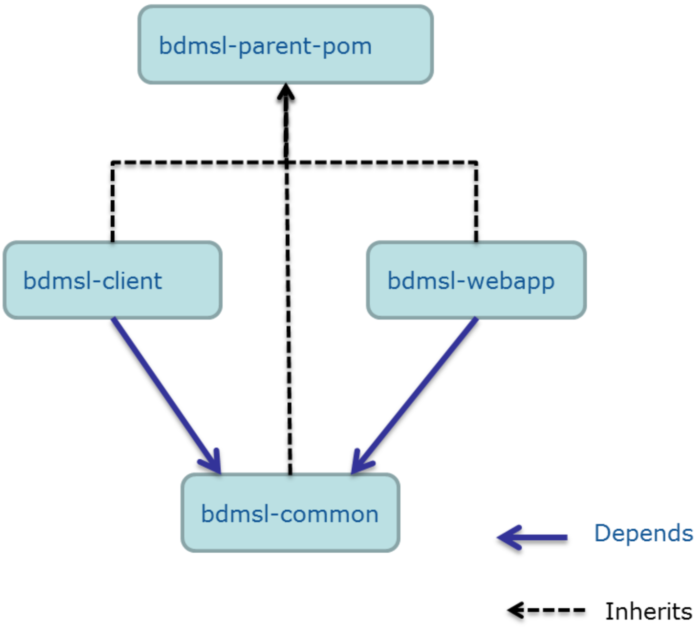


Figure 18 - Dependency tree of the maven projects

5. Logging

5.1. Implementation

The logs use the Log4j framework. The `bdmsl-common` library provides the logging manager `ILoggingService` and its main implementation class `LoggingServiceImpl`. This logging manager must be used for all the logs within the eDelivery BDMSL application.

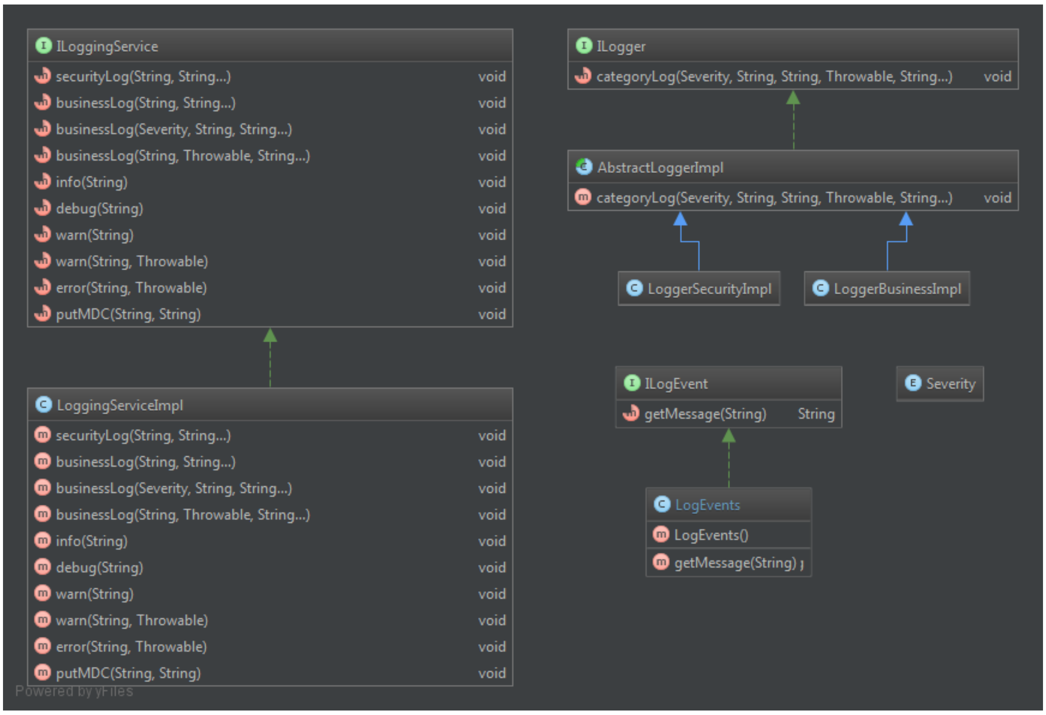

ILoggingService
securityLog(String, String...) void
businessLog(String, String...) void
businessLog(Severity, String, String...) void
businessLog(String, Throwable, String...) void
info(String) void
debug(String) void
warn(String) void
warn(String, Throwable) void
error(String, Throwable) void
putMDC(String, String) void
LoggingServiceImpl
securityLog(String, String...) void
businessLog(String, String...) void
businessLog(Severity, String, String...) void
businessLog(String, Throwable, String...) void
info(String) void
debug(String) void
warn(String) void
warn(String, Throwable) void
error(String, Throwable) void
putMDC(String, String) void
ILogger
categoryLog(Severity, String, String, Throwable, String...) void
AbstractLoggerImpl
categoryLog(Severity, String, String, Throwable, String...) void
LoggerSecurityImpl
LoggerBusinessImpl
ILogEvent
getMessage(String) String
Severity
LogEvents
LogEvents()
getMessage(String)
Powered by yFiles

Figure 19 - Logging class diagram

There are 3 types of logs: security logs, business logs and miscellaneous logs. Each category of log has its own appender defined in the `log4j.xml` file. By default, each category will log in a separate file:

- `bdmsl-security.log` : This log file contains all the security related information. For example, you can find information about the clients who connect to the application.
- `bdmsl-business.log`: This log file contains all the business related information. For example, when a participat is created, when a SMP is deleted, etc.
- `bdmsl.log` : This log file contains both the security and business logs plus miscellaneous logs like debug information, logs from one of the framework used by the application, etc.

The security and business logs require a code that is defined in the implementation of the `ILogEvent` interface. In the eDelivery BDMSL application, all the security and business messages are defined in the `LogEvents` class.

The pattern of the logs is defined in the `log4j.xml` file. The default pattern is:

```
%d{ISO8601}{Europe/Brussels} [%X{user}] [%X{requestId}] %-5p %c{1}:%L - %m%n
```

- `user`: The client authenticated by its certificate.
- `requestId`: the UUID of the request (provided by the application server)

The values for the `user` and `requestId` properties can be set by calling the method `ILoggingService.putMDC(String key, String value)`.

5.2. Log event codes

Category	Log event code	Description
SECURITY	SEC-001	The host %s attempted to access %s without any certificate
SECURITY	SEC-002	The host %s has been granted access to %s with roles %s
SECURITY	SEC-003	The host %s has been refused access to %s
SECURITY	SEC-004	The certificate is revoked : %s
SECURITY	SEC-005	The root certificate of the client certificate is unknown in the database. It means that the certificate is accepted at transport level (SSL) but refused at application level. %s
SECURITY	SEC-006	Certificate is not valid at the current date %s. Certificate valid from %s to %s
SECURITY	SEC-007	Certificate is not yet valid at the current date %s. Certificate valid from %s to %s
BUSINESS	BUS-001	Technical error while authentication process
BUSINESS	BUS-002	Error while configuring the application.
BUSINESS	BUS-003	The SMP was successfully created: %s.
BUSINESS	BUS-004	The SMP couldn't be created: %s.
BUSINESS	BUS-005	The following SMP was read: %s.
BUSINESS	BUS-006	The SMP couldn't be read: %s.
BUSINESS	BUS-007	The SMP was successfully deleted: %s.
BUSINESS	BUS-008	The SMP couldn't be deleted: %s.
BUSINESS	BUS-009	The SMP was successfully updated: %s.
BUSINESS	BUS-010	The SMP couldn't be updated: %s.
BUSINESS	BUS-011	The participant was successfully created: %s.
BUSINESS	BUS-012	The participant couldn't be created: %s.
BUSINESS	BUS-013	The list of participant couldn't be created: %s.
BUSINESS	BUS-014	The list of participants couldn't be created: %s.
BUSINESS	BUS-015	The participant was successfully deleted: %s.
BUSINESS	BUS-016	The participant couldn't be deleted: %s.
BUSINESS	BUS-017	The list of participant couldn't be deleted: %s.
BUSINESS	BUS-018	The list of participants couldn't be deleted: %s.
BUSINESS	BUS-019	The participants of SMP %s have been successfully listed.
BUSINESS	BUS-020	The participants of SMP %s couldn't be listed.
BUSINESS	BUS-021	The prepare to migrate service was successfully called for participant: %s.
BUSINESS	BUS-022	The prepare to migrate service failed for participant: %s.
BUSINESS	BUS-023	The call to migrate service was successfully called for participant: %s.
BUSINESS	BUS-024	The call to migrate service failed for participant: %s.
BUSINESS	BUS-025	The call to the list service succeeded
BUSINESS	BUS-026	The call to the list service failed
BUSINESS	BUS-027	The new certificate was successfully planned for change for current certificate: %s
BUSINESS	BUS-028	The certificate change failed for current certificate: %s
BUSINESS	BUS-029	The following CNAME record has been added to the DNS for the participant %s : %s

BUSINESS	BUS-030	The following NAPTR record has been added to the DNS for the participant %s : %s
BUSINESS	BUS-031	The following CNAME record has been added to the DNS for the SMP %s : %s
BUSINESS	BUS-032	The following A record has been added to the DNS for the SMP %s : %s
BUSINESS	BUS-033	The CertificateChangeJob ran successfully. %s certificates have been migrated
BUSINESS	BUS-034	The CertificateChangeJob failed.
BUSINESS	BUS-035	The ChangeCertificate service has been executed successfully
BUSINESS	BUS-036	The ChangeCertificate service has failed

Table 1 - Log event codes

6. CACHING

In order to enhance performance, in-memory caches are used in the application. They rely on the `ehcache` implementation. To put objects in a cache, we use annotations:

```
@Override
@Cacheable(value = "crlByUrl", key = "#crlDistributionPointURL")
public void verifyCertificateCRLs(String serial, String crlDistributionPointURL){
  [...]
}
```

The `@Cacheable` annotation triggers cache population. In the previous example, the name of the cache is `crlByUrl`. The `key` attribute is one of the parameters of the method: `crlDistributionPointURL`. The next time this method is called, if the cache is already populated with a value for the given key, then the method won't actually be called and the result will be returned from the cache.

Sometimes, it is useful to clear the caches. This can be done by calling the method `IBDMSLService.clearCache()`.

7. Exception handling

7.1. Exception types

When exceptions are thrown in the business, persistence and service layers, they are transformed into technical or business exceptions to ensure to the client of the service that all the possible exceptions are declared in the service signature.

All the methods of the exposed interfaces in the persistence, business and service layer can only throw two kinds of exceptions:

- `TechnicalException` : Technical exceptions happen when a technical component of a business process acts in an unexpected way. Examples of technical exceptions are: IO exception, timeout, bad configuration, etc.
- `BusinessException` : Business Exceptions are exceptions that are designed and managed in the specification of a business process. In other words, Business Exceptions are exceptions which happen at the process or workflow level, and are not related to the technical components.

