



European
Commission

DomSML 5.0

eDelivery

Contents

- 1. Architecture Description 2
- 2. Quick Start Guide 98
- 3. Interface Description 114
- 4. Support 248

Chapter 1. Architecture Description

1.1. Solution Overview

SML was initiated by [PEPPOL](#). 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 \(Business Document Metadata Service Location\)](#).

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

The DomiSML is the sample implementation of the Service Location Architectural Building Block (ABB).

This document is the Software Architecture document of the DomiSML sample implementation. It intends to provide detailed information about the project:

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

▼ Useful References

[eDelivery BDXL - 2.0](#)

eDelivery uses this eDelivery profile of OASIS BDXL 1.0 as its technical specification for Service Location. eDelivery BDX Location specifies a method to query DNS resource records to retrieve a URL for metadata services. The result of a query is a full URI, which can use HTTPS and supports server (and optionally client) authentication.

[OASIS Business Document Metadata Service Location Version 1.0 \(OASIS BDXL 1.0\)](#)

This specification defines the 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 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.

[SML Specification](#)

Defines the profiles for the discovery and management interfaces for the Business Document Exchange Network (BUSDOX) Service Metadata Locator service.

[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.

1.1.1. Solution's Specification

eDelivery's DomiSML is a solution conforming with:

- [eDelivery SML 2.0 specification](#) defining the eDelivery profile of OASIS BDXL 1.0 for Dynamic discovery of the publisher.
- [OpenPeppol SML Specification](#) defining the management application interfaces for SMP to SML component integration.

Service Metadata Locator (SML)

The SML is the only central component of the eDelivery Messaging Infrastructure. It leverages a domain name system (DNS) service for DNS-based lookups to determine the publisher's (SMP) URL address. The DNS query utilizes the hashed value of the recipient's identifier to associate the final recipient of the message with the corresponding publisher's URL address.

Business Document Metadata Service Location (BDXL)

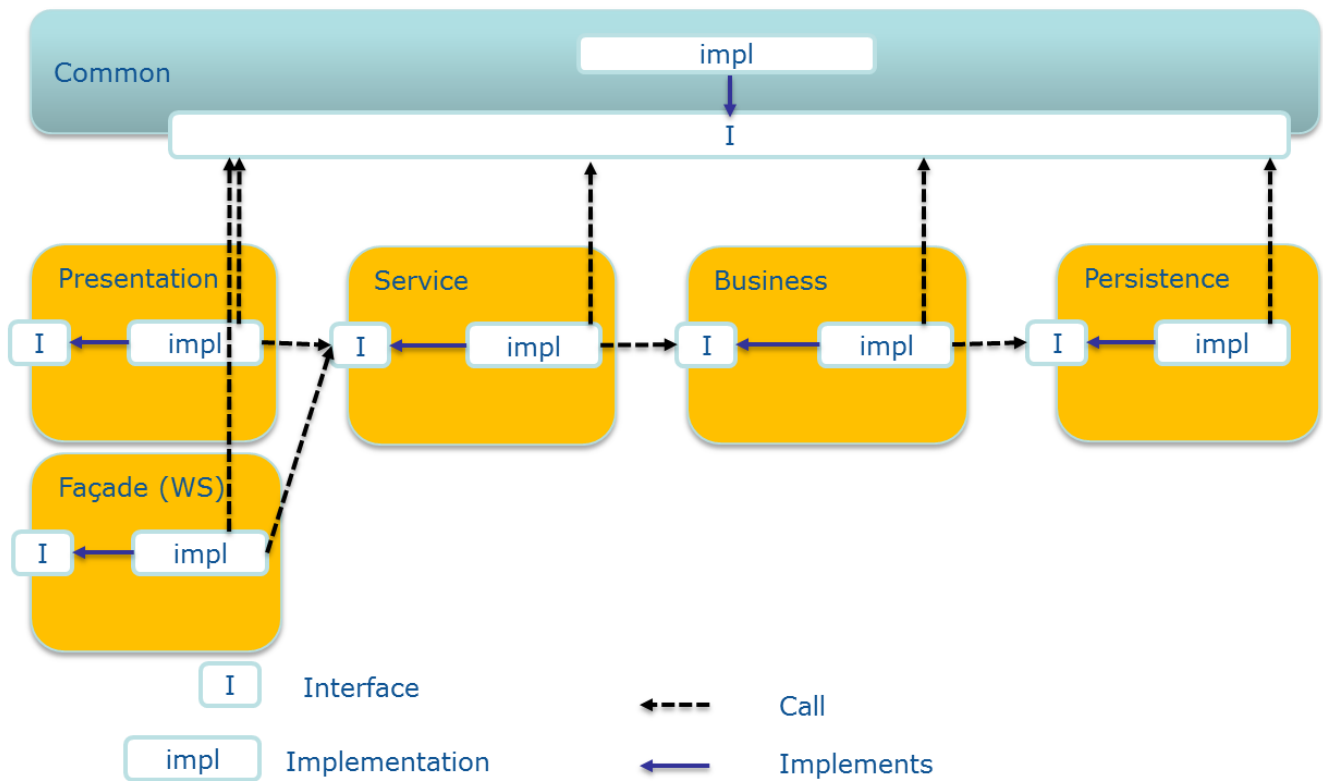
The discovery functionality of SML has been described in a more general technical specification called the [Business Document Metadata Service Location Version \(BDXL\)](#).

This specification is based on DNS, like SML, using the 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.

1.1.2. Solution 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:



1.1.3. 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.

Naming convention

- Controller's java package:
`eu.europa.ec.bdmsl.presentation`
- Implementation:
`eu.europa.ec.bdmsl.presentation.<PageName>Controller`
- JSP files folder:
`src/main/webapp/WEB-INF/jsp`

Dependencies

In this layer, the only allowed calls are the calls to (the):

- Technical components;
- Service's layer;

- Common package.

Frameworks and patterns

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

- Views are represented by JSP files. These files are bound to controllers.
- Models are built from calls to the service layer, and then returned to the Views.
- The Presentation layer includes the controllers.

For example, the `ListDNSController` implementation class:

`ListDNSController`

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

// ...

@Controller
public class ListDNSController {

    private static final Logger LOG = LoggerFactory.getLogger(ListDNSController.class);

    private final IDnsClientService dnsClientService;
    private final IBDMSSLService bdmslService;
    private final IConfigurationBusiness configurationBusiness;

    public ListDNSController(IDnsClientService dnsClientService,
                             IBDMSSLService bdmslService,
                             IConfigurationBusiness configurationBusiness) {
        this.dnsClientService = dnsClientService;
        this.bdmslService = bdmslService;
        this.configurationBusiness = configurationBusiness;
    }

    // Path to the service
    @RequestMapping(path="/listDNS", method = RequestMethod.GET)
    public String listDNS(Model model) {
        LOG.debug("Calling listDNS...");
        String searchType = searchParams.getSearchType();
        // We can add any object in the Model and retrieve them in the Views
        model.addAttribute("searchType", searchType);
        model.addAttribute("dnsEnabled", configurationBusiness.isDNSEnabled());
        if (!configurationBusiness.isDNSEnabled()) {
            model.addAttribute(ERROR, "DNS is not enabled!");
            return VIEW_SEARCH_ERROR;
        }
    }

    // ...
}
```

```

        // bound to listDNS.jsp file in the src/main/webapp/WEB-INF/jsp folder
        return VIEW_LIST_DNS;
    }
}

```

In the `listDNS.jsp` file, the Model can be accessed as in the example below:

`listDNS.jsp` file

```

<%@ 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}">
                {empty}{...}
            </c:when>
            <c:otherwise>
                <ul>
                    <li>The DNS client is disabled.</li>
                </ul>
            </c:otherwise>
        </c:choose>
    </body>
</html>

```

1.1.4. 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 **readOnly = 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.

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**

Dependencies

In this layer, the only calls allowed are the calls to (the):

- Technical components
- Business' layer
- Common package

Frameworks and Patterns

The Service's layer uses these frameworks:

- Spring: transaction management, exception handling, dependency injection.

Service Layer's Development

▼ 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;

    ...
}

}
```

▼ 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 smpBO;
    }

    ...
}
```

```
}
```

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

▼ 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 can 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 DomiSML component, we rollback for any type of exception (checked and runtime), so we set the following value: `rollbackFor = Exception.class`.

1.1.5. Web Service Package

This section 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 DomiSML application are declared in Spring and implement the Singleton pattern.

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.bdms1.ws`.

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. 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. Façade also performs the conversion of the objects from JAXB to BO and vice-versa.

The reference to the underlying service is injected in Façade with Spring.

For each method of the interface implemented by Façade, we invoke the same method as on the service implementation class:

▼ Click to open

```
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 e) {
            // ...
        }
    }
}
```

```

    } catch (Exception exc) {
//      [...]
      handleException(exc);
    }
    return result;
  }
  // ...
}

```

Dependencies

In this layer, the only calls allowed are the calls to (the):

- Technical components
- Service's layer
- Common package

SOAP Web Services

This section describes the general principles governing SOAP web services.

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`

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
// ...
}

```

NOTE

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)`.

Exposed Services Declaration

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">

  {empty}[...]

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

</beans>
```

WSDL

The exposed web services are defined in a contract interface file named WSDL. In eDelivery's DomiSML application, we use a *WSDL-first approach*. This 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 DomiSML application, they are:

- `ManageBusinessIdentifierService-1.0.wsdl` – The services for managing the participant identifiers.
- `ManageServiceMetadataService-1.0.wsdl` – The services for managing the service metadata entry

(SMP).

- **BDMSLSERVICE-1.0.WSDL** – Additional custom or experimental services for managing SMP Metadata and participant entries.
- **BDMSLMonitoringService-1.0.WSDL** – The services for internal monitoring the DomiSML instance.
- **BDMSLAdminService-1.0.WSDL** – The services for managing the DomiSML instance.

IMPORTANT

To enhance security, ensure that the **BDMSLMonitoringService-1.0.WSDL** and **BDMSLAdminService-1.0.WSDL** endpoints are not exposed to the internet. These services are designed for DomiSML management and monitoring purposes only.

These WSDL files are located in the **bdmsl-api/src/main/resources** directory.

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 **bdmsl-api/src/main/resources** directory.

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).

See [Object Mapping](#), for more information on the mapping of JAXB objects to BO.

Frameworks and Patterns

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

Services Specifications

DomiSML's implementation details:

Interface	Description
ManageServiceMetadataService-1.0.WSDL	• See ManageService Metadata Service. • Defined in the PEPPOL SML Specification, in Appendix B: WSDLs.
ManageBusinessIdentifierService-1.0.WSDL	• See Manage Participant Identifier. • Defined in the PEPPOL SML Specification, in Appendix B: WSDLs.

Interface	Description
BDMSLSERVICE-1.0.wsdl	• See BDMSLSERVICE Interface. • 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.
BDMSLMonitoringService-1.0.wsdl	• Contains services for internal monitoring the DomiSML instance.

ManageService Metadata Service

WSDL file

- [ManageServiceMetadataService-1.0.wsdl](#)

▼ Operation [Create\(\)](#)

Description

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

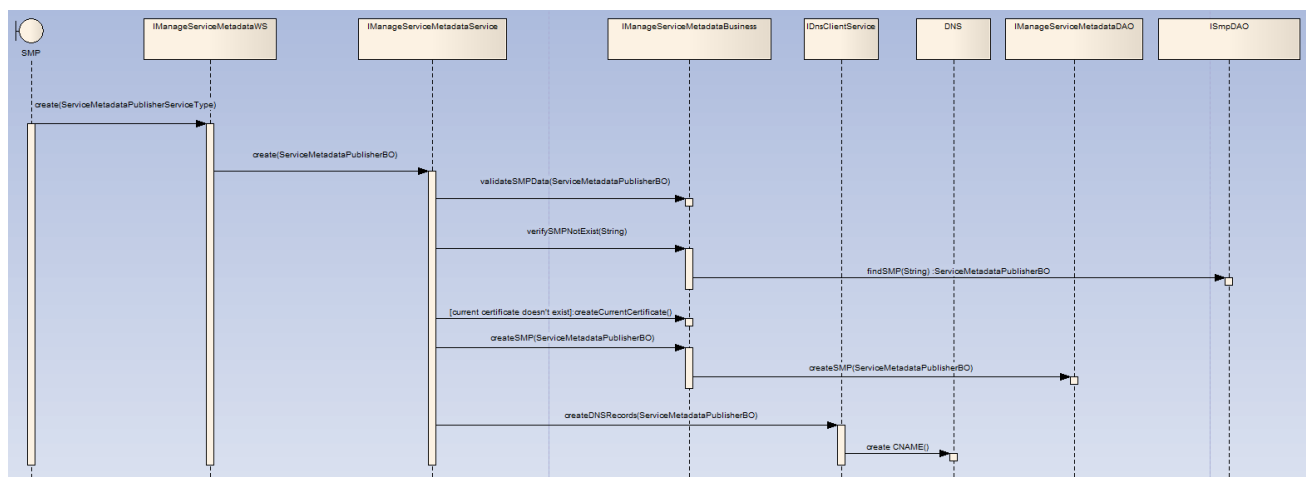


Fig. 1. - ManageService Metadata Service/ Operation [Create\(\)](#).

Pre-requisites

- User has a valid certificate.

NOTE

We consider a certificate is valid when 1) it has not been revoked or 2) has not expired. x

- The role associated with the certificate is [ROLE_SMP](#).
- The SMP does not yet exist in the system.

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(s)

<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the <code>Create</code> operation.
<code>badRequestFault</code>	Returned if the supplied <code>CreateServiceMetadataPublisherService</code> does not contain consistent data.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

▼ Operation `Read()`

Description

Retrieves the SMP record for the service metadata publisher.

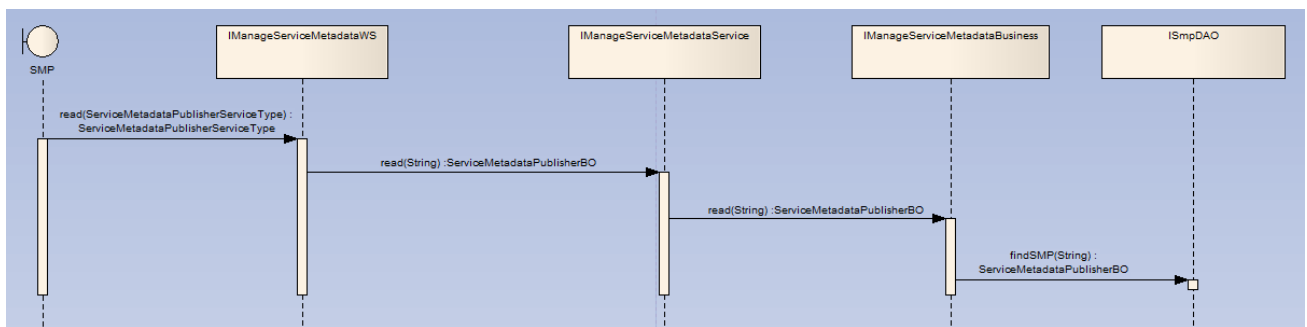


Fig. 2. - ManageService Metadata Service/ Operation `Read()`.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- The SMP already exists.

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(s)

notFoundFault	Returned when the identifier of the SMP cannot be found.
unauthorizedFault	Returned when the caller is not authorized to invoke the Read operation.
badRequestFault	Returned when the supplied parameter does not contain consistent data.
internalErrorFault	Returned when the SML service is unable to process the request for any reason.

▼ *Operation* Update()

Description

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

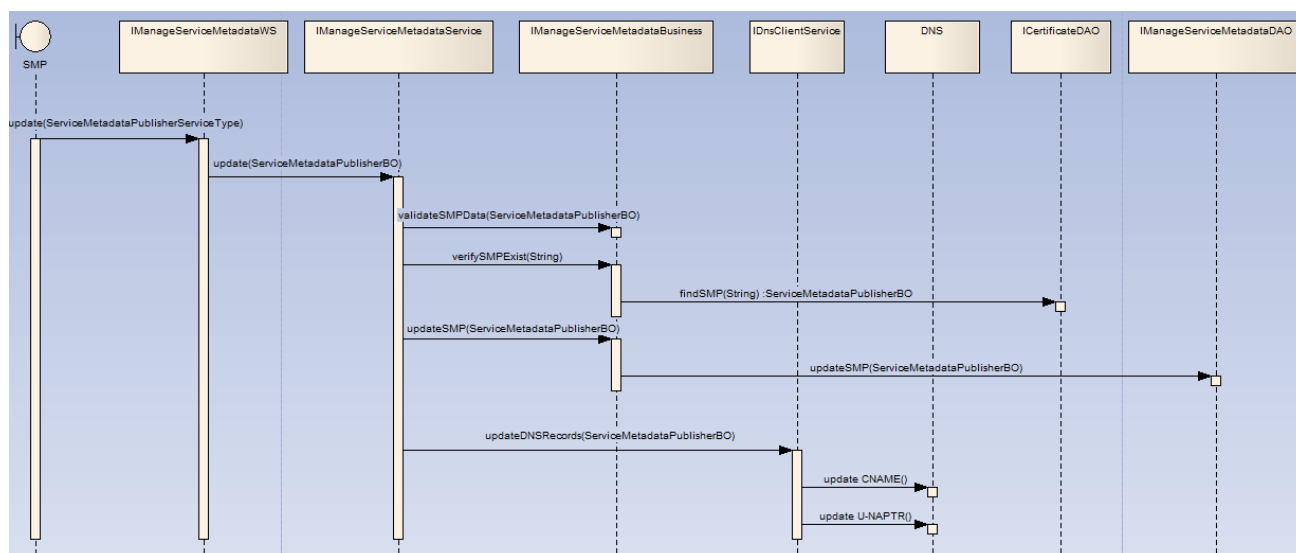


Fig. 3. - ManageService Metadata Service/ Operation Update().

Pre-requisites

- User has a valid certificate.
- User's role is **ROLE_SMP**.
- The SMP already exists.

Input

- **UpdateServiceMetadataPublisherService:**
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(s)

<code>notFoundFault</code>	Returned if the identifier of the SMP could not be found.
<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the Update operation.
<code>badRequestFault</code>	Returned if the supplied <code>UpdateServiceMetadataPublishService</code> does not contain consistent data.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

▼ Operation `Delete()`

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- The SMP already exists.

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(s)

<code>notFoundFault</code>	Returned if the identifier of the SMP could not be found.
<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the Delete operation.
<code>badRequestFault</code>	Returned if the supplied <code>DeleteServiceMetadataPublisherService</code> does not contain consistent data.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

IMPORTANT

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. In fact, the RFC1035 of the DNS standard states that the messages are bound to 65535 bytes length.

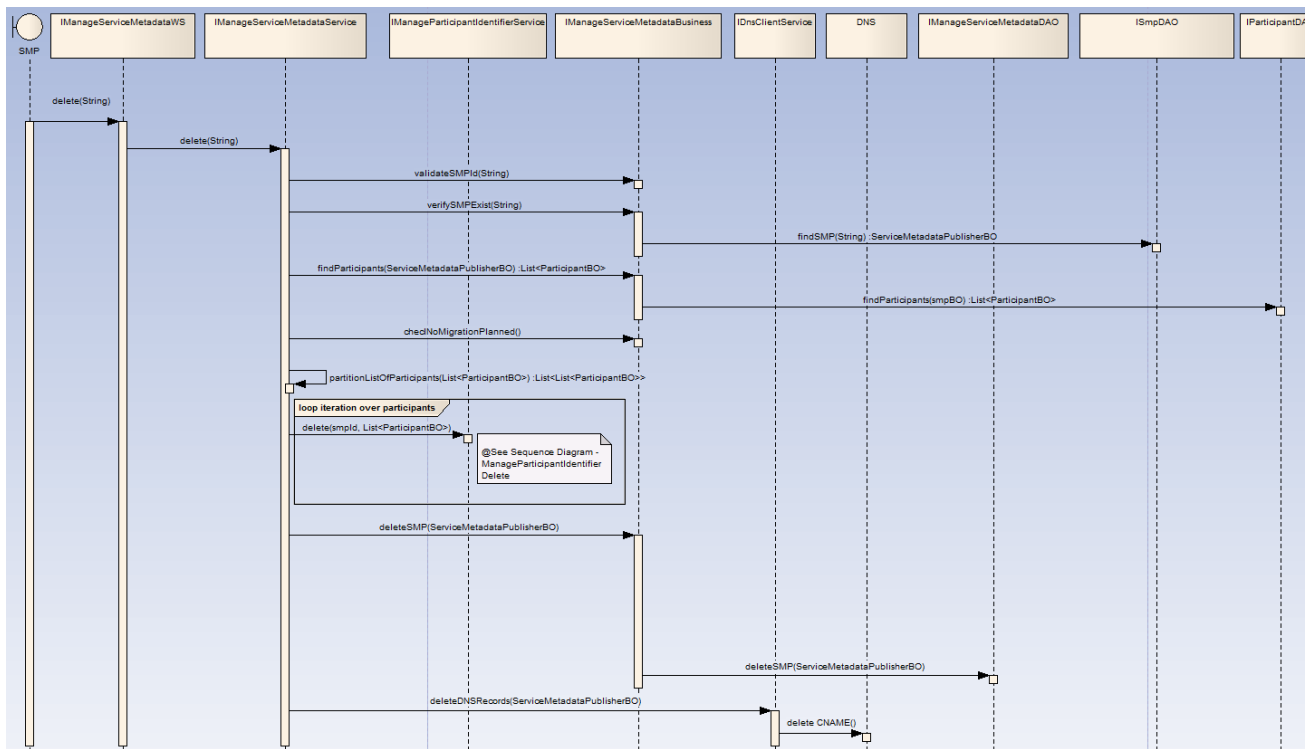


Fig. 4. - ManageService Metadata Service/ Operation Delete().

Manage Participant Identifier

WSDL file

- [ManageBusinessIdentifierService-1.0.wsdl](#)

▼ Operation Create()

Description

This method manages participant entries for the SMP by either creating or updating them. If the participant:

- does not exist → a new entry is created;
- exists but lacks the required DNS record for a specified NAPTR service → the entry is updated accordingly.

- Manage Participant Identifier/ Operation Create().

plantuml::content/_common/diagrams//architecture/sd-create-participant.puml[svg, width=904, align="center"]

Pre-requisites

- User has a valid certificate.
- User's role is **ROLE_SMP**.
- The SMP already exists.
- Participants do not yet exist or its NAPTR record for given service name is not yet created.

Input

- **CreateParticipantIdentifier:**

ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP which holds its data.

Fault(s)

notFoundFault	Returned if the identifier of the SMP could not be found.
unauthorizedFault	Returned if the caller is not authorized to invoke the Create() operation.
badRequestFault	Returned if the supplied CreateParticipantIdentifier does not contain consistent data.
internalErrorFault	Returned if the SML service is unable to process the request for any reason.

▼ Operation **CreateList()**

Description

This method manages the list of participants entries for the SMP by either creating or updating them.

If the participant item from the list:

- does not exist → a new entry is created;
- exists but lacks the required DNS record for a specified NAPTR service → the entry is updated accordingly.

- Manage Participant Identifier/ Operation **CreateList()**.