7.2. SOAP Faults

Because of the design of the WSDL in the SML specification, it is not possible to use an interceptor to transform the exceptions into SOAP fault. Thus, it is the `AbstractWSImpl` class which handles exceptions and convert any type of exception into appropriate SOAP faults. In the eDelivery BDMSL, there are 4 types of SOAP faults, all mapped to `TechnicalException`:

- NotFoundFault
- UnauthorizedFault
- BadRequestFault
- InternalErrorFault

A typical SOAP fault example would be:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <soap:Fault>
         <faultcode>soap:Server</faultcode>
         <faultstring>5378C6571DE2DD3FD026704338FF678B</faultstring>
         <detail>
            <NotFoundFault
xmlns:ns2="http://busdox.org/transport/identifiers/1.0/"
xmlns="http://busdox.org/serviceMetadata/locator/1.0/">
               <FaultMessage>[ERR-100] The SMP 'testSMLUpdate' doesn't
exist.</FaultMessage>
            </NotFoundFault>
         </detail>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

In the previous SOAP fault, the `faultstring` contains the request unique identifier provided by the application server. This request unique identifier is traced in the logs, in order to easily find the logs associated to an exception:

```
2015-07-24 10:32:08,562 [unsecure-http-client] [5378C6571DE2DD3FD026704338FF678B]
ERROR LoggingServiceImpl:83 - [ERR-100] The SMP 'testSMLUpdate' doesn't exist.
```

The error codes are all listed in the `IErrorCodes` interface (see table in the following paragraph).

7.3. Error codes

Error code	Description	Exception type
100	SMP not found error	TechnicalException
101	Unauthorized error	TechnicalException
102	Certificate authentication issue	TechnicalException
103	The root alias is not found in the list of trusted issuers in the database	TechnicalException
104	The certificate is revoked	TechnicalException
105	Generic technical error	TechnicalException
106	Bad request error	TechnicalException
107	DNS communication problem	TechnicalException
108	Problem with the SIG0 signature	TechnicalException
109	Bad configuration	TechnicalException
110	Participant not found error	TechnicalException
111	Migration data not found	TechnicalException
112	Duplicate participant error	TechnicalException
113	Error when deleting a SMP	TechnicalException
114	The deletion failed because a migration is planned for the given participant or SMP	TechnicalException
115	The certificate couldn't be found	TechnicalException

Table 2 - Error Codes

8. Object mapping

There are 3 types of objects used in the application:

- JAXB objects : Generated objects from the WSDL
- Business objects (BO) : POJO used in the business logic in the service, business and persistence layers
- JPA entities : Persistence domain objects

2 types of mapping are required:

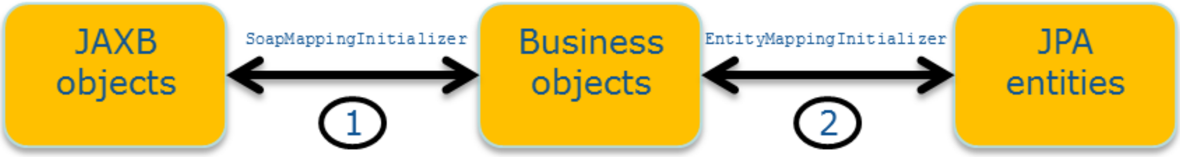


Figure 20 - Object mappings

The first type of mapping converts JAXB objects to BO and vice-versa. The implementation class is `SoapMappingInitializer`.

The second type of mapping converts JPA entities to BO and vice-versa. The implementation class is `EntityMappingInitializer`.

In order to avoid hand coding value object assemblers to copy data from one object type to another, we use a generic framework named Orika. Orika is a Java Bean mapping framework that recursively copies data from one object to another.

An example of mapping would be:

```
@Component
public class SoapMappingInitializer {

    @Autowired
    private MapperFactory mapperFactory;

    @PostConstruct
    public void init() {
           [...]
        mapperFactory.classMap(PageRequestType.class, PageRequestBO.class)
                .field("serviceMetadataPublisherID", "smpId")
                .byDefault()
                .register();
           [...]
    }
}
```

In the previous mapping, we map the field `serviceMetadataPublisherID` of the class `PageRequestType` to the field `smpId` of the class `PageRequestBO`. The other fields have the same name so they are automatically mapped thanks to the `byDefault()` method. This mapping is bidirectional.

To map an object, the singleton instance of the MapperFactory object can be used. For instance, in the Façade layer (ws) :

```
[...]

public class BDMSLServiceWSImpl extends AbstractWSImpl implements IBDMSLServiceWS
{
    [...]

    @Autowired
    private MapperFactory mapperFactory;
    [...]

    @Override
    @WebMethod([...])
    public void create(@WebParam ParticipantType participantType) {
     [...]
     // Convert the ParticipantType JAXB object into a ParticipantBO object
     ParticipantBO participantBO =
mapperFactory.getMapperFacade().map(participantType, ParticipantBO.class);
    [...]
    }
[...]
}
```

9. Database management

9.1. Auditing

In order to automatically audit the changes in the database, all the DAOs must extend the `AbstractDAOImpl` class and use its `persist()` and `merge()` methods. This way, the date of the changes of any business data is automatically logged.

For each table containing business data, these 2 following columns are present:

- `created_on`: date of creation of the row
- `last_updated_on`: date of the last update of the row

Data changes are also logged with hibernate envers. Each table has an audit table with the suffix '_aud'.

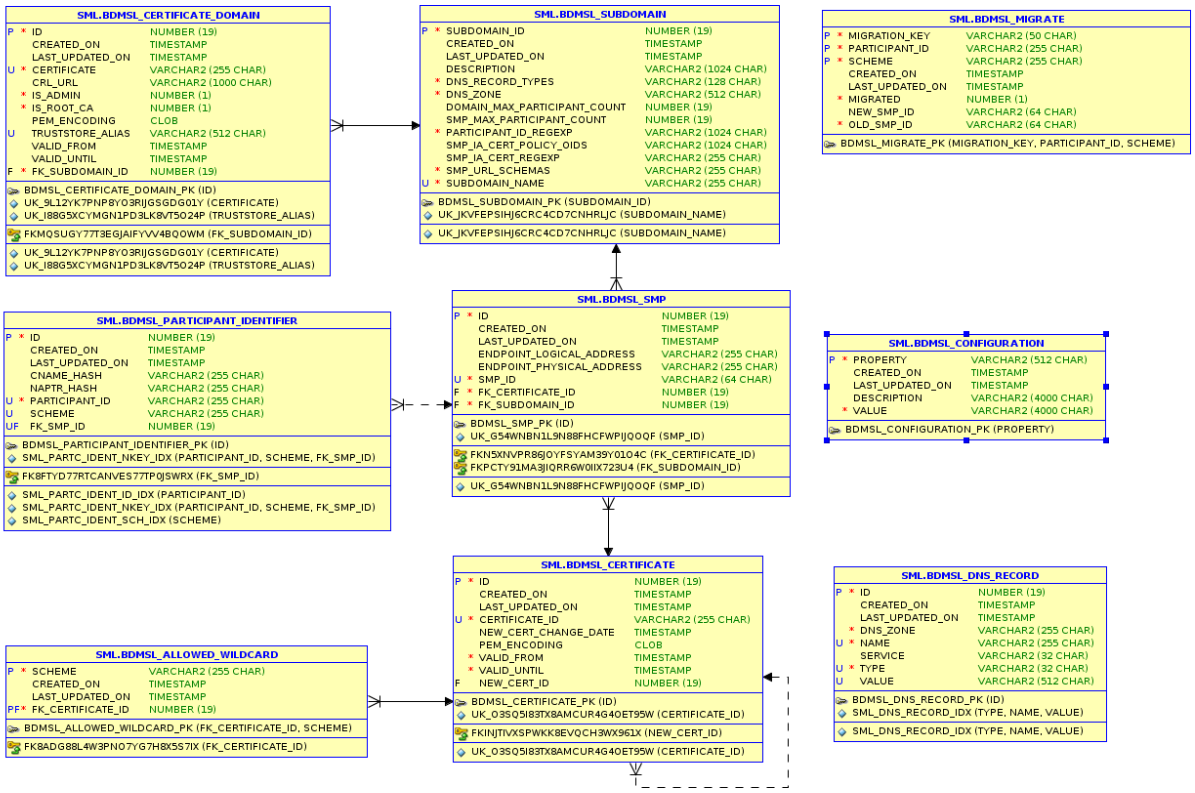
9.2. Data model

Java entity classes are located in eu.europa.ec.bdmsl.dao.entity package. Entity annotations define the Model with the use of JPA2 annotations. Database ddl scripts are generated automatically during the build time by the maven plugin in the bdmsl-webapp subproject:

```
<plugin>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <id>generate-ddl</id>
      <phase>process-classes</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <target>
<!-- ANT Task definition
Class generates ddl scripts
1. Parameter: comma separated hibernate database dialects
2. script version
3. export scripts.-->
          <java       classname="eu.europa.ec.bdmsl.dao.utils.SMLSchemaGenerator"
fork="true"
failonerror="true">
            <arg
value="org.hibernate.dialect.Oracle10gDialect,org.hibernate.dialect.MySQL5InnoDBD
ialect"/>
            <arg value="${project.version}"/>
            <arg value="${project.basedir}/src/main/sml-setup/database-scripts"/>
<!-- reference to the passed-in classpath reference -->
            <classpath refid="maven.compile.classpath"/>
          </java>
        </target>
      </configuration>
    </execution>
  </executions>
</plugin>
```

By default, the ddl scripts for Oracle10gDialect and MySQL5InnoDBDialect database are generated.

9.2.1. Overview



SML.BDMSL_CERTIFICATE_DOMAIN

P	*	ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
U	*	CERTIFICATE	VARCHAR2 (255 CHAR)
		CRL_URL	VARCHAR2 (1000 CHAR)
	*	IS_ADMIN	NUMBER (1)
	*	IS_ROOT_CA	NUMBER (1)
		PEM_ENCODING	CLOB
U		TRUSTSTORE_ALIAS	VARCHAR2 (512 CHAR)
		VALID_FROM	TIMESTAMP
		VALID_UNTIL	TIMESTAMP
F	*	FK_SUBDOMAIN_ID	NUMBER (19)

BDMSL_CERTIFICATE_DOMAIN_PK (ID)
UK_9L12YK7PNP8YO3RIJGSGDG01Y (CERTIFICATE)
UK_I88G5XCYMGN1PD3LK8VT5O24P (TRUSTSTORE_ALIAS)
FKMQSUGY77T3EGJAIFYVV4BQOWM (FK_SUBDOMAIN_ID)
UK_9L12YK7PNP8YO3RIJGSGDG01Y (CERTIFICATE)
UK_I88G5XCYMGN1PD3LK8VT5O24P (TRUSTSTORE_ALIAS)

SML.BDMSL_SUBDOMAIN

P	*	SUBDOMAIN_ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
		DESCRIPTION	VARCHAR2 (1024 CHAR)
	*	DNS_RECORD_TYPES	VARCHAR2 (128 CHAR)
	*	DNS_ZONE	VARCHAR2 (512 CHAR)
		DOMAIN_MAX_PARTICIPANT_COUNT	NUMBER (19)
		SMP_MAX_PARTICIPANT_COUNT	NUMBER (19)
	*	PARTICIPANT_ID_REGEXP	VARCHAR2 (1024 CHAR)
		SMP_IA_CERT_POLICY_OIDS	VARCHAR2 (1024 CHAR)
		SMP_IA_CERT_REGEXP	VARCHAR2 (255 CHAR)
	*	SMP_URL_SCHEMAS	VARCHAR2 (255 CHAR)
U	*	SUBDOMAIN_NAME	VARCHAR2 (255 CHAR)