plantuml::content/_common/diagrams//architecture/sd-create-participants.puml[svg, width=904, align="center"]

Pre-requisites

- User has a valid certificate.
- User's role is **ROLE_SMP**.
- The SMP already exists.
- The participants do not yet exist or its NAPTR record for given service name is not yet created.
- Listed participants are less than 100.

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(s)

notFoundFault	Returned if the identifier of the SMP could not be found.
unauthorizedFault	Returned if the caller is not authorized to invoke the CreateList() operation.
badRequestFault	Returned if: • The supplied CreateList does not contain consistent data. • The number of participants in the list is greater than 100.
internalErrorFault	Returned if the SML service is unable to process the request for any reason.

▼ Operation **Delete()**

Description

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

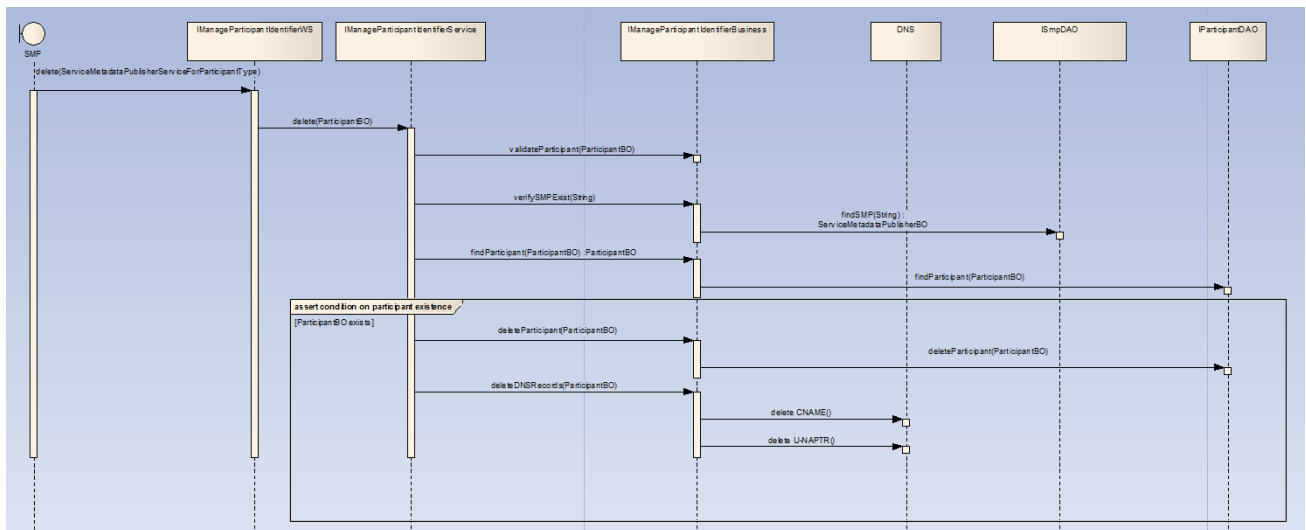


Fig. 5. - Manage Participant Identifier/ Operation **Delete()**.

Pre-requisites

- User has a valid certificate.
- User's role is **ROLE_SMP**.
- The SMP already exists.
- The participants already exist with required DNS naptr service.
- Participants are not under migration process and do not have an active migration key.

Input

- **DeleteParticipantIdentifier:**
ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP that publishes its metadata.

Fault(s)

<code>notFoundFault</code>	Returned if the identifier of the SMP could not be found.
<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the <code>Delete()</code> operation or if the migration to another SMP is in progress.
<code>badRequestFault</code>	Returned if the supplied <code>DeleteParticipantIdentifier</code> does not contain consistent data.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

▼ Operation `DeleteList()`

Description

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

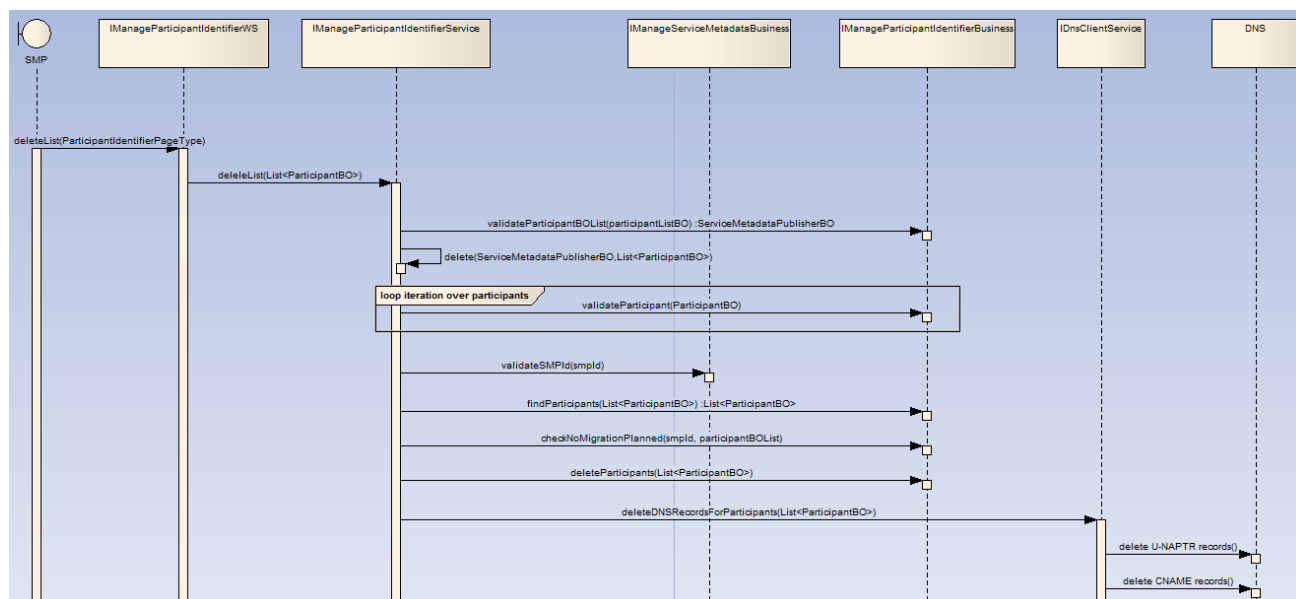


Fig. 6. - Manage Participant Identifier/ Operation `DeleteList()`.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- The SMP already exists.
- The participants already exist with required DNS naptr service.
- Participants are not under migration process.
- Listed participants are less than 100.

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(s)

notFoundFault	Returned if one or more participant identifiers or the identifier of the SMP could not be found.
unauthorizedFault	Returned if the caller is not authorized to invoke the DeleteList() operation or if one of the participants is under migration process.
badRequestFault	Returned if: - The supplied DeleteList does not contain consistent data. - The number of participants in the list is greater than 100.
internalErrorFault	Returned if the SML service is unable to process the request for any reason.

▼ Operation **PrepareToMigrate()**

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.

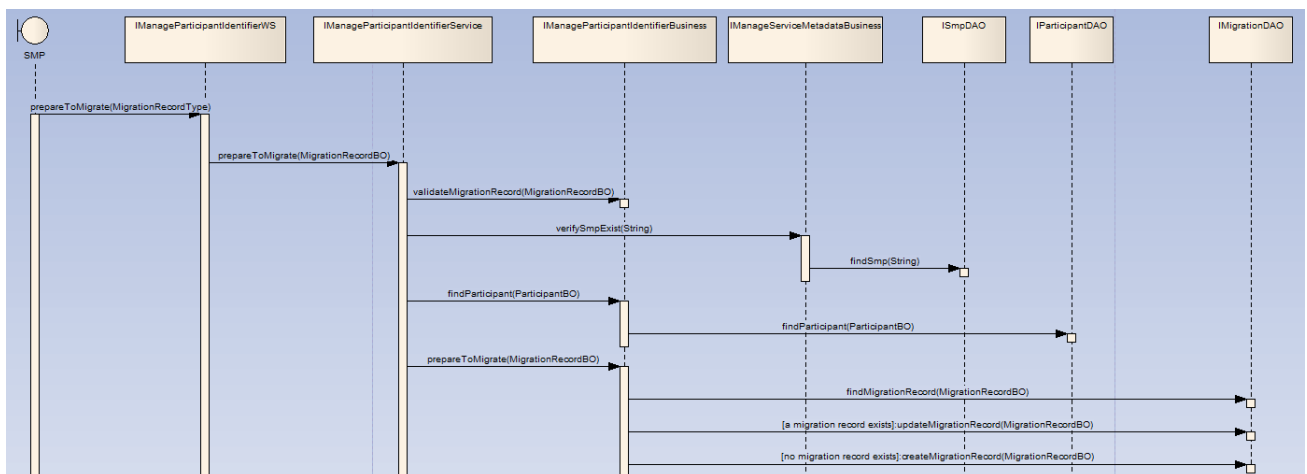


Fig. 7. - Manage Participant Identifier/ Operation **PrepareToMigrate()**.

Pre-requisites

- User has a valid certificate.
- User's role is **ROLE_SMP**.
- The SMP already exists.

- The participants already exist.

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(s)

notFoundFault	Returned if one or more participant identifiers or the identifier of the SMP could not be found.
unauthorizedFault	Returned if the caller is not authorized to invoke the PrepareToMigrate() operation.
badRequestFault	Returned if the supplied PrepateMigrationRecord does not contain consistent data.
internalErrorFault	Returned if the SML service is unable to process the request for any reason.

▼ Operation **Migrate()**

Description

Migrates a Participant Identifier already held by the Service Metadata Locator Service to target a new SMP. 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 SMP.

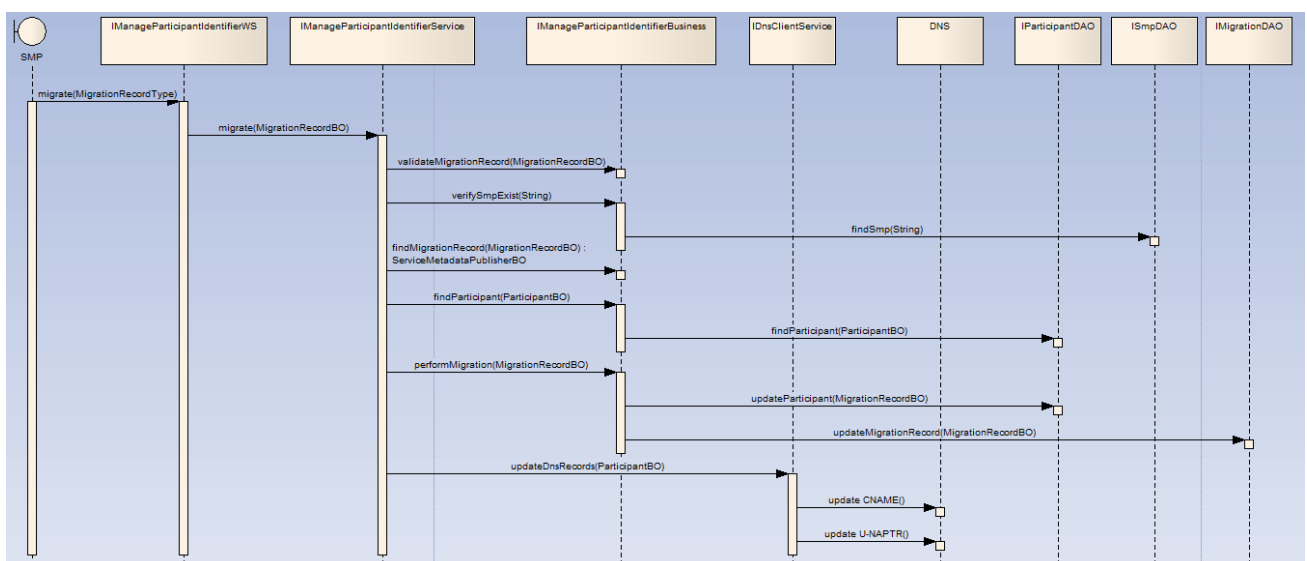


Fig. 8. - Manage Participant Identifier/ Operation **Migrate()**.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- The SMP already exists.
- The participants already exist.
- The `prepareToMigrate` service has been called for participant in question.

Input

- `CompleteMigrationRecord`:
`MigrationRecordType` - contains the Migration Key and the Participant Identifier which is to be migrated from one SMP to another.

Fault(s)

<code>notFoundFault</code>	Returned if the migration key or the identifier of the SMP could not be found.
<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the <code>Migrate()</code> operation.
<code>badRequestFault</code>	Returned if the supplied <code>CompleteMigrationRecord</code> does not contain consistent data.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

▼ Operation `List()`

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- The SMP already exists.

Description

`List()` is used to retrieve a list of all participant identifiers associated with a single SMP 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.

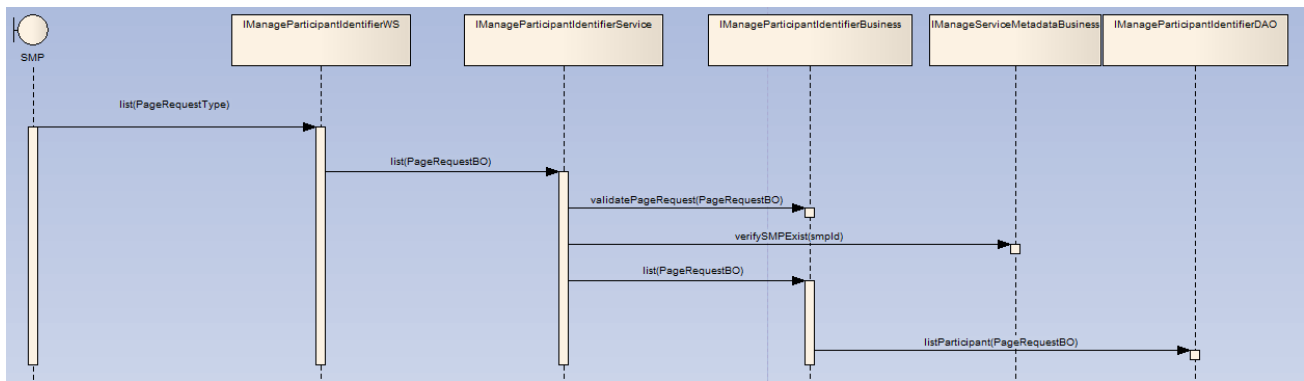


Fig. 9. - Manage Participant Identifier/ Operation List().

Input

- 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 SMP, also containing a **<Page/>** element containing the identifier that represents the next page, if any.

Fault(s)

notFoundFault	Returned if the next page or the identifier of the SMP could not be found.
unauthorizedFault	Returned if the caller is not authorized to invoke the List() operation.
badRequestFault	Returned if the supplied NextPage does not contain consistent data.
internalErrorFault	Returned if the SML service is unable to process the request for any reason.

NOTE 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.

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.

WSDL file

- **BDMSLService-1.0.wsdl**

▼ Operation `PrepareChangeCertificate()`

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.

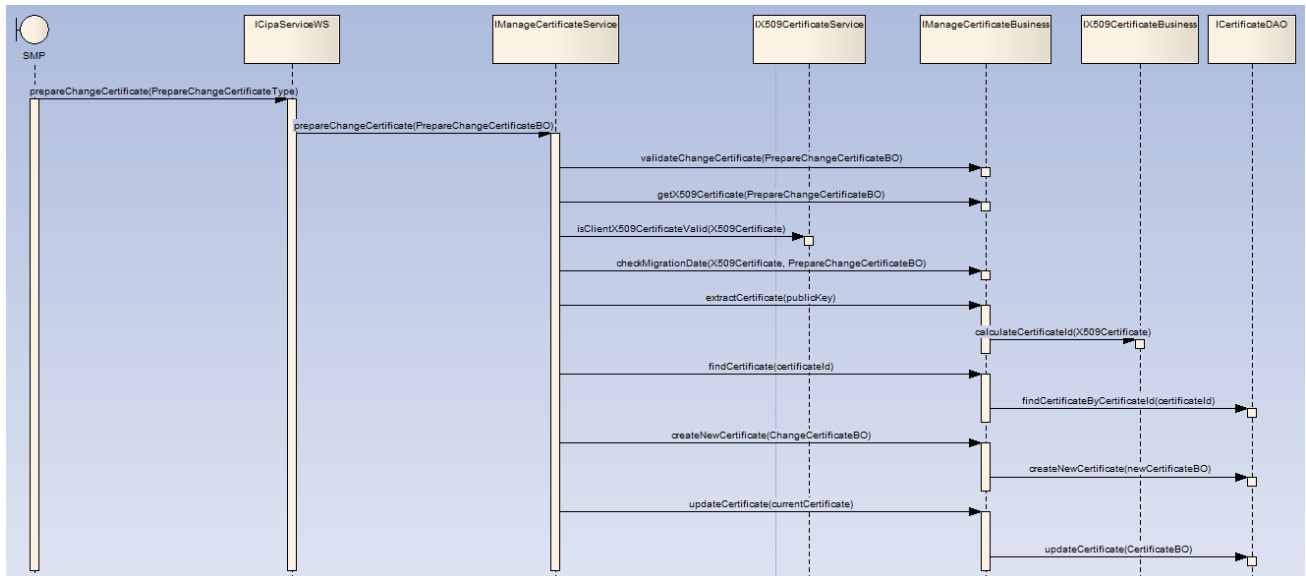


Fig. 10. - BDMSLService interface/ Operation `PrepareChangeCertificate()`.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP`.
- User has the new certificate for the SMP(s).

Fault(s)

<code>notFoundFault</code>	Returned if the next page or the identifier of the SMP could not be found.
<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the <code>PrepareChangeCertificate</code> operation.

<code>badRequestFault</code>	Returned if • The supplied request does not contain consistent data • The new certificate is not valid at the date provided in the <code>migrationDate</code> element • The <code>migrationDate</code> is not: ◦ in the future. ◦ provided and the "Not Before" date of the new certificate is not in the future. ◦ provided and the "Valid From" is in the past
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

NOTES

A nightly job performs an analysis to actually perform the change of certificates.

The algorithm is as following (in pseudocode):

```
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`.

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.

▼ Operation `CreateParticipantIdentifier()`

Description

This service has the same behaviour as the `Create()` operation in the `ManageParticipantIdentifier` interface but it has one additional input which is optional: 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.

Pre-requisites

- User has a valid certificate.
- The SMP already exists.
- The participant does not yet exist or its NAPTR record for given service name is not yet created.

- *BDMSLService interface/ Operation `CreateParticipantIdentifier()`.*

plantuml::content/_common/diagrams//architecture/sd-bdmslservice-create-participant.puml[svg, width=903, align="center"]

Input (optional)

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

TIP

The flow for the create method of `ManageParticipantIdentifierServiceImpl` can be found in [Technical Design](#).

BDMSLMonitoringService interface

This interface describes monitoring services to allow operators of the SML service to check the health status of the DomiSML instance. The services are restricted only for role `ROLE_MONITOR`.

▼ *Operation `IsAlive()`*

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.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_MONITOR`.

Input

None.

Output

None.

HTTP 200 OK expected.

Fault(s)

<code>internalErrorFault</code>	Returned if the BDMSL service is unable to process the request for any reason.
---------------------------------	--

BDMSLAdminService interface

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

▼ Operation `ClearCache()`

Description

The application manages in-memory caches to enhance performance. 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, to avoid the cost of downloading each time the CRLM for each certificate.