BDMSL_SUBDOMAIN_PK (SUBDOMAIN_ID)
UK_JKVFEPSIHJ6CRC4CD7CNHRLJC (SUBDOMAIN_NAME)
UK_JKVFEPSIHJ6CRC4CD7CNHRLJC (SUBDOMAIN_NAME)

SML.BDMSL_MIGRATE

P	*	MIGRATION_KEY	VARCHAR2 (50 CHAR)
P	*	PARTICIPANT_ID	VARCHAR2 (255 CHAR)
P	*	SCHEME	VARCHAR2 (255 CHAR)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
	*	MIGRATED	NUMBER (1)
		NEW_SMP_ID	VARCHAR2 (64 CHAR)
	*	OLD_SMP_ID	VARCHAR2 (64 CHAR)

BDMSL_MIGRATE_PK (MIGRATION_KEY, PARTICIPANT_ID, SCHEME)

SML.BDMSL_PARTICIPANT_IDENTIFIER

P	*	ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
		CNAME_HASH	VARCHAR2 (255 CHAR)
		NAPTR_HASH	VARCHAR2 (255 CHAR)
U	*	PARTICIPANT_ID	VARCHAR2 (255 CHAR)
U		SCHEME	VARCHAR2 (255 CHAR)
UF		FK_SMP_ID	NUMBER (19)

BDMSL_PARTICIPANT_IDENTIFIER_PK (ID)
SML_PARTC_IDENT_NKEY_IDX (PARTICIPANT_ID, SCHEME, FK_SMP_ID)
FK8FTYD77RTCANVES77TP0JSWRX (FK_SMP_ID)
SML_PARTC_IDENT_ID_IDX (PARTICIPANT_ID)
SML_PARTC_IDENT_NKEY_IDX (PARTICIPANT_ID, SCHEME, FK_SMP_ID)
SML_PARTC_IDENT_SCH_IDX (SCHEME)

SML.BDMSL_SMP

P	*	ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
		ENDPOINT_LOGICAL_ADDRESS	VARCHAR2 (255 CHAR)
		ENDPOINT_PHYSICAL_ADDRESS	VARCHAR2 (255 CHAR)
U	*	SMP_ID	VARCHAR2 (64 CHAR)
F	*	FK_CERTIFICATE_ID	NUMBER (19)
F	*	FK_SUBDOMAIN_ID	NUMBER (19)

BDMSL_SMP_PK (ID)
UK_G54WNBN1L9N88FHCFWPIJQOQF (SMP_ID)
FKN5XNVPR86JOYFSYAM39Y01O4C (FK_CERTIFICATE_ID)
FKPCTY91MA3JIQRR6W0IIX723U4 (FK_SUBDOMAIN_ID)
UK_G54WNBN1L9N88FHCFWPIJQOQF (SMP_ID)

SML.BDMSL_CONFIGURATION

P	*	PROPERTY	VARCHAR2 (512 CHAR)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
		DESCRIPTION	VARCHAR2 (4000 CHAR)
	*	VALUE	VARCHAR2 (4000 CHAR)

BDMSL_CONFIGURATION_PK (PROPERTY)

SML.BDMSL_CERTIFICATE

P	*	ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
U	*	CERTIFICATE_ID	VARCHAR2 (255 CHAR)
		NEW_CERT_CHANGE_DATE	TIMESTAMP
		PEM_ENCODING	CLOB
	*	VALID_FROM	TIMESTAMP
	*	VALID_UNTIL	TIMESTAMP
F		NEW_CERT_ID	NUMBER (19)

BDMSL_CERTIFICATE_PK (ID)
UK_O3SQ5I83TX8AMCUR4G4OET95W (CERTIFICATE_ID)
FKINJTIVXSPWKK8EVQCH3WX961X (NEW_CERT_ID)
UK_O3SQ5I83TX8AMCUR4G4OET95W (CERTIFICATE_ID)

SML.BDMSL_DNS_RECORD

P	*	ID	NUMBER (19)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
	*	DNS_ZONE	VARCHAR2 (255 CHAR)
U	*	NAME	VARCHAR2 (255 CHAR)
		SERVICE	VARCHAR2 (32 CHAR)
U	*	TYPE	VARCHAR2 (32 CHAR)
U		VALUE	VARCHAR2 (512 CHAR)

BDMSL_DNS_RECORD_PK (ID)
SML_DNS_RECORD_IDX (TYPE, NAME, VALUE)
SML_DNS_RECORD_IDX (TYPE, NAME, VALUE)

SML.BDMSL_ALLOWED_WILDCARD

P	*	SCHEME	VARCHAR2 (255 CHAR)
		CREATED_ON	TIMESTAMP
		LAST_UPDATED_ON	TIMESTAMP
PF	*	FK_CERTIFICATE_ID	NUMBER (19)

BDMSL_ALLOWED_WILDCARD_PK (FK_CERTIFICATE_ID, SCHEME)
FK8ADG88L4W3PNO7YG7H8X5S7IX (FK_CERTIFICATE_ID)