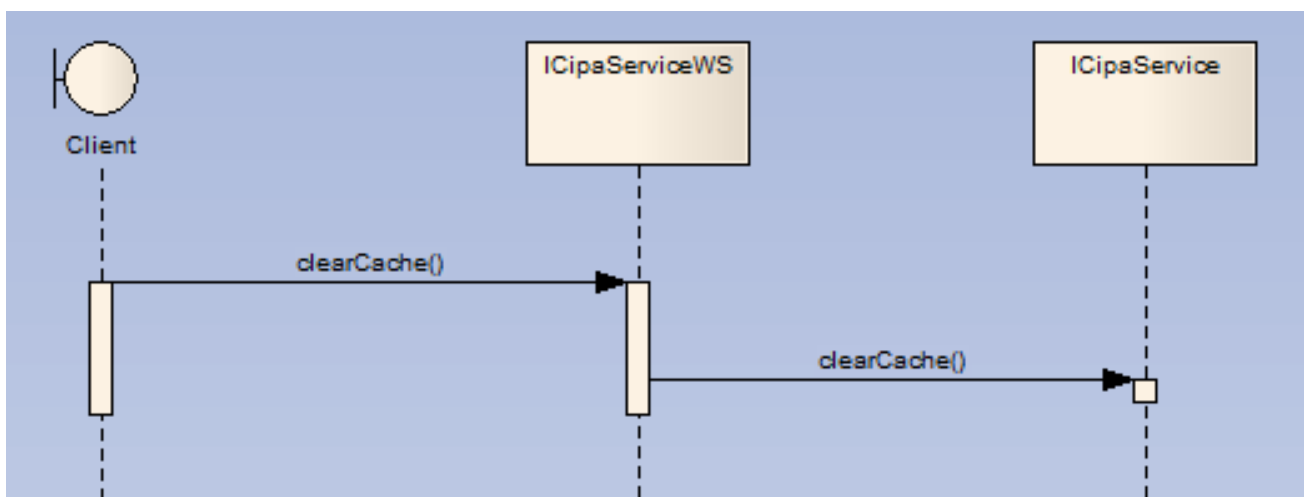


Fig. 11. - BDMSLAdminService interface/ Operation `ClearCache()`.

Pre-requisites

- User has a valid certificate.
- User's role is `ROLE_SMP` or `ROLE_ADMIN`.

Input

None.

Output

None.

HTTP 200 OK expected.

Fault(s)

internalErrorFault	Returned if the SML service is unable to process the request for any reason.
---------------------------	--

▼ Operation **ChangeCertificate()**

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.

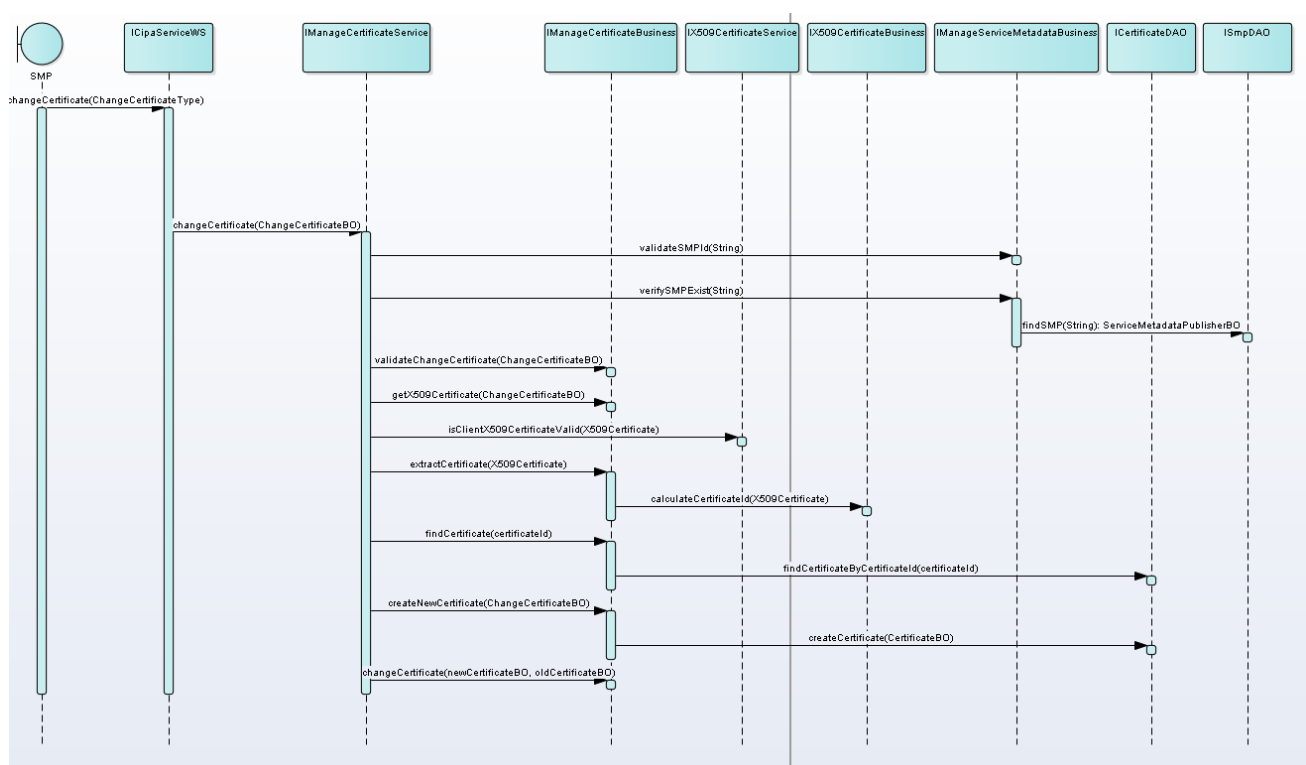


Fig. 12. - BDMSLAdminService interface/ Operation **ChangeCertificate()**.

Pre-requisites

- Certificate is valid.
- User's role is **ROLE_ADMIN**.
- User has the new certificate for the SMP.

Input

- SMP id, Certificate public key

Output

None.

HTTP 200 OK expected

.Fault(s)

<code>unauthorizedFault</code>	Returned if: • The caller is not authorized to invoke the <code>ChangeCertificate</code> operation (ie. the user doesn't have the <code>ROLE_ADMIN</code> role). • The public key already exists
<code>BadRequestFault</code>	Returned if: • Supplied request does not contain consistent data. • Invalid public key. • New certificate is not valid at the moment of request. • SMP ID is unknown.
<code>internalErrorFault</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `SetProperty()`

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's properties are only refreshed with the restart of the BDMSL server.

Pre-requisites

- User's role is `ROLE_ADMIN`.

Input

- Property name, Property value,
- Optional input(s): Property description.

Output

- Property data stored to BDMSL database.

Fault(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
--------------------------------	---

BadRequestFault	Returned if: • The supplied request does not contain consistent data. • Invalid propertyName.
internalErrorFault	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **GetProperty()**

Description

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

Pre-requisites

- User's role is **ROLE_ADMIN**.

Input

- Property name

Output

- Property data stored to BDMSL database

Fault(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if "Invalid `propertyName`".
internalErrorFault	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **DeleteProperty()**

Pre-requisites

- User's role is **ROLE_ADMIN**.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if "Invalid `propertyName`".
<code>internalErrorFault</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `CreateSubDomain()`

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.

Pre-requisite

- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned when the certificate provided is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if "Invalid <code>SubDomain</code> data".

▼ Operation `UpdateSubDomain()`

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`.

Pre-requisites

- User's role is **ROLE_ADMIN**.

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(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if "Invalid SubDomain data".
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **GetSubDomain()**

Description

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

Pre-requisites

- User's role is **ROLE_ADMIN**.

Input

- SubDomain name.

Output

- Subdomain data stored to BDMSL database.

Fault(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if "Invalid SubDomain data".
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `DeleteSubDomain()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

Description

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

Input

- SubDomain name.

Output

- Deleted Subdomain data from BDMSL database

Fault(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if <code>Subdomain</code> has registered participants.
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `AddSubDomainCertificate()`

Pre-requisites

- Certificate is valid.
- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if "Invalid Subdomain Certificate data".
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `UpdateSubDomainCertificate()`

Pre-requisites

- The certificate is already added to database.
- User's role is `ROLE_ADMIN`.

Description

This operation allows the admin team to update SubDomain certificate data. Admins 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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if "Invalid Subdomain Certificate data".
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `ListSubDomainCertificate()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `DeleteSubDomainCertificate()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

Description

This operation enables the Admin team to delete Domain certificates, with the exception of the certificate used by the Admin(s) for processing the request in progress, which cannot be deleted.

The request removes the certificate from the `CertificateDomain` table. To edit the truststore, use the `DeleteTruststoreCertificate` operation.

Input

- `SubDomain` name and `CertificateId` to be deleted.

Output

- Result of the deletion.

Fault(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if "Invalid Subdomain Certificate data".
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `AddDNSRecord()`

Pre-requisites

- BDMSL is integrated with the DNS server.
- User's role is `ROLE_ADMIN`.

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(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if: • Invalid DNS name; • Invalid DNS value; • Invalid DNS record type.
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation DeleteDNSRecord()

Pre-requisites

- BDMSL is integrated with the DNS server.
- User's role is **ROLE_ADMIN**.

Description

This operation allows the admin team to remove records from DNS server.

Various 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(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if "Invalid DNS record name".
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **AddTruststoreCertificate()**

Pre-requisites

- Certificate is valid.
- User's role is **ROLE_ADMIN**.

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(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if: • Invalid Certificate data. • Certificate already exists in truststore. • Certificate with given alias already exists in truststore.
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **GetTruststoreCertificate()**

Pre-requisites

- The certificate with given alias is already added to the truststore.
- User's role is **ROLE_ADMIN**.

Description

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

Input

- Truststore alias.

Output

- X509Certificate data.

Fault(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if alias is not given in parameter.
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `DeleteTruststoreCertificate()`

Pre-requisites

- Certificate with provided alias is in the truststore.
- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if alias is not given in parameter.
<code>NotFoundFault</code>	Returned if alias is not present in truststore.
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `ListTruststoreCertificateAliases()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>InternalFaultError</code>	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation `GenerateInconsistencyReport()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

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(s)

<code>UnauthorizedFault</code>	Returned if the provided certificate is not an ADMIN certificate.
<code>BadRequestFault</code>	Returned if email is invalid.

InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.
---------------------------	--

▼ Operation **GenerateReport()**

Description

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

Pre-requisites

- User's role is **ROLE_ADMIN**.

Input

- Email for receiving the report. Report type.

Output

- Email with the report.

Fault(s)

UnauthorizedFault	Returned if the provided certificate is not an ADMIN certificate.
BadRequestFault	Returned if email or report type are invalid.
InternalFaultError	Returned if the BDMSL service is unable to process the request for any reason.

▼ Operation **ManageServiceMetadataPublisher()**

Description

This operation allows the admin team to manage (read, enable, disable, delete, and update) SMP entries. Actions are executed asynchronously and the email is sent when completed. The only synchronous action is the read action which returns the SMP data in the SOAP response.

The method's asynchronous execution is well-suited for managing SMPs with numerous participants, and the SMP size limitations are not applicable to this method.

Pre-requisites

- User's role is **ROLE_ADMIN**.