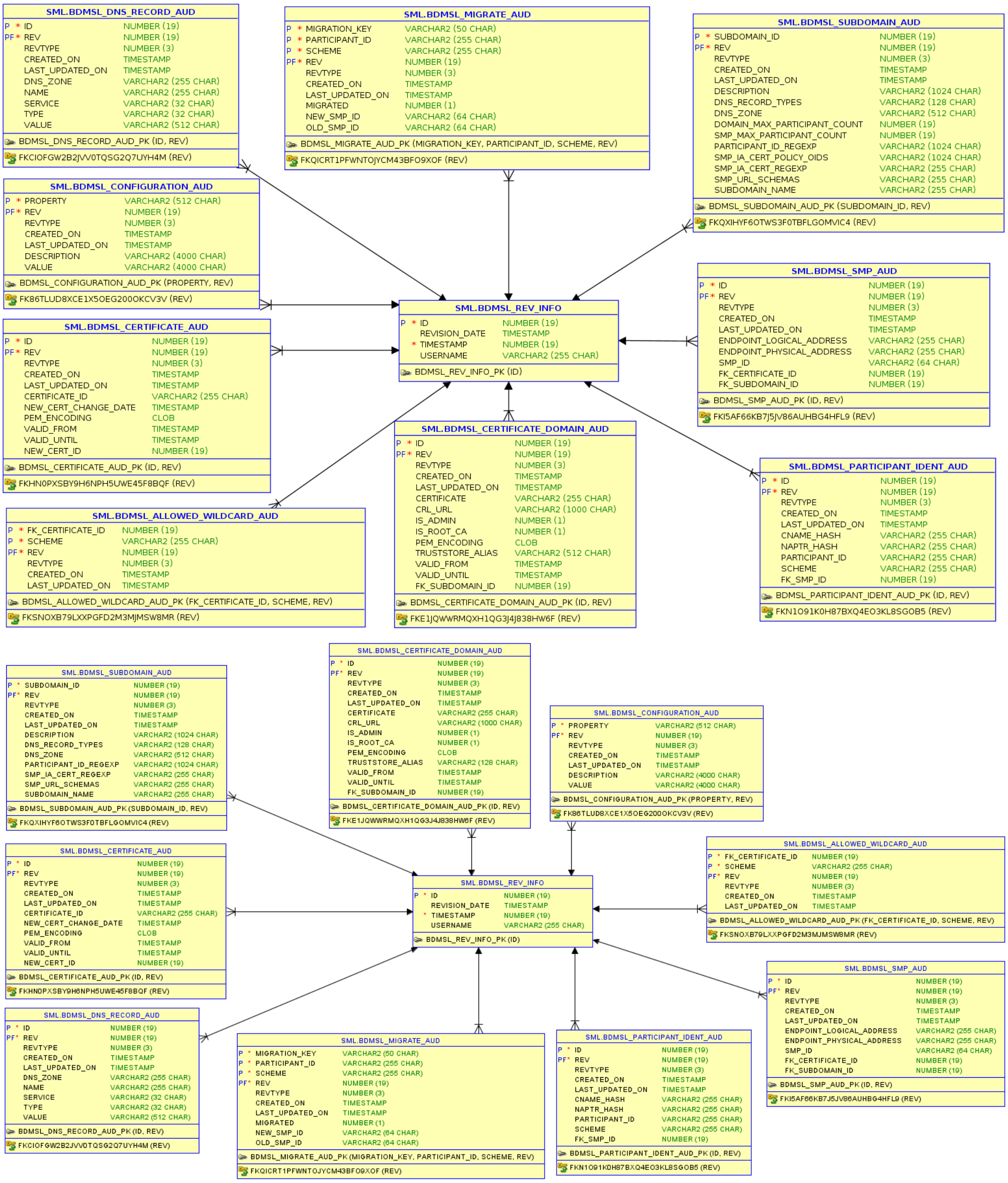

SML.BDMSL_DNS_RECORD_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DNS_ZONE VARCHAR2 (255 CHAR)
NAME VARCHAR2 (255 CHAR)
SERVICE VARCHAR2 (32 CHAR)
TYPE VARCHAR2 (32 CHAR)
VALUE VARCHAR2 (512 CHAR)
BDMSL_DNS_RECORD_AUD_PK (ID, REV)
FKCIOFGW2B2JVV0TQSG2Q7UYH4M (REV)
SML.BDMSL_MIGRATE_AUD
P * MIGRATION_KEY VARCHAR2 (50 CHAR)
P * PARTICIPANT_ID VARCHAR2 (255 CHAR)
P * SCHEME VARCHAR2 (255 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
MIGRATED NUMBER (1)
NEW_SMP_ID VARCHAR2 (64 CHAR)
OLD_SMP_ID VARCHAR2 (64 CHAR)
BDMSL_MIGRATE_AUD_PK (MIGRATION_KEY, PARTICIPANT_ID, SCHEME, REV)
FKQICRT1PFWNTOJYCM43BFO9XOF (REV)
SML.BDMSL_SUBDOMAIN_AUD
P * SUBDOMAIN_ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DESCRIPTION VARCHAR2 (1024 CHAR)
DNS_RECORD_TYPES VARCHAR2 (128 CHAR)
DNS_ZONE VARCHAR2 (512 CHAR)
DOMAIN_MAX_PARTICIPANT_COUNT NUMBER (19)
SMP_MAX_PARTICIPANT_COUNT NUMBER (19)
PARTICIPANT_ID_REGEXP VARCHAR2 (1024 CHAR)
SMP_IA_CERT_POLICY_OIDS VARCHAR2 (1024 CHAR)
SMP_IA_CERT_REGEXP VARCHAR2 (255 CHAR)
SMP_URL_SCHEMAS VARCHAR2 (255 CHAR)
SUBDOMAIN_NAME VARCHAR2 (255 CHAR)
BDMSL_SUBDOMAIN_AUD_PK (SUBDOMAIN_ID, REV)
FKQXIHYF6OTWS3F0TBFLGOMVIC4 (REV)
SML.BDMSL_CONFIGURATION_AUD
P * PROPERTY VARCHAR2 (512 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DESCRIPTION VARCHAR2 (4000 CHAR)
VALUE VARCHAR2 (4000 CHAR)
BDMSL_CONFIGURATION_AUD_PK (PROPERTY, REV)
FK86TLUD8XCE1X5OEG200OKCV3V (REV)
SML.BDMSL_SMP_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
ENDPOINT_LOGICAL_ADDRESS VARCHAR2 (255 CHAR)
ENDPOINT_PHYSICAL_ADDRESS VARCHAR2 (255 CHAR)
SMP_ID VARCHAR2 (64 CHAR)
FK_CERTIFICATE_ID NUMBER (19)
FK_SUBDOMAIN_ID NUMBER (19)
BDMSL_SMP_AUD_PK (ID, REV)
FKI5AF66KB7J5JV86AUHBG4HFL9 (REV)
SML.BDMSL_REV_INFO
P * ID NUMBER (19)
REVISION_DATE TIMESTAMP
* TIMESTAMP NUMBER (19)
USERNAME VARCHAR2 (255 CHAR)
BDMSL_REV_INFO_PK (ID)
SML.BDMSL_CERTIFICATE_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CERTIFICATE_ID VARCHAR2 (255 CHAR)
NEW_CERT_CHANGE_DATE TIMESTAMP
PEM_ENCODING CLOB
VALID_FROM TIMESTAMP
VALID_UNTIL TIMESTAMP
NEW_CERT_ID NUMBER (19)
BDMSL_CERTIFICATE_AUD_PK (ID, REV)
FKHN0PXSBY9H6NPH5UWE45F8BQF (REV)
SML.BDMSL_CERTIFICATE_DOMAIN_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CERTIFICATE VARCHAR2 (255 CHAR)
CRL_URL VARCHAR2 (1000 CHAR)
IS_ADMIN NUMBER (1)
IS_ROOT_CA NUMBER (1)
PEM_ENCODING CLOB
TRUSTSTORE_ALIAS VARCHAR2 (512 CHAR)
VALID_FROM TIMESTAMP
VALID_UNTIL TIMESTAMP
FK_SUBDOMAIN_ID NUMBER (19)
BDMSL_CERTIFICATE_DOMAIN_AUD_PK (ID, REV)
FKE1JQWWRMQXH1QG3J4J838HW6F (REV)
SML.BDMSL_PARTICIPANT_IDENT_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CNAME_HASH VARCHAR2 (255 CHAR)
NAPTR_HASH VARCHAR2 (255 CHAR)
PARTICIPANT_ID VARCHAR2 (255 CHAR)
SCHEME VARCHAR2 (255 CHAR)
FK_SMP_ID NUMBER (19)
BDMSL_PARTICIPANT_IDENT_AUD_PK (ID, REV)
FKN1O91K0H87BXQ4EO3KL8SGOB5 (REV)
SML.BDMSL_ALLOWED_WILDCARD_AUD
P * FK_CERTIFICATE_ID NUMBER (19)
P * SCHEME VARCHAR2 (255 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
BDMSL_ALLOWED_WILDCARD_AUD_PK (FK_CERTIFICATE_ID, SCHEME, REV)
FKSNOXB79LXXPGFD2M3MJMSW8MR (REV)
SML.BDMSL_SUBDOMAIN_AUD
P * SUBDOMAIN_ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DESCRIPTION VARCHAR2 (1024 CHAR)
DNS_RECORD_TYPES VARCHAR2 (128 CHAR)
DNS_ZONE VARCHAR2 (512 CHAR)
PARTICIPANT_ID_REGEXP VARCHAR2 (1024 CHAR)
SMP_IA_CERT_REGEXP VARCHAR2 (255 CHAR)
SMP_URL_SCHEMAS VARCHAR2 (255 CHAR)
SUBDOMAIN_NAME VARCHAR2 (255 CHAR)
BDMSL_SUBDOMAIN_AUD_PK (SUBDOMAIN_ID, REV)
FKQXIHYF6OTWS3F0TBFLGOMVIC4 (REV)
SML.BDMSL_CERTIFICATE_DOMAIN_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CERTIFICATE VARCHAR2 (255 CHAR)
CRL_URL VARCHAR2 (1000 CHAR)
IS_ADMIN NUMBER (1)
IS_ROOT_CA NUMBER (1)
PEM_ENCODING CLOB
TRUSTSTORE_ALIAS VARCHAR2 (128 CHAR)
VALID_FROM TIMESTAMP
VALID_UNTIL TIMESTAMP
FK_SUBDOMAIN_ID NUMBER (19)
BDMSL_CERTIFICATE_DOMAIN_AUD_PK (ID, REV)
FKE1JQWWRMQXH1QG3J4J838HW6F (REV)
SML.BDMSL_CONFIGURATION_AUD
P * PROPERTY VARCHAR2 (512 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DESCRIPTION VARCHAR2 (4000 CHAR)
VALUE VARCHAR2 (4000 CHAR)
BDMSL_CONFIGURATION_AUD_PK (PROPERTY, REV)
FK86TLUD8XCE1X5OEG200OKCV3V (REV)
SML.BDMSL_CERTIFICATE_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CERTIFICATE_ID VARCHAR2 (255 CHAR)
NEW_CERT_CHANGE_DATE TIMESTAMP
PEM_ENCODING CLOB
VALID_FROM TIMESTAMP
VALID_UNTIL TIMESTAMP
NEW_CERT_ID NUMBER (19)
BDMSL_CERTIFICATE_AUD_PK (ID, REV)
FKHN0PXSBY9H6NPH5UWE45F8BQF (REV)
SML.BDMSL_REV_INFO
P * ID NUMBER (19)
REVISION_DATE TIMESTAMP
* TIMESTAMP NUMBER (19)
USERNAME VARCHAR2 (255 CHAR)
BDMSL_REV_INFO_PK (ID)
SML.BDMSL_ALLOWED_WILDCARD_AUD
P * FK_CERTIFICATE_ID NUMBER (19)
P * SCHEME VARCHAR2 (255 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
BDMSL_ALLOWED_WILDCARD_AUD_PK (FK_CERTIFICATE_ID, SCHEME, REV)
FKSNOXB79LXXPGFD2M3MJMSW8MR (REV)
SML.BDMSL_SMP_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
ENDPOINT_LOGICAL_ADDRESS VARCHAR2 (255 CHAR)
ENDPOINT_PHYSICAL_ADDRESS VARCHAR2 (255 CHAR)
SMP_ID VARCHAR2 (64 CHAR)
FK_CERTIFICATE_ID NUMBER (19)
FK_SUBDOMAIN_ID NUMBER (19)
BDMSL_SMP_AUD_PK (ID, REV)
FKI5AF66KB7J5JV86AUHBG4HFL9 (REV)
SML.BDMSL_DNS_RECORD_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
DNS_ZONE VARCHAR2 (255 CHAR)
NAME VARCHAR2 (255 CHAR)
SERVICE VARCHAR2 (32 CHAR)
TYPE VARCHAR2 (32 CHAR)
VALUE VARCHAR2 (512 CHAR)
BDMSL_DNS_RECORD_AUD_PK (ID, REV)
FKCIOFGW2B2JVV0TQSG2Q7UYH4M (REV)
SML.BDMSL_MIGRATE_AUD
P * MIGRATION_KEY VARCHAR2 (50 CHAR)
P * PARTICIPANT_ID VARCHAR2 (255 CHAR)
P * SCHEME VARCHAR2 (255 CHAR)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
MIGRATED NUMBER (1)
NEW_SMP_ID VARCHAR2 (64 CHAR)
OLD_SMP_ID VARCHAR2 (64 CHAR)
BDMSL_MIGRATE_AUD_PK (MIGRATION_KEY, PARTICIPANT_ID, SCHEME, REV)
FKQICRT1PFWNTOJYCM43BFO9XOF (REV)
SML.BDMSL_PARTICIPANT_IDENT_AUD
P * ID NUMBER (19)
PF* REV NUMBER (19)
REVTYPE NUMBER (3)
CREATED_ON TIMESTAMP
LAST_UPDATED_ON TIMESTAMP
CNAME_HASH VARCHAR2 (255 CHAR)
NAPTR_HASH VARCHAR2 (255 CHAR)
PARTICIPANT_ID VARCHAR2 (255 CHAR)
SCHEME VARCHAR2 (255 CHAR)
FK_SMP_ID NUMBER (19)
BDMSL_PARTICIPANT_IDENT_AUD_PK (ID, REV)
FKN1O91K0H87BXQ4EO3KL8SGOB5 (REV)

Figure 21 - Data model overview

9.2.2. Global description of the tables

Table	Description
bdmsl_allowed_wildcard	It is possible for a given Service Metadata Publisher to provide the metadata for all participant identifiers belonging to a particular participant identifier scheme. If this is the case, then it corresponds to the concept of a "wildcard" CNAME record in the DNS, along the lines: *.<schemeID>.<SML domain> CNAME

Table	Description
	<SMP domain><SMP domain> may either be the domain name associated with the SMP, or an alias for it. This implies that all participant identifiers for that schemeID will have addresses that resolve to the single address of that one SMP - and that as result only one SMP can handle the metadata for all participant identifiers of that scheme. Wildcard records are indicated through the use of “*" as the participant identifier in the operations of the ManageParticipantIdentifier interface. This table identifies the SMP with their certificates and map them to schemes for which they can create wildcard records.
bdmsl_certificate	List of SMPs identified with their certificates.
bdmsl_certificate_domain	Associates the root certificates to the DNS domains
bdmsl_configuration	Table containing all the configuration
bdmsl_migrate	Contains the participants migrated or to be migrated
bdmsl_participant_identifier	List of the participants
bdmsl_smp	List of the SMPs
bdmsl_subdomain	List of Subdomains
bdmsl_allowed_wildcard_aud	Audit table for: bdmsl_allowed_wildcard
bdmsl_certificate_aud	Audit table for: bdmsl_certificate
bdmsl_certificate_domain_aud	Audit table for: bdmsl_certificate_domain
bdmsl_configuration_aud	Audit table for: bdmsl_configuration
bdmsl_migrate_aud	Audit table for: bdmsl_migrate
bdmsl_participant_identifier_aud	Audit table for: bdmsl_participant_identifier
bdmsl_smp_aud	Audit table for: bdmsl_smp
bdmsl_subdomain_aud	Audit table for: bdmsl_subdomain
bdmsl_info_rev	Audit table for: audit info table with date and user who created the change.