Input

- **SMP identifier:** The SMP identifier to be managed.
- **Action type:**
with following options:

- **READ**: The current SMP data (logical/physical address, registered certificate, domain etc.) is returned in the SOAP response.
 - **'ENABLE'**: Enables the SMP and register all DNS records if the DNS integration is enabled. The action is executed asynchronously when finished the email is sent.
 - **DISABLE**: Disables the SMP and removes all DNS records from DNS if the DNS integration is enabled. The action is executed asynchronously when finished the email is sent.
 - **DELETE**: Deletes the SMP with all registered participants from Database. The SMP must be in DISABLED state.
The action is executed asynchronously; once finished, the email is sent.
 - **UPDATE**: Updates smp logical and physical address and updates the DNS records if the DNS integration is enabled.
The action is executed asynchronously; once finished, the email is sent.
 - **SYNC_DNS**: Synchronizes SMP records in the DNS server with the DomiSML database. This process updates the CNAME and NAPTR records based on the subdomain/SMP DNS record types (CNAME/NAPTR) and the participant NAPTR service configurations.
The synchronization occurs asynchronously, and an email notification is sent upon completion.
When the SMP ID is set to **SYNC_ALL_DOMAIN_SMP**, the action is performed for all SMPs. This command is particularly useful when the subdomain record type has been modified, ensuring that records are synchronized SMP by SMP. If the process is interrupted (e.g., due to a server restart), the next execution will resume synchronization from the SMPs not yet updated.
- **Email address**: Email receiving the action result. Optional for **READ** actions.
 - **DNS Domain**: The domain name where SMP is registered. Optional for **READ** actions.
 - **Owner Certificate Id**: The certificate id of the SMP owner. Optional for **READ** actions.
 - **Logical and/or physical address of the SMP**: Mandatory to provide at least one address for **UPDATE** actions.

Output

- SOAP response with action's status.
- Email with action's result.

Fault(s)

UnauthorizedFault	Returned when the certificate provided is not an ADMIN certificate.
BadRequestFault	Returned when email or other data are invalid.
InternalFaultError	Returned if the DomiSML service is unable to process the request for any reason.

▼ Operation `ManageParticipantIdentifier()`

Pre-requisites

- User's role is `ROLE_ADMIN`.

Description

This operation allows the Admin team to create or delete participants on behalf of publisher owners.

It is designed to help Admins manage participants in situations such as:

- the publisher' owner is unable to access or manage their own participant entries;
- the SMP is no longer operational;
- the ecosystem owner requests participant management for the publisher owner.

plantuml:content/_common/diagrams//architecture/sd-delete-sub-domain-certificate.puml[svg, width=604, align="center"]

Input

- **Action type:** with following values:
 - `CREATE`: Create new participant identifier for given publisher identifier.
 - `DELETE`: Delete participant identifier from the given publisher identifier.
- **Publisher Identifier:** The identifier of the publisher for which the participant identifier is created or deleted.
- **Participant Identifier:** The list of identifiers to be created or deleted.
- **Owner Certificate Id:** The certificate id of the SMP owner.

Output

- The action results.

Fault(s)

<code>unauthorizedFault</code>	Returned if the caller is not authorized to invoke the <code>Create</code> operation.
<code>badRequestFault</code>	Returned if provided data is invalid.
<code>internalErrorFault</code>	Returned if the SML service is unable to process the request for any reason.

1.1.6. 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 the [Business Objects \(BO\)](#) section.

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`

Dependencies

In this layer, the only calls allowed are the calls to (the):

- Technical components;
- Persistence's layer;
- Common package.

Frameworks

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

Business Layer's Development

▼ *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);
}
```

▼ *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;
}
//...
}

```

1.1.7. 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.

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`

Dependencies

In this layer, the only calls allowed are the calls to (the):

- Technical components
- Common package

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.

Data Access Layer's Development

▼ 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;

}
```

▼ 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;
    }
    //...
}
```

▼ 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 mapping BO/Entities, see [Object Mapping](#).

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

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`

Dependencies

In this layer, the only calls allowed are the calls to (the):

- Technical components

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. So, 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.

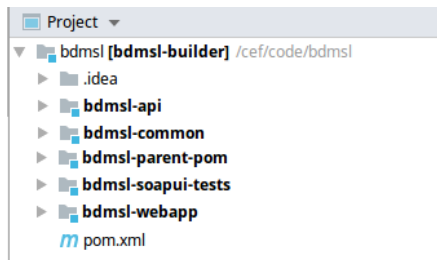
Each BO must define a `serialVersionUID` and implement the three previous methods.

1.1.8. Software Architecture

The development of DomiSML involves five different maven projects.

- `bdmsl-api`
- `bdmsl-common`
- `bdmsl-webapp`
- `bdmsl-parent-pom`

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



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.

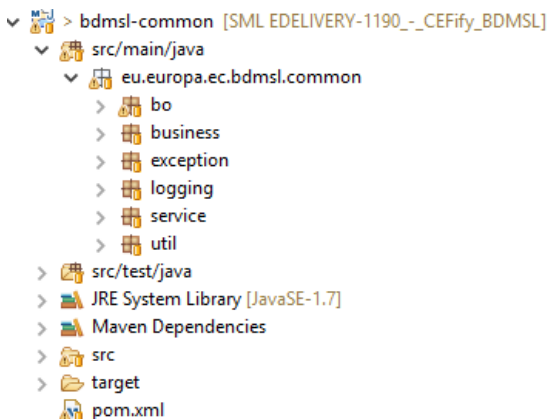
The WSDL files contain all the methods that are exposed, the objects and the exceptions.

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

Library "bdmsl-common"

Project name : bdmsl-common

Packages:



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

bdmsl-webapp

Project name : bdmsl-webapp

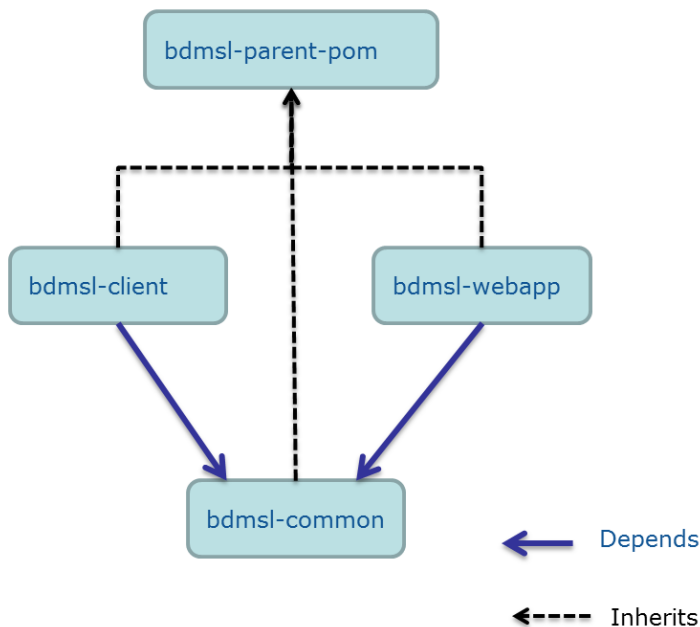
This is a Maven project that contains all the services, business logic and persistence code of the application. It produces a war file that can be deployed in the supported application servers and servlet containers.

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.

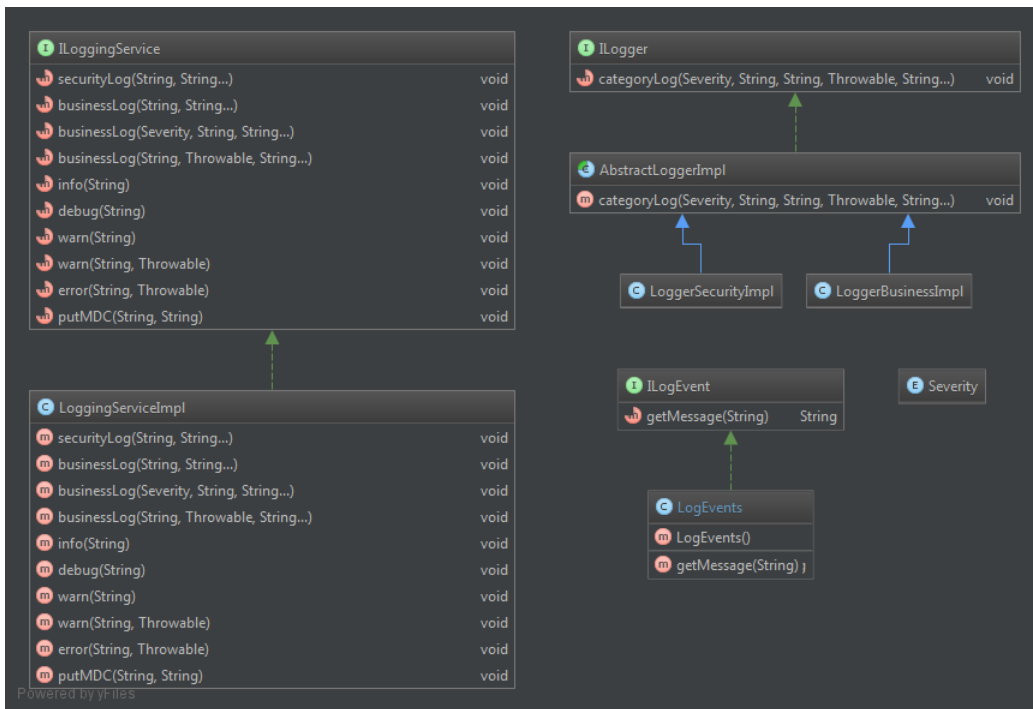
Maven configuration



1.2. Logging

1.2.1. 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.



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 participant 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).

1.2.2. Log event codes

Category	Log event code	Description
SECURITY	SEC-001	The host %s attempted to access %s without any certificate

Category	Log event code	Description
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.

Category	Log event code	Description
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

1.3. Caching

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()`.

1.4. Exception handling

1.4.1. 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.

1.4.2. 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`
- `InternalServerError`

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 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 `IErrCodes` interface (see the table below).

1.4.3. Error Codes

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

1.5. Object mapping

There are three 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.

Two types of mapping are required:



The mapping is done using the spring convert service

`org.springframework.core.convert.ConversionService`

and implementing the functional interface

`org.springframework.core.convert.converter.Converter`

An example of mapping would be:

```

@Component
public class ParticipantB0ToParticipantIdentifierTypeConverter extends
    AutoRegisteringConverter<ParticipantB0, ParticipantIdentifierType> {

    public ParticipantB0ToParticipantIdentifierTypeConverter(ConversionService
conversionService) {
        super(conversionService);
    }

    @Override
    public ParticipantIdentifierType convert(ParticipantB0 source) {
        if (source == null) {
            return null;
        }

        ParticipantIdentifierType target = new ParticipantIdentifierType();
        BeanUtils.copyProperties(source, target);
        target.setValue(source.getParticipantId());
        target.setScheme(source.getScheme());

        return target;
    }
}
  
```

The abstract class `eu.europa.ec.bdmsl.conversion.AutoRegisteringConverter` autoregister the spring `Converter` instance.

```

public abstract class AutoRegisteringConverter<S, T> implements Converter<S, T> {
  
```

```

private ConversionService conversionService;

public AutoRegisteringConverter(ConversionService conversionService) {
    this.conversionService = conversionService;
}

public ConversionService getConversionService() {
    return conversionService;
}

@PostConstruct
private void register() {
    if (conversionService instanceof GenericConversionService) {
        ((GenericConversionService) conversionService).addConverter(this);
    }
}
}

```

The converter is utilized in code by simply calling the Sprint Convert service, as demonstrated in the following example:

```

@Repository
public class ParticipantDAOImpl extends AbstractDAOImpl implements IParticipantDAO {
    @Autowired
    private ConversionService conversionService;

    @Override
    public void createParticipant(ParticipantB0 participantB0) throws
    TechnicalException {
        // map the business object to a JPA entity
        ParticipantIdentifierEntity participantIdentifierEntity = conversionService
        .convert(participantB0, ParticipantIdentifierEntity.class);
        super.persist(participantIdentifierEntity);
    }
}

```

1.6. Database management

1.6.1. Auditing

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'.