Table 3 - Tables list

9.2.3. Detailed description of the tables

Table	Column	Description
bdmsl_allowed_wildcard	scheme	The scheme on which the wildcard applies
	fk_certificate_id	The foreign key to the certificate
	created_on	Date of creation
	last_updated_on	Date of the last update
bdmsl_certificate	id	The id is a primary key of the certificate entry.
	certificate_id	The certificate_id is a natural key composed of the subject and the serial number of the certificate
	valid_from	Start validity date of the certificate
	valid_until	Expiry date of the certificate
	pem_encoding	PEM encoding for the certificate
	new_cert_change_date	The date of the change for the new certificate
	new_cert_id	The new certificate id. Links to the

Table	Column	Description
		certificate that will be valid after the current one is expired. At the migration date, it aims to replace the existing certificate
	created_on	Date of creation
	last_updated_on	Date of the last update
bdmsl_certificate_domain	certificate	Natural key for authorization domain certificate. Value is created from certificate Subject value
	id	Domain certificate entry primary key.
	fk_subdomain_id	The foreign key to the subdomain
	truststore_alias	Alias which correspond to certificate truststore alias.
	crl_url	URL to the certificate revocation list (CRL)
	created_on	Date of creation
	valid_from	Start validity date of the certificate
	valid_until	Expiry date of the certificate
	us_admin	If user identified by this certificate has role ADMIN
	pem_encoding	PEM encoded certificate
	is_root_ca	If certificate is root CA or not
	last_updated_on	Date of the last update
bdmsl_configuration	property	Name of the property
	value	Value of the property
	description	Description of the property
	created_on	Date of creation
	last_updated_on	Date of the last update
bdmsl_migrate	scheme	The scheme of the participant identifier to be migrated
	participant_id	The participant identifier to be migrated
	migration_key	The migration key is a code that must be passed out-of-band to the SMP which is taking over the publishing of the metadata for the participant identifier. This code must contain: • 8 characters minimum • 24 characters maximum • 2 Special Characters @#$%()[]{}*^-!~|+= • 2 Upper Case letters minimum • 2 Lower Case letters minimum • 2 Numbers minimum • No white spaces
	new_smp_id	The id of the SMP after the migration

Table	Column	Description
	old_smp_id	The id of the old SMP (before the migration)
	migrated	True if the migration is done
	created_on	Date of creation
	last_updated_on	Date of the last update
bdmsl_participant_identifier	id	Surrogate key of participant
	participant_id	The participant identifier
	naptr_hash	Hash of participant used for naptr record
	cname hash	Hash of participant used for cname record
	scheme	The scheme of the participant identifier
bdmsl_smp	fk_smp_id	The foreign key to the SMP identifier
	created_on	Date of creation
	last_updated_on	Date of the last update
	id	Surrogate key of smp
	smp_id	The SMP identifier
	fk_certificate_id	The foreign key to the certificate
	endpoint_physical_address	The physical address of the endpoint. This physical address is used as the ALIAS on the CNAME DNS record.
	endpoint_logical_address	The logical address of the endpoint
	created_on	Date of creation
	fk_subdomain_id	The foreign key to the subdomain
	last_updated_on	Date of the last update
bdmsl_subdomain	subdomain_id	The subdomain identifier
	subdomain_name	The subdomain name
	created_on	Date of creation
	last_updated_on	Date of the last update
	description	Subdomain description
	dns_record_type	Type of DNS Record when registering/updating participant, “all” means that both DNS record types are accepted as possible values: [cname, naptr, all].
	dns_zone	Domain (dns zone) for subdomain.
	participant_id_regexp	Regex allows specific and described ids only or * instead for having wildcards.
	smp_url_schemas	Protocol that MUST be used for LogicalAddress when registering new SMP. “all” means both protocols are accepted as possible values: [http, https, all].
	smp_ia_cert_regexp	Regex for authorizing certificates when using issuer based domain authorization.

Table 4 - Tables fields

10. Scheduler

The Spring Framework provides abstractions for asynchronous execution and scheduling of tasks.

In the `applicationContext.xml` file, we can define the jobs to be scheduled:

```
<task:scheduler id="scheduler" pool-size="1"/>
<task:scheduled-tasks scheduler="scheduler">
    <task:scheduled ref="manageCertificateService" method="changeCertificates"
cron="${certificateChangeCronExpression}"/>
</task:scheduled-tasks>
```

The previous example will execute every day at 2 am the method `changeCertificate` of the bean name `manageCertificateService`.

In case of the execution of the application on a clustered environment, it is necessary to make sure that multiple jobs do not perform the same task at the same time. The use of a pessimistic lock can be useful:

```
@Override
public List<CertificateBO> findCertificatesToChange(Calendar currentDate) throws
TechnicalException {
    // This method is used in the context of a job that can be run on a clustered
environment. To avoid concurrency issues, we do here a SELECT FOR UPDATE
    Query query = getEntityManager().createQuery("SELECT cert from
CertificateEntity cert where cert.newCertificateChangeDate <= :currentDate")
            .setParameter("currentDate",
currentDate).setLockMode(LockModeType.PESSIMISTIC_WRITE);

    [...]
}
```

All cron expressions are initialized from values in the database. When a parameter is changed, server needs to be restarted.

10.1. Change Certificate

This job changes the certificates that have a migration date in the past or at the present day and deletes the older ones.

This task runs according to this parameter:

BDMSL_CONFIGURATION		
PROPERTY	**VALUE**	**DESCRIPTION**
certificateChangeCronExpression	0 0 2 ? * *	Cron expression for the changeCertificate job. Example: 0 0 2 ? * * (everyday at 2:00 am)

This parameter can be updated manually on the database or by webservice SetProperty().When parameter is changed, the server needs to be restarted.

10.2. Update database properties

If SML is set in “cluster mode” (property sml.cluster.enabled) then this job updates application business properties from database. Cron task is used to ensure that all nodes in a cluster update properties at the same time. Task first checks if there are changed properties according to last_update_on values. If there is a last_update_on value newer then the one from the last property update, the update of properties is triggered.

This task runs according to this parameter:

BDMSL_CONFIGURATION		
PROPERTY	**VALUE**	**DESCRIPTION**
sml.property.refresh.cronJobExpression	0 53 */1 * * *	Property refresh cron task expression (every hour, 7 minutes before the hour)
sml.cluster.enabled	false	Property defines if SML is running in cluster mode.

This parameter can be updated manually on the database or by webservice SetProperty().

10.3. Data Inconsistency Analyzer

This job looks for inconsistencies between the database and the DNS. It first accesses the DNS to retrieve all SMPs and Participants. It then compares DNS data against Database. All discrepancies in entries are reported to the user by means of a report email.

As the previous job, this task will run according to the parameters below:

BDMSL_CONFIGURATION		
PROPERTY	**VALUE**	**DESCRIPTION**
dataInconsistencyAnalyzer.cronJobExpression	0 0 3 ? * *	Cron expression: 0 0 3 ? * * (every day at 3:00 am)

dataInconsistencyAnalyzer.recipientEmail	email@example.com	Email address to receive Data Inconsistency Checker results
dataInconsistencyAnalyzer.senderEmail	email@example.com	Sender email address for reporting Data Inconsistency Analyzer.
dataInconsistencyAnalyzer.serverInstance	localhost	Server instance (hostname) to generate report. Property is needed in cluster where we define which instance should generate the report.

These parameters can be updated manually on database or by webservice SetProperty().

10.4. SMP with expired certificates Analyzer

This job looks for SMPs with expired certificates. All SMPs with expired certificates are reported to the user by means of a report email.

As the previous job, this task will run according to the parameters below:

BDMSL_CONFIGURATION		
PROPERTY	**VALUE**	**DESCRIPTION**
report.expiredSMPCertificates.cron	0 22 6 ? * *	Cron expression for triggering the report generation of the expired SMP certificates

report.expiredSMPCertificates.recipientEmail	email@example.com	Email address to receive the report
report.expiredSMPCertificates.senderEmail	email@example.com	Sender email address of the report.
report.expiredSMPCertificates.serverInstance	localhost	Server instance (hostname) to generate report. Property is needed in cluster where we define which instance should generate the report.

These parameters can be updated manually on database or by webservice SetProperty().

11. Email SMTP configuration

An inconsistency report is sent by email. As a consequence a mail server needs to be configured in the database. Below are smtp server configuration properties.

BDMSL_CONFIGURATION		
PROPERTY	**VALUE**	**DESCRIPTION**
mail.smtp.host	smtp.server.com	Smtp server host
mail.smtp.port	465	Smtp server port
mail.smtp.protocol	smtp	Protocol (smtp, smtps)
mail.smtp.username	smtpuser	Username for authentication on server
mail.smtp.password	P/npBabppDazizAjWkNs6Q==	Encrypted password
mail.smtp.properties	mail.smtp.ssl:true;mail.smtp.auth:true;mail.smtp.socketFactory.class:javax.net.ssl.SSLSocketFactory	Additional properties

12. VALIDATIONS

12.1. Participant ID validation per Domain

SML provides to each existent domain the possibility to validate its participant ids through Regular Expression. The following property in the table BDMSL_SUBDOMAIN allows validating participant ids:

Example:

For subdomain with name: **peppol.acc.edelivery.tech.ec.europa.eu**

PARTICIPANT_ID_REGEXP = ^((((1234|45678|9584|9635):).*)|(*))$

For subdomain with name: **generalerds.acc.edelivery.tech.ec.europa.eu**

PARTICIPANT_ID_REGEXP = ^.*$

12.2. Logical Address validation per Domain

Two addresses are needed to create a SMP: the Logical and the Physical Addresses. As from SML version 3.1, the configuration allows to specify if the Logical Address may accept **HTTP** or **HTTPS** protocol for the Create SMP Operation.

An additional property 'SMP_URL_SCHEMAS' has been introduced in the table BDMSL_ SUBDOMAN in that purpose.

The possible values for this property are (all, http or https). The option 'all' means that both protocols are accepted.

Example for **test.acc.edelivery.tech.ec.europa.eu**:

SMP_URL_SCHEMAS = all

13. Security

13.1. DNS

13.1.1. DNS specifications

The SML specification [REF1] states in the chapter *5. DNS spoof mitigation*:

"The regular lookup of the address of the SMP for a given participant ID is performed using a standard DNS lookup. There is a potential vulnerability of this process if there exists at least one "rogue" certificate (e.g. stolen or otherwise illegally obtained). In this vulnerability, someone possessing such a rogue certificate could perform a DNS poisoning or a man-in-the-middle attack to fool senders of documents into making a lookup for a specific identifier in a malicious SMP (that uses the rogue certificate), effectively routing all messages intended for one or more recipients to a malicious access point. This attack could be used for disrupting message flow for those recipients, or for gaining access to confidential information in these messages (if the messages were not separately encrypted). One mitigation for this kind of attack on the DNS lookup process is to use DNSSEC rather than plain DNS. DNSSEC allow the authenticity of the DNS resolutions to be checked by means of a trust anchor in the domain chain. Therefore, it is recommended that an SML instance uses the DNSSEC infrastructure."

Thus, in order to mitigate the risk of DNS spoofing, the DNSSEC can be used in the eDelivery BDMSL application. The Domain Name System Security Extensions (DNSSEC) is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol (IP) networks.

3 properties allow the administrator to configure the DNSSEC:

Property	Description
dnsClient.SIG0Enabled	'true' if the SIG0 signing is enabled. Required fr DNSSEC. Possible values: true/false
dnsClient.SIG0PublicKeyName	The public key name of the SIG0 key
dnsClient.SIG0KeyFileName	The actual SIG0 key file. Should be just the filename if the file is in the classpath or in the 'configurationDir'

Table 5 - DNS Properties

Remark: It is important to be aware that the BDMSL deployed at the European Commission is not configured to use DNSSEC on the actual public DNS server:

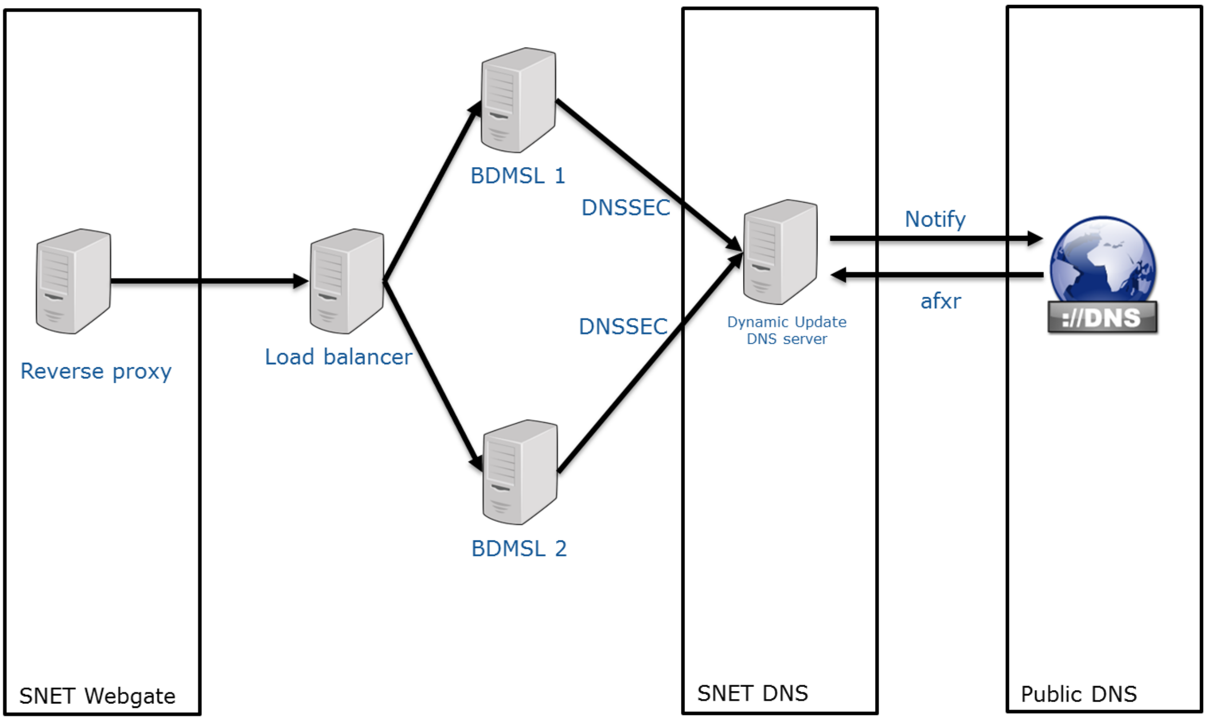


Figure 22 - BDMSL hosting at the EC

13.1.2. DNS implementation

The BDMSL registers 1 CNAME record for each SMP.

The BDMSL registers 2 types of DNS records for each participant:

- 1 CNAME record with the prefix "B-"
- 1 U-NAPTR record without prefix "B-"

Thus, for each participant, 2 records exist at the same time in the DNS and don't conflict because they don't use the same hash algorithm. For example, if a SMP registers the participant "0010:5798000000001" then:

- The MD5 hash is "e49b223851f6e97cbfce4f72c3402aac"
- The SHA-256 Base32 hash is "XUKHFQABQZIKI3YKVR2FHR4SNFA3PF5VPQ6K4TONV3LMVSY5ARVQ"

As a result, the BDMSL registers 2 records in the DNS:

```
>dig CNAME B-e49b223851f6e97cbfce4f72c3402aac.iso6523-actorid-
upis.acc.edelivery.tech.ec.europa.eu @ddnsext.tech.ec.europa.eu

B-e49b223851f6e97cbfce4f72c3402aac.iso6523-actorid-upis.edelivery.eu. 60 IN CNAME
smp.edelivery.tech.ec.europa.eu

>dig NAPTR XUKHFQABQZIKI3YKVR2FHR4SNFA3PF5VPQ6K4TONV3LMVSY5ARVQ.iso6523-actorid-
upis.edelivery.eu @ddnsext.tech.ec.europa.eu

XUKHFQABQZIKI3YKVR2FHR4SNFA3PF5VPQ6K4TONV3LMVSY5ARVQ.iso6523-actorid-
upis.edelivery.eu. 60 IN NAPTR 100 10 "U" "Meta:SMP" "!.*!http://smp.edelivery.eu/iso6523-
actorid-upis::0010:5798000000001!" .
```

In order to mitigate the risk of DNS spoofing, the BDMSL can use the DNSSEC infrastructure. The deployment infrastructure is described in section *13.1 DNS*.

13.2. Encryption Key

SML uses a private key to encrypt and decrypt the keystore password used by SML to sign any response and the proxy password.

13.2.1. How to generate a private key

- Download one of the latest BDMSL .war files from the repository on the Digital site
- Extract the .war file using any extracting tool
- Run the following commands to create a private key

  1. cd bdmsl-webapp-XXX-weblogic-oracle (XXX being the SML version number you are intalling)
  2. java -cp "WEB-INF/lib/*" eu.europa.ec.bdmsl.common.util.PrivateKeyGenerator c:\temp\encriptionPrivateKey.private

  Required parameter = Full directory path where the private key will be created

Example:

Printed result:

Private key created at c:\temp\encriptionPrivateKey.private

NOTE: Once the private key is generated, please copy the private key file name "Ex: `encriptionPrivateKey.private`" to the value of the property `encriptionPrivateKey` in the table BDMSL_Configuration, and copy the private file to the path configured in the property `configurationDir`.

13.2.2. How to encrypt a password

After generating a private key at the section 12.2.1, please configure the proxy or keystore (used to sign response) password if needed as follows:

- Inside the folder already extracted from the BDMSL .war file, please run below command:

  java -cp "WEB-INF/lib/*" eu.europa.ec.bdmsl.common.util.EncryptPassword c:\temp\privateKey.private Password123

  1st parameter = private key location

  2nd parameter = plain text password

- To configure the proxy password, please copy the printed encrypted and base64 encoded password to the value of the property `httpProxyPassword` in the table `BDMSL_CONFIGURATION`

Example:

Property	Description
httpProxyPassword	vXA7JjCy0iDQmX1UEN1Qwg==

Table 6 - DNS Properties

- To configure keystore password, please copy the printed encrypted and base64 encoded password to the value of the property `keystorePassword` in the table BDMSL_CONFIGURATION

  Example:

Property	Description
keystorePassword	vXA7JjCy0iDQmX1UEN1Qwg==

Table 7 - DNS Properties

13.3. Authentication

The authentication relies on the use of a Public Key Infrastructure (PKI). The services are all secured at the transport level with a two-way SSL / TLS connection. The requestor must authenticate using a client certificate issued for use in the infrastructure by a trusted third-party. The server will reject SSL clients that do not authenticate with a certificate issued under a trusted root.

WS-Security is only used for signing the response from the BDMSL to the SMP. It allows the SMP to validate that the request was correctly processed and acknowledged by the BDMSL.

The authentication for the user and admin interface is also performed with 2-way SSL and the user must provide the SMP's certificate.

The authentication is performed through a custom interceptor named `CertificateAuthenticationInterceptor`. This interceptor is configured to intercept any incoming request in the `cxf-servlet.xml` configuration file:

```
<cxf:bus>
    <cxf:inInterceptors>
        <ref bean="certificateAuthenticationInterceptor"/>
    </cxf:inInterceptors>
    [...]
</cxf:bus>
```

The interceptor extracts the certificate information from the request and then validates it.

A certificate is valid if:

- The direct issuer or certificate itself is trusted in the bdmsl_certificate_domain table and truststore.
- If certificate is trusted by direct issuer and subject matches the regular expression

- If the whole certificate chain is registered in truststure and is valid:
  - It is not revoked according to its certificate revocation list (CRL)
  - It is valid for the current date

This certificate is then automatically used to authenticate the client using the Spring security framework. If the certificate is valid, then the client is authenticated and the certificate details are stored in the security context. Otherwise, a `UnauthorizedFault` is thrown.

13.3.1. SSL configured on the application server

The 2-way SSL configuration can be directly set up on the application server hosting the application:

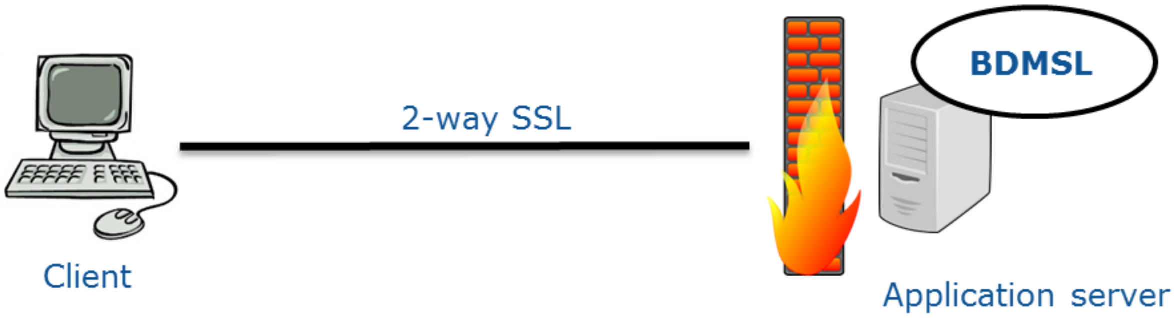


Figure 23 - SSL configured on the application server

In this type of configuration, the client certificate is passed in the request and can be intercepted in the `javax.servlet.request.X509Certificate` attribute.

13.3.2. Reverse proxy with SSL

The server can be behind a reverse proxy. In this case, 2-way SSL is set up on the reverse proxy and the application server hosting the application can use the HTTP protocol:

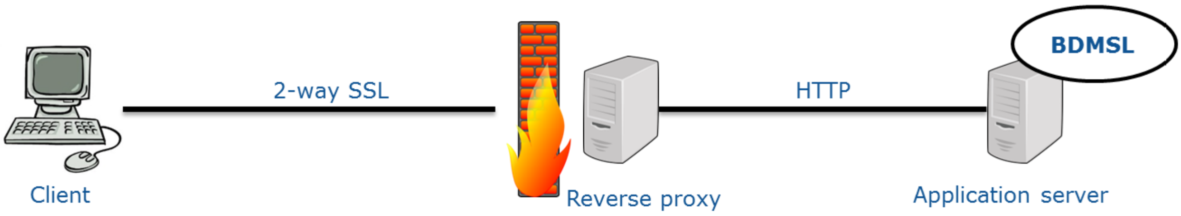


Figure 24 - Reverse proxy with SSL

In this configuration, the certificate information is stored in the HTTP header, in the `Client-Cert` attribute. From BDMSL 4.1 version on, BDMSL supports HTTP header `SSLClientCert` with base64 encoded Client X509Certificate.

13.3.3. Admin Access

The system administrators can access the services like ChangeCertificate by using certificate authentication. The domain certificate in the bdmsl_certificate_domain must have flag Is_Admin_ set to true. Admin certificate can only be NonRootPKI certificate.

13.3.4. Monitor Access

The Monitor user can only access service isAlive(). User is authenticated by security token included in the HTTP Header in the following way:

- The HTTP header needs to have the following attributes: Monitor-Token, Admin-Pwd (obsolete)
- The password needs to be hashed with BCrypt algorithm
- The password will be stored in the configuration table under the key adminPassword

13.3.5. Enable/disable BlueCoat Authentication flag

In order to authenticate into SML using the header Client-Cert attribute, the flag authentication.bluecoat.enabled and authorization.domain.legacy.enabled must be enabled in the table BDMSL_CONFIGURATION (BlueCoat Authentications are rejected otherwise).