1.6.2. 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 with target filename base for e.g.
          org.hibernate.dialect.MySQL5InnoDBDialect:mysql5innodbwill
          generate mysql5innodb.ddl and mysql5innodb-drop.ddl
          2. script version
          3. export scripts.-->
          <java
            classname="eu.europa.ec.bdmsl.dao.utils.SMLSchemaGenerator"
            fork="true"
            failonerror="true">
              <arg value="org.hibernate.dialect.MySQL5InnoDBDialect:mysql,
org.hibernate.dialect.OracleDialect:oracle" /> ①
              <arg value="${project.version}" />
              <arg value="${project.basedir}/src/main/sml-setup/database-
scripts" />

              <arg value="${hibernate.version}" />
              <!-- Reference to the passed-in classpath reference -->
              <classpath refid="maven.compile.classpath" />
            </java>
          </target>
        </configuration>
      </execution>
    </executions>
  </plugin>
```

Where:

- ① Lists the database dialects.

By default, the ddl scripts for `OracleDialect` and `MySQLDialect` are generated in the `bdmsl-webapp/src/main/sml-setup/database-scripts` folder, named as:

- `mysql.ddl` and `mysql-drop.ddl` for MySQL;
- `oracle.ddl` and `oracle-drop.ddl` for Oracle.

To add a new database dialect, you need to:

- add the dialect name;
- the target filename base;

in the `<arg>` tag, which takes a comma separated list of dialects as value.

Example

```
<arg value="org.hibernate.dialect.MySQL5InnoDBDialect:mysql,
org.hibernate.dialect.OracleDialect:oracle"/>.
```

TIP

For database schema details, see the [Hibernate documentation](#).

Database Entity Relationship Diagram

Database tables plantuml:content/_common/diagrams//architecture/er-database-model.puml[svg, width=903, align="center"]

Database Audit tables plantuml:content/_common/diagrams//architecture/er-database-model-aud.puml[svg, width=903, align="center"]

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 <i>wildcard</i> CNAME record in the DNS, along the lines: .<schemeID>.<SML domain> CNAME <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.

Detailed description of the tables

Table	Column	Description
bdmsl_allowed_wildcard	scheme	The scheme on which the wildcard applies

Table	Column	Description
	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 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

Table	Column	Description
	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
	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

Table	Column	Description
	scheme	The scheme of the participant identifier
	naptr_services	The comma separated list of NAPTR services (e.g.: <code>Meta:SMP,oasis-bdxc-smp-2,meta:cipa3</code>). If the value is null or empty, the <code>Meta:SMP</code> service is used.
	disabled	Disabled status of the participant: true/false. Disabled participant identifiers do not have DNS records.
	naptr_hash	Hash of participant used for naptr record
	cname hash	Hash of participant used for cname record
	created_on	Date of creation
	last_updated_on	Date of the last update
bdmsl_smp	fk_smp_id	The foreign key to the SMP identifier
	id	Surrogate key of smp
	smp_id	The SMP identifier
	dns_record_types	The DNS record types created for a participant (e.g., <code>CNAME</code> , <code>NAPTR</code> , or <code>ALL</code>) typically align with the subdomain DNS record type. However, if the value is null, the subdomain's DNS record type is assumed. While the record type usually matches the subdomain's DNS record type, it may differ in cases where the subdomain transitions to a different type of DNS records.
	smp_disabled	Disabled status of the smp: true/false. Disabled SMPs do not have DNS records.
	fk_certificate_id	The foreign key to the certificate

Table	Column	Description
	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
	fk_subdomain_id	The foreign key to the subdomain
	created_on	Date of creation
	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.

1.7. 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 {

```

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.

1.7.1. 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.

1.7.2. 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
<code>sml.property.refresh.cronJobExpression</code>	<code>0 53 */1 * * *</code>	Property refresh cron task expression (every hour, 7 minutes before the hour)
<code>sml.cluster.enabled</code>	<code>false</code>	Property defines if SML is running in cluster mode.

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

1.7.3. 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
<code>dataInconsistencyAnalyzer.cronJobExpression</code>	<code>0 0 3 ? * *</code>	Cron expression: 0 0 3 ? * * (every day at 3:00 am)
<code>dataInconsistencyAnalyzer.recipientEmail</code>	email@example.com	Email address to receive Data Inconsistency Checker results
<code>dataInconsistencyAnalyzer.senderEmail</code>	email@example.com	Sender email address for reporting Data Inconsistency Analyzer.

BDMSL_CONFIGURATION		
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().

1.7.4. 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.

BDMSL_CONFIGURATION		
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().

1.8. 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
	NOTE Set as semicolon(;) separated list/one string.	

1.9. Participant Identifiers

1.9.1. Participant Identifier Uniqueness

A *participant identifier* is a unique identifier assigned to a party within a business network or ecosystem. It serves to identify the party and locate the connectivity metadata (URL, X509Certificates) it has published.

Typically represented as an *hash value* of the participant identifier, it is registered in the *DNS server* and used to resolve the address of the party's metadata publisher (e.g., a Service Metadata Publisher (SMP)).

The publisher service provides the necessary connectivity metadata for interactions within the ecosystem.

The participant identifier can be assigned by an *ecosystem operator*, ensuring uniqueness across the network. Alternatively, *existing identifiers*, already assigned to the party by a third party (e.g., a government authority) may be used.

A non-exhaustive examples list of participant identifiers for businesses includes:

- **VAT Number** – Assigned by tax authorities for companies engaging in value-added tax transactions.
- **Company Registration Number (CRN)** – Issued by government authorities upon official business registration.
- **EORI Number** – Used for businesses participating in international trade within the European Union.
- **Global Location Number (GLN)** – A unique identifier for legal entities and locations, commonly used in supply chain management.
- **Custom Identifier** – A proprietary identifier specific to a particular business network.
- etc.

Scope and Identifiers Uniqueness

Most identifiers assigned to companies by government authorities ensure uniqueness only at national level.

For example, a VAT number is unique within the country where it is issued. On a international scope, to keep a VAT number unique, it is prefixed by an unique code representing the country.

Interoperable systems subscribe international standards that define such codes (and code requesting procedures). One example is the *ISO 6523* standard developed by ISO, the International Organization for Standardization.

ISO 6523 assigns an International Code Designator (ICD) to organizations. The ICD is a four-digit code which can be used in identification schemes to uniquely identify an organization.

1.9.2. Participant Identifier Format

Another important aspect of the participant identifier is its format. For example, the *ISO6523* identifier can be formatted following the *ebCoreId* standard or a custom format, defined by the

ecosystem.

NOTE For more about *ebCoreId*, see [OASIS eBCore party identifier](#).

The *ebCoreId* is URN based and it is defined as follows:

```
urn:oasis:tc:ebcore:partyid-type:<catalog-identifier>:<scheme-in-catalog><identifier>
```

Where:

- **<catalog-identifier>** represents the catalog of the identifiers (e.g., ISO6523, DUNS, etc.)
- **<scheme-in-catalog>** stands for the scheme of the identifier in the catalog (e.g. for ISO6523 are the ICD numbers: 0088, 9925, etc.)
- **<identifier>** stands for the specific identifier assigned to the party by the authority.

An example of an *ebcoreId* identifier is:

```
urn:oasis:names:tc:ebcore:partyid-type:iso6523:0088:123456789
```

PEPPOL Identifiers Format

An example of a custom party identifier format is the PEPPOL identifier format. PEPPOL is a global network of participants for invoicing exchange and procurement, originally established in the EU.

The PEPPOL identifier format consists of a scheme identifier, such as *iso6523-actorid-upis*, followed by a double colon (:) and the actual identifier.

Unlike *ebCoreId*, the PEPPOL identifier incorporates the ICD code as part of the identifier. The format is defined as **<scheme>::<identifier>**, and an example of this identifier is:

```
iso6523-actorid-upis::0088:123456789.
```

1.9.3. Participant DNS Lookup

The main purpose of SML is to provide a DNS lookup service based on the *participant identifier*. When a participant identifier is registered in the SML, it's associated with a unique DNS record with DNS name consisting in the hash of the participant identifier and the domain name of the SML, i.e:

```
<hash(formated participant identifier)> + "." + <SML domain name>.
```

NOTE

The correct calculation of the hash value for a participant identifier depends significantly on its formatting. For example, the PEPPOL identifier format *is not compatible with* the *ebCoreId* format, resulting in different hash values. The ecosystem operator determines both the identifier's format and the method used to compute its hash value.

Some ecosystems may define different DNS name formats, as outlined below. For example, in the PEPPOL ecosystem, the DNS name is structured so that the scheme remains outside the hashing function, while only the identifier value is hashed:

`hash(<identifier-value>) + "." + <scheme-value> + "." + <SML domain name>`

But, when the ebCoreId format is used, the entire URN – including the scheme and catalog identifier – is hashed, as demonstrated in the previous example.

1.9.4. DomiSML Party Identifier Handling

The DomiSML implementation is designed to support multiple participant identifier formats, including PEPPOL and ebCoreId.

To achieve this, DomiSML utilizes a *stack of PartyIdentifier handlers*, each responsible for managing a specific format:

DomiSML PartyIdentifier Handlers

- **Template Party Identifier** – This identifier is the first in the sequence for participant identification. It is activated when the `partyIdentifier.template` property is provided and the identifier's schema matches the template configuration.
- **URN Party Identifier** – A specialized version of the template identifier, this handler is activated when the `partyIdentifier.splitPattern` property is set.
- **ebCoreId Identifier** – This handler is triggered when the identifier begins with value `urn:oasis:names:tc:ebcore:partyid-type:.`
- **OasisSMP/PEPPOL Identifier** – If none of the preceding handlers is activated, this handler is engaged when the identifier starts with the scheme name.

Template Party Identifier

The *Template Party Identifier* is the first in the sequence for participant identification handler.

The DomiSML property `partyIdentifier.template` is a type of property, where properties are semi-colon (;) separated.