13.4. Authorizations

13.4.1. Roles

There are 3 roles defined in the application:

Property	Description	Current condition
ROLE_SMP	The role specific to SMP clients	The CN (Common Name) must start with "SMP_" or "DN" (Distinguished name) and must match regular expression from configuration property: authorization.smp.certSubjectRegex. Please see §14.1 for further information For a Non Root Certificate Authority, the CN (Common Name) must contain "_SMP_". Please see §13.4.2 for further information.
ROLE_MONITOR	The role only for invoking isAlive function	No certificate needed, the right credentials must be sent via the HTTP header attribute Admin-Pwd or Monitor-Token
ROLE_ADMIN	The role for the administrator of the BDMSL	Admin is authenticated by Non Root Certificate Authority in domain certificate table. Certificate must have flag is_admin set to true.

Table 8 - Roles

The authorizations are set using the Spring security framework using the `@PreAuthorize` annotation on the methods of the service layer:

```
@Override
@PreAuthorize("hasAnyRole('ROLE_SMP', 'ROLE_ADMIN')")
@Transactional(readOnly = false, rollbackFor = Exception.class)
public void prepareChangeCertificate(PrepareChangeCertificateBO
prepareChangeCertificateBO) throws BusinessException, TechnicalException {
    [...]
}
```

In the previous example, the method can only be called if the current client has any of the roles `ROLE_SMP` or `ROLE_ADMIN`. Otherwise, a `UnauthorizedFault` SOAP fault is thrown.

13.4.2. Granting ROLE_SMP

The BDMSL application can perform 2 different types of SMP domain authorizations:

- One is the Domain **certificate-issuer-based authorization** (also known as **Root Certificate Authority)** method that would automatically authorize all the certificate-issuer trusted certificates.
- The other is the **certificate-based authorization** (also known as **NON Root Certificate Authority)** method to authorize an individual SMP X.509 certificate.

13.4.2.1. Certificate-issuer-based authorization

The authorization is suitable for business domains with a high number of SMP service providers. The SML domain owner must provide a dedicated issuer certificate for issuing all SMP service providers certificates in a particular domain. When the business owner issues a certificate to the SMP service provider, it automatically authorizes the certificate to access the BDMSL business domain. The SMP service provider can then create and manage an SMP entry and its Participant list. At the SMP entry creation, the SMP's Certificate is automatically registered to the SMP entry. After the SMP entry registration, the SMP entry itself and its participant list can be modified only by the SMP entry's registered certificate.

BDMSL introduces the ability to define regular expressions on the SMP X509 Certificate’s subject DN to filter the SMP authorization to specific certificates issued by the registered domain issuer. The functionality also enables business owners to use certificate issued for other purposes, as the case for also issuing the AP certificates. The regular expression is defined by the BDMSL configuration property, authorization.smp.certSubjectRegex or in the the domain table column: SMP_IA_CERT_REGEXP.

Below is an example of a regular expression where the only SMP certificates allowed have subject CN starting with “SMP_“ or the subject DN containing organization unit (OU) with value: “PRODUCTION SMP:

Regular expression: ^.*(CN=SMP_|OU=PRODUCTION SMP).*$

REMARK: The system administrator must register and authorize the issuer certificate in the BDMSL and associate it to the domain.

For granting a certificate to the domain, BDMSL checks the *issuer* of the certificate against the trusted RootCA list provided by the SML database certificate domain table and (optionally) SML truststore. The database flag isRootCA for the issuer certificate must be set to true.

A Root Certificate Authority owns a PKI (Public Key Infrastructure) to manage certificates.

13.4.2.2. Certificate-based authorization

This authorization is suitable for business domains with fewer SMP service providers and in cases where maintaining a dedicated certificate issuer for the domain's SMP certificates is not an option. In this case, each SMP certificate must be added and authorized in the BDMSL business domain by the administrator.

For granting a certificate as trusted, BDMSL checks the certificate itself against the trusted NON-RootCA provided by the SML database. The database flag isRootCA must be set false.

A Non Root Certificate Authority does not own any PKI (Public Key Infrastructure) to manage certificates, a third party entity is responsible for managing certificates for such case.

Non Root and Root Certificate Priority

Apart from the aforementioned cases, SML allows certificates that are configured with Root and Non Root CA simultaneously. In such cases, SML gives priority to the Non Root CA, meaning that if a certificate matches "Non Root CA", the SML ignores "Root CA".

13.5. WS-Security

If the property `signResponse` is set to true, then the responses are signed using the WS-Security framework.

The response signature is performed through a custom interceptor named `SignResponseInterceptor`. This interceptor is configured to intercept any outgoing request in the `cxf-servlet.xml` configuration file:

```
<cxf:bus>
    [...]
    <cxf:outInterceptors>
        <ref bean="signResponseInterceptor" />
    </cxf:outInterceptors>
</cxf:bus>
```

14. Technical requirements

This chapter describes the minimum and recommended system requirements to operate a BDMSL component.

14.1. Hardware

Type	Minimum	Recommended
Processor	1 CPU core	4 CPU core
Memory (RAM)	2GB	8GB or more
Disk space	5GB	Depends on usage

Table 9 - Hardware requirements

14.2. Software

14.2.1. Recommended stack

- Ubuntu 18.04 LTS 64 bits
- Oracle Java SE 8
- Oracle WebLogic Server 12c (12.2.1.4+)
- Oracle Database 11g (11.2.0.4.0)

14.2.2. Operating Systems

Any operating system that is compliant with one of the supported JVM.

14.2.3. Java Virtual Machines

- Oracle Java SE JRE 8
- OpenJDK 8

14.2.4. Java Application Servers

- Apache Tomcat 9
- Oracle WebLogic Server 12c (12.2.1.4+)

14.2.5. Databases

- MySQL 8
- Oracle Database 11g (11.2.0.4.0)

14.2.6. Web Browsers

- Internet Explorer 8 or newer

- Mozilla Firefox
- Google Chrome

15. Configuration

15.1. Application Configuration

Property	Example	Mandatory	Description	Enc.
adminPassword	$2a$10$...Bi	FALSE	BCrypt Hashed password to access admin services	FALSE
authentication.bluecoat.enabled	FALSE	TRUE	Is blue coat enabled. Possible values: true/false. NOTE: The property should be enabled only if is protected by the reverse proxy.	FALSE
authentication.sslclientcert.enabled	FALSE	TRUE	Enable/Disable SSLClientCert header authentication. Possible values: true/false. NOTE: The property should be enabled only if is protected by the reverse proxy.	FALSE
authorization.smp.certSubjectRegex	^.*(CN=SMP_|OU=PEPPOL TEST SMP).*$	TRUE	User with ROOT-CA is granted SMP_ROLE only if its certificates Subject matches configured regexp	FALSE
authorization.domain.legacy.enabled	TRUE	TRUE	If legacy authorization is enabled, then domain authorization is done based only on domain certificate table data comparing certificate Subject or Issuer Values. In case of false: BDMSL must have SML truststore configured. And the Domain Trust is verified also by the BDMSL trustststore. In case of false value Clien-Cert header cannot be used.	FALSE
cert.revocation.validation.graceful	TRUE	TRUE	In case of authorization.domain.legacy.enabled is ser to false. All certificate in truststore chain are validated and CRL url is retrieved from the certificates directly. Graceful validation of certificate revocation. If URL retrieving does not succeed, do not throw error!.	FALSE
cert.revocation.validation.crl.protocols	http://,https://",	TRUE	In case of authorization.domain.legacy.enabled is set to false. All certificate in	FALSE

Property	Example	Mandatory	Description	Enc.
			truststore chain are validated and CRL url is retrieved from the certificates directly. Comma separated list of allowed crl protocols for fetching the CRL list.	
unsecureLoginAllowed	FALSE	TRUE	true if the use of HTTPS is not required. If the VALUES is set to true, then the user unsecure-http-client is automatically created. Possible VALUES: true/false	FALSE
configurationDir	./	TRUE	The path to the folder containing all the configuration files (keystore and sig0 key)	FALSE
sml.property.refresh.cronJobExpression	0 53 */1 * * *	TRUE	Property refresh cron expression (def 7 minutes to each hour)!	FALSE
certificateChangeCronExpression	0 0 2 ? * *	TRUE	Cron expression for the changeCertificate job. Example: 0 0 2 ? * * (everyday at 2:00 am)	FALSE
smp.update.max.part.size	1000	FALSE	Maximum number of participants on SMP which are automatically updated/deleted when calling services: ManageServiceMetadataService/Update ManageServiceMetadataService/Delete If SMP has more participants then for - delete: the participants must be deleted first using delete participant service - update (only for SMP logical address when using NAPTR records): the creation of new SMP ID and migration participant to new SMP is only option.	FALSE
dataInconsistencyAnalyzer.cronJobExpression	0 0 3 ? * *	TRUE	Cron expression for dataInconsistencyChecker job. Example: 0 0 3 ? * * (everyday at 3:00 am)	FALSE
dataInconsistencyAnalyzer.recipientEmail	email@domain.com	TRUE	Email address to receive Data Inconsistency Checker results	FALSE
dataInconsistencyAnalyzer.senderEmail	automated-notifications@nsome-mail.eu	TRUE	Sender email address for reporting Data Inconsistency Analyzer.	FALSE

Property	Example	Mandatory	Description	Enc.
dataInconsistencyAnalyzer.serverInstance	localhost	TRUE	Server instance (hostname) to generate report.	FALSE
mail.smtp.host	mail.server.com	TRUE	Email server - configuration for submitting the emails.	FALSE
mail.smtp.port	25	TRUE	Smtp mail port - configu1ration for submitting the emails.	FALSE
mail.smtp.protocol	smtp	TRUE	smtp mail protocol- configuration for submitting the emails.	FALSE
mail.smtp.username		FALSE	smtp mail protocol- username for submitting the emails.	FALSE
mail.smtp.password		FALSE	smtp mail protocol - encrypted password for submitting the emails.	TRUE
mail.smtp.properties		FALSE	smtp mail ;-separated properties: ex: mail.smtp.auth:true;mail.smtp.starttls.enable:true;mail.smtp.quitwait:false.	FALSE
dnsClient.SIG0Enabled	FALSE	TRUE	true if the SIG0 signing is enabled. Required for DNSSEC. Possible VALUES: true/false	FALSE
dnsClient.show.entries	TRUE	FALSE	If true than service ListDNS transfer and show the DNS entries. (Not recommended for large zones) Possible VALUES: true/false	FALSE
dnsClient.tcp.timeout	TRUE	FALSE	DNS TCP timeout in seconds. If the value is not given then tcp timeout is set to default value 60 seconds.	FALSE
dnsClient.use.legacy.regexp	FALSE	TRUE	If value is 'true', then OASIS_BDXL regexp '^.*$' is used for NAPTR value generation else it is used the regular expression '.*' as defined in IETF RFC 4848.	FALSE
dnsClient.SIG0KeyFileName	SIG0.private	TRUE	The actual SIG0 key file. Should be just the filename if the file is in the classpath or in the configurationDir	FALSE
dnsClient.SIG0PublicKeyName	sig0.acc...ec.test.eu.	TRUE	The public key name of the SIG0 key	FALSE
dnsClient.enabled	FALSE	TRUE	true if registration of DNS records is required. Must be true in production. Possible VALUES: true/false	FALSE
dnsClient.publisherPrefix	publisher	TRUE	This is the prefix for the publishers (SMP). This is to be concatenated with the associated DNS domain in the table bdmsl_certificate_domain	FALSE
dnsClient.server	ddnsext.tech.ec.europa.eu	TRUE	The DNS server	FALSE
encriptionPrivateKey	encriptionPrivateKey.private	TRUE	Name of the 256 bit AES secret key to encrypt or decrypt passwords.	FALSE

Property	Example	Mandatory	Description	Enc.
useProxy	FALSE	TRUE	true if a proxy is required to connect to the internet. Possible VALUES: true/false	FALSE
httpProxyHost	localhost	TRUE	The http proxy host	FALSE
httpProxyPassword	vXA7JjCyEN1Qwg==	TRUE	Base64 encrypted password for Proxy.	TRUE
httpProxyPort	8012	TRUE	The http proxy port	FALSE
httpProxyUser	user	TRUE	The proxy user	FALSE
signResponse	FALSE	TRUE	true if the responses must be signed. Possible values: true/false	FALSE
keystoreAlias	senderalias	TRUE	The alias in the keystore for signing reponses.	FALSE
keystoreFileName	keystore.jks	TRUE	The (JKS or P12) keystore file. Should be just the filename if the file is in the classpath or in the configurationDir	FALSE
keystorePassword	vXA7JjCy0EN1Qwg==	TRUE	Base64 encrypted password for Keystore.	TRUE
truststoreFileName	truststore.p12	TRUE	The truststore file (JKS or p12) should be just the filename if the file is in the classpath or in the configurationDir	FALSE
truststorePassword	vXA7JjCy0EN1Qwg==	TRUE	Base64 encrypted password for Truststure.	TRUE
partyIdentifier.splitPattern	^(?i)\s*?(?<scheme>urn:oasis:names:tc:ebcore:partyid-type:(iso6523:[0-9]{4}|unregistered(:[^:]+)?))::?(?<identifier>.+)?\s*$	FALSE	Regular expression with groups scheme and identifier for splitting the URN identifiers to scheme and identifier part!	FALSE
report.expiredSMPCertificates.cron	0 22 6 ? * *	TRUE	Cron expression for triggering the report generation of expired SMP certificates job. Example: 0 22 6 ? * * (everyday at 3:00 am)"	FALSE
eport.expiredSMPCertificates.recipientEmail	email@domain.com	FALSE	Email address to receive expired SMP certificates report	FALSE
report.expiredSMPCertificates.senderEmail	notifications@nsome-mail.eu	FALSE	Sender email address for expired SMP certificates report.	FALSE
report.expiredSMPCertificates.serverInstance	localhost	FALSE	If sml.cluster.enabled is set to true then then instance (hostname) to generate report.	FALSE

Table 10 - Application properties

15.2. Multiple domains

SML is able to manage DNS records per domain. Any domain must be linked to only one certificate in the database.

Domain: It is used by the SML to authenticate to the DNS server and gain update privileges.

Example: **acc.edelivery.tech.ec.europa.eu** or **edelivery.tech.ec.europa.eu**

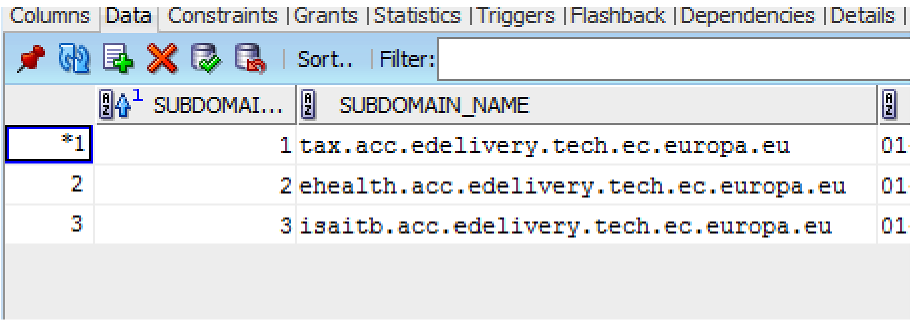
Subdomain: It belongs to a domain and must be provided to create DNS entries.

Example: **mycompany.acc.edelivery.tech.ec.europa.eu**

In order to configure a subdomain please follow the steps below:

1. Create a subdomain in the table BDMSL_Subdomain:

   Example:


Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details |
Sort..
Filter:

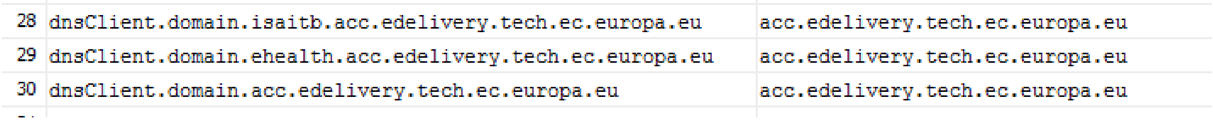
	SUBDOMAI...	SUBDOMAIN_NAME	
*1	1	tax.acc.edelivery.tech.ec.europa.eu	01
2	2	ehealth.acc.edelivery.tech.ec.europa.eu	01
3	3	isaitb.acc.edelivery.tech.ec.europa.eu	01

Figure 25 – Creating a subdomain

NOTE: It is mandatory to define the new subdomain as NON ROOT CA or ROOT CA in the column IS_ROOT_CA. Please check section 13.2.1 Granting ROLE_SMP

Define the subdomain configurations in the table BDMSL_SUBDOMAIN:

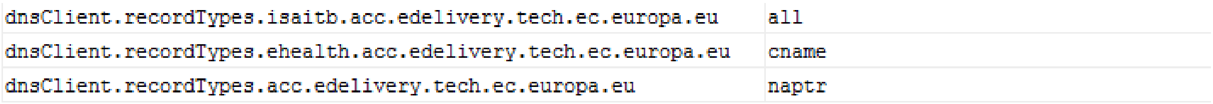
- *DNS_ZONE*= specify for every domain the name of the domain in the DNS server responsible for the subdomains.



28	dnsClient.domain.isaitb.acc.edelivery.tech.ec.europa.eu	acc.edelivery.tech.ec.europa.eu
29	dnsClient.domain.ehealth.acc.edelivery.tech.ec.europa.eu	acc.edelivery.tech.ec.europa.eu
30	dnsClient.domain.acc.edelivery.tech.ec.europa.eu	acc.edelivery.tech.ec.europa.eu

Figure 26 – Defining a domain for every subdomain

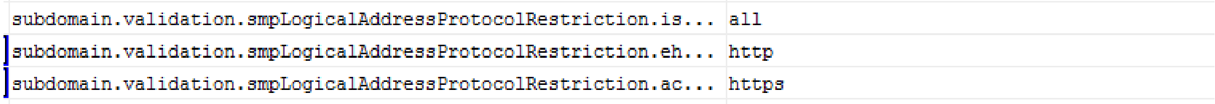
- *DNS_RECORD_TYPES* = specify for every domain the type of DNS Record accepted when registering/updating participant, 'all' means that both DNS record CNAME and NAPTR are accepted, possible values are [cname, naptr, all].



dnsClient.recordTypes.isaitb.acc.edelivery.tech.ec.europa.eu	all
dnsClient.recordTypes.ehealth.acc.edelivery.tech.ec.europa.eu	cname
dnsClient.recordTypes.acc.edelivery.tech.ec.europa.eu	naptr

Figure 27 – Defining the accepted types of DNS records

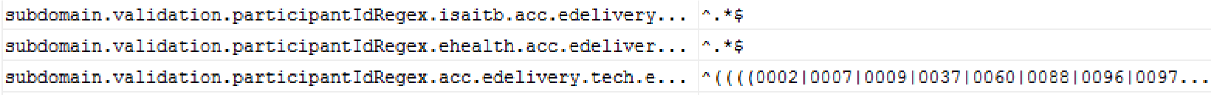
- *SMP_URL_SCHEMAS*= specify for every domain the protocol that must be used for LogicalAddress when registering new SMP, 'all' means that both protocols HTTP and HTTPS are accepted, possible values are [http, https, all].



subdomain.validation.smpLogicalAddressProtocolRestriction.is...	all
subdomain.validation.smpLogicalAddressProtocolRestriction.eh...	http
subdomain.validation.smpLogicalAddressProtocolRestriction.ac...	https

Figure 28 – Defining the accepted protocol

- *PARTICIPANT_ID_REGEXP*= specify for every domain the regular expression that validates the participant ID. By default the regular expression "^.*$" is used.



subdomain.validation.participantIdRegex.isaitb.acc.edelivery...	^.*$
subdomain.validation.participantIdRegex.ehealth.acc.edeliver...	^.*$
subdomain.validation.participantIdRegex.acc.edelivery.tech.e...	^((((0002|0007|0009|0037|0060|0088|0096|0097...

Figure 30 – Defining regular expression for valid participant IDs

NOTE: The values of the properties aforementioned are case insensitive.

15.3. Application server specific configuration

To ensure compatibility with all the supported application servers, some configuration is required.

- For technical reasons, these parameters are not in database but in property file: sml.config.properties. Property file must be located in classpath of application server.
- The sml.config.properties property file contains the following properties:

Property	Example	Description
sml.hibernate.dialect	org.hibernate.dialect.Oracle10gDialect	Hibernate database dialect for accessing the database
sml.datasource.jndi	jdbc/cipaeDeliveryDs	Datasource JNDI name configured on application server.
sml.jsp.servlet.class	weblogic.servlet.JSPServlet	Application server implementation of JSP framework

sml.log.folder	=./logs/	Logging folder.

Example:

```
# *********************************
# Hibernate dialect configuration
# *********************************
# Oracle hibernate example
#sml.hibernate.dialect=org.hibernate.dialect.Oracle10gDialect
# Mysql dialect
sml.hibernate.dialect=org.hibernate.dialect.MySQLDialect

# *********************************
# Datasource JNDI configuration
# *********************************
# weblogic datasource JNDI example
#sml.datasource.jndi=jdbc/cipaeDeliveryDs
# tomcat datasource JNDI example
sml.datasource.jndi=java:comp/env/jdbc/edelivery

# *********************************
# JSP implementation configuration
# *********************************
# Weblogic
#sml.jsp.servlet.class=weblogic.servlet.JSPServlet
# tomcat, jboss
sml.jsp.servlet.class=org.apache.jasper.servlet.JspServlet


# *********************************
# Logging implementation
# *********************************
sml.log.folder=./logs/
```

15.3.1. Weblogic

The file `src/main/webapp/WEB-INF/weblogic.xml` has 3 purposes in the context of the BDMSL:

- Define the context root of the application
- Specify the class loading preferences for some package names (from the weblogic libraries or from the war)
- Configure the work manager to optimize the performance of the application

15.3.2. Tomcat

Tomcat is not an application server because it only supports the servlet API (including JSP, JSTL). An application server supports the whole JavaEE stack.

The file `src/main/webapp/META-INF/context.xml` has 2 purposes:

- Define the context root of the application
- Link the datasource to the globally defined JNDI datasource

16. List of Figures

List of Tables

17. Contact Information

eDelivery Support Team

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

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