This handler has the following properties:

▼ Collapse/Expand

Template Party Identifier Properties

Property	Description
<code>MATCH_SCHEME_REGEX</code>	The regular expression used to match the scheme of the identifier. If the scheme is not matched, the identifier is not processed by this handler. If value is null, the identifier is always processed by this handler. Example: <code>^(?i)(special-scheme-id)</code>
<code>ID_TMPL</code>	The identifier is formatted according to the template provided in this property when normalized. Example: <code>\${scheme}::\${identifier}</code>

Property	Description
ID_SPLIT_REGEX	The regular expression used to split the identifier into its components. The scheme and identifier are extracted from the identifier string. The values are used for formatting and calculating the hash in the DNS query. Example: <pre>^(?i)\s*(::)?(?<scheme>(special-scheme-id))::?(?<identifier>.+)?\s*\$</pre>
DNS_SUFFIX_TMPL	The DNS suffix is formatted according to the template provided in this property. The DNS suffix is used to create the DNS name for the participant identifier. Example: <pre>\${icd}.\${scheme}</pre>
DNS_SUFFIX_SPLIT_REGEX	The DNS suffix is split into its components using the regular expression provided in this property. The scheme and ICD are extracted from the DNS suffix string. The values are used for forming the DNS name suffix. Example: <pre>(?i)\s*(::)?(?<scheme>(special-scheme-id))::?(?<icd>[0-9]{4})?:?(?<identifier>.+)?\s*\$</pre>
DNS_HASH_MODE	The hash mode used to calculate the hash value for the DNS name. <i>The possible values are:</i> • ALL_IN_HASH: The complete participant identifier is used for the hashed value. • IDENTIFIER_IN_HASH: Only the participant identifier is used for the hashed value. • SCHEMA_AFTER_HASH: Only the participant identifier is used for the hashed value and the scheme is added after the hash value. Example: <pre>IDENTIFIER_IN_HASH</pre>

URN Party Identifier

The **URN Party Identifier** is a URN case format of the template identifier with the following default values:

- **MATCH_SCHEME_REGEX:** null
- **ID_TMPL:** \${scheme}:\${identifier}
- **ID_SPLIT_REGEX:** the value of `partyIdentifier.splitPattern` property.
- **DNS_SUFFIX_TMPL:** null

- `DNS_SUFFIX_SPLIT_REGEX`: `null`
- `DNS_HASH_MODE`: `ALL_IN_HASH`

EBCoreId Identifier

This handler is activated when the identifier begins with `urn:oasis:names:tc:ebcore:partyid-type:.`. It successfully processes ebCoreId identifiers formatted with a double colon (`::`), as defined in the Oasis SMP standard, as well as those formatted with a single colon (`:`) to maintain the correct URN format.

OasisSMP/PEPPOL Identifier

The Oasis SMP identifier requires the scheme name and identifier to be separated by a double colon (`::`). When generating the DNS name, the scheme is excluded from the hash value and appended after it. The scheme is optional, but if mandated, it can be configured using the `partyIdentifier.scheme.mandatory` property (default: `false`).

1.10. Validations

1.10.1. 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 1

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

```
PARTICIPANT_ID_REGEX = ^((((1234|45678|9584|9635):).*)|(\*))$
```

Example 2

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

```
PARTICIPANT_ID_REGEX = ^.*$
```

1.10.2. Logical Address validation per Domain

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

An additional property `SMP_URL_SCHEMAS` has been introduced in the table `BDMSL_SUBDOMAIN` for 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
```

1.11. Security

1.11.1. DNS

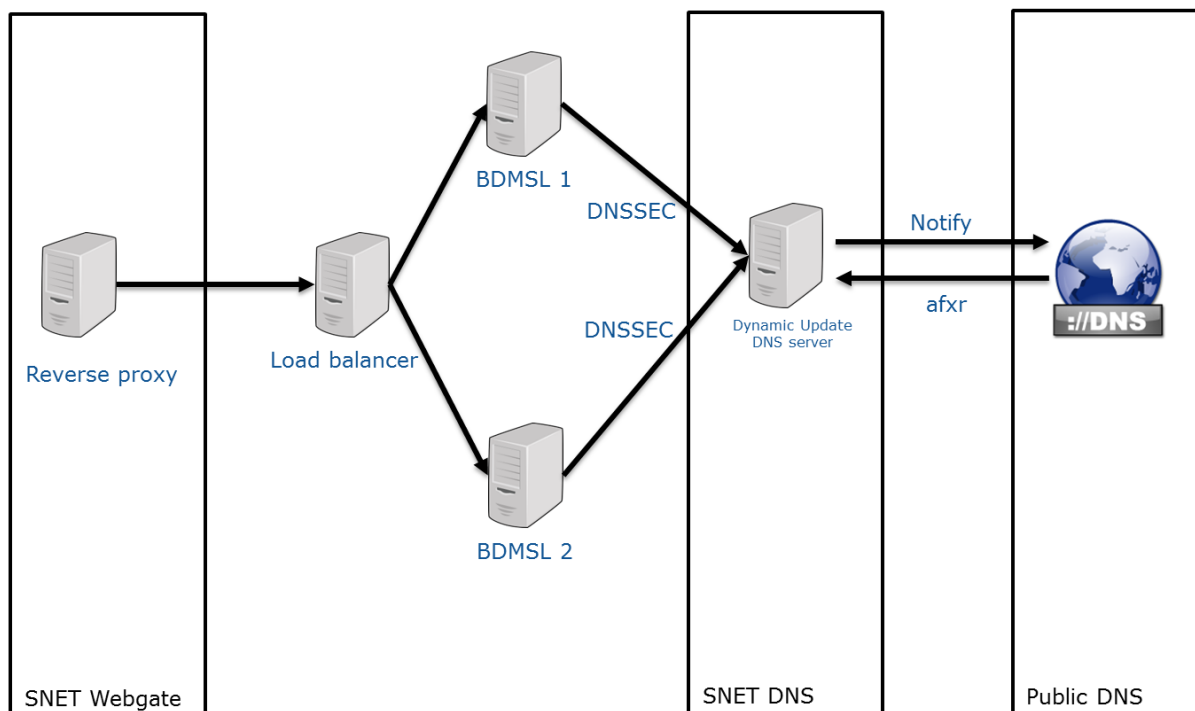
DNS specifications

The [SML Specification](#) states in its 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, to mitigate the risk of DNS spoofing, the DNSSEC can be used in the eDelivery DomiSML 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.

The Configuration of the DNSSEC is done by the DNS server and it is out of the scope of this document. Example of DNSSEC configuration for BIND9 DNS server can be found in the [BIND9 DNSSEC Guide](#).



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 two 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!" .
```

To mitigate the risk of DNS spoofing, the BDMSL can use the DNSSEC infrastructure. The deployment infrastructure is described in the [DNS](#) section.

1.11.2. 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.

How to Generate a Private Key

1. Download one of the latest BDMSL .war files from the repository on the [Digital site](#).
2. Extract the .war file using any extracting tool.
3. Run the following commands to create a private key.

```
cd bdmsl-webapp-{thisversion}
java -cp "WEB-INF/lib/*" eu.europa.ec.bdmsl.common.util.PrivateKeyGenerator c:
\temp\encryptionPrivateKey.private ①
```

Where:

- ① is a **required parameter** is to be replaced by the full directory path where the private key is to be created.

Example:

Printed result:

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

NOTE

Once the private key is generated, please copy the private key file name, for example `encryptionPrivateKey.private` to the value of the `encryptionPrivateKey` property in the `BDMSL_Configuration` table, and copy the private file to the path configured in the `configurationDir` property.

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:

1. 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 ① ②
```

Where:

- ① 1st parameter is the private key location.
- ② 2nd parameter is the password in plain text.

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

Example:

Property	Description
<code>httpProxyPassword</code>	<code>vXA7JjCy0iDQmX1UEN1Qwg==</code>

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

Example:

Property	Description
<code>keystorePassword</code>	<code>vXA7JjCy0iDQmX1UEN1Qwg==</code>

1.11.3. Vault Integration

The alternative to storing encrypted passwords in the database is to use an external vault. Vaults serve as secure storage for sensitive information, such as passwords, keys, and other secrets, in a protected component: a vault instance. DomiSML uses the factory model implemented in eDelivery's Vault API library to access various vault implementations. This library provides a unified interface for interacting with different vaults.

The actual implementation of the integration to an external vault instance is done via this library which must be deployed in the DomiSML library folder.

Vault Integration with Factory Model

plantuml::content/_common/diagrams//architecture/c4-vault-integration.puml[svg, width=904, align="center"]

To add the vault integration to the DomiSML, the following steps are required:

1. Add the vault instance implementation to the classpath (e.g. `edelivery-vault-hashicorp-2.0.jar`). The classpath is set in the application server configuration file: `sml.config.properties` with the `sml.libraries.folder` key.

```
# *****
# Additional Library folders which are loaded
# at domisml startup
# *****
sml.libraries.folder=./domisml-libs/
```

2. Configure the DomiSML using the Admin SOAP API or by setting the following configuration options in to the `BDMSL_CONFIGURATION` table:

```
# *****
```

```
# Vault configuration
# *****
vault.enabled=true
vault.implementation.classname=eu.europa.ec.edelivery.vault.hashicorp.HashicorpVault
vvault.configuration=hashicorp-vault.url:http://vault-service:8200/;hashicorp-
vault.mount.path:secret;hashicorp-vault.value.key.prefix:domisml/test-
vault.authentication.type=token
vault.authentication.value=my-root-token
```

NOTE See vault instance’s provider’s documentation for details regarding configuration.

Once the vault is enabled, DomiSML:

- stores encrypted passwords in the vault;
- retrieves passwords from the vault as needed, ensuring that sensitive information is stored securely and accessed only by authorized components.

1.11.4. 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.

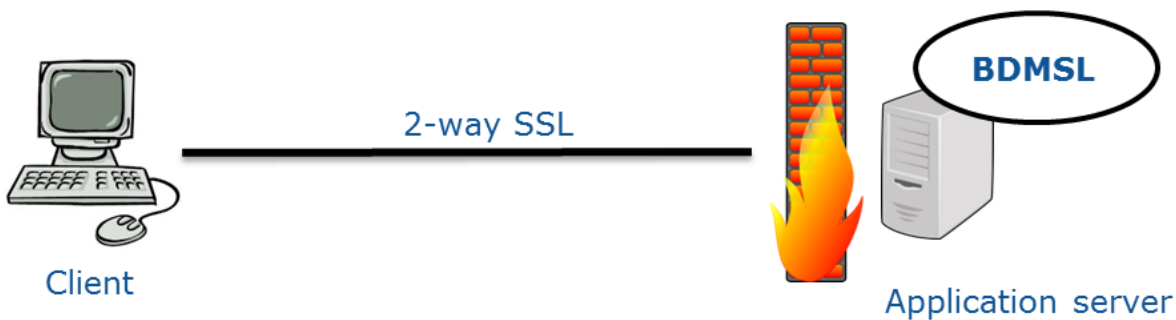
A certificate is valid if:

- 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 truststore 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.

Application Server SSL Configuration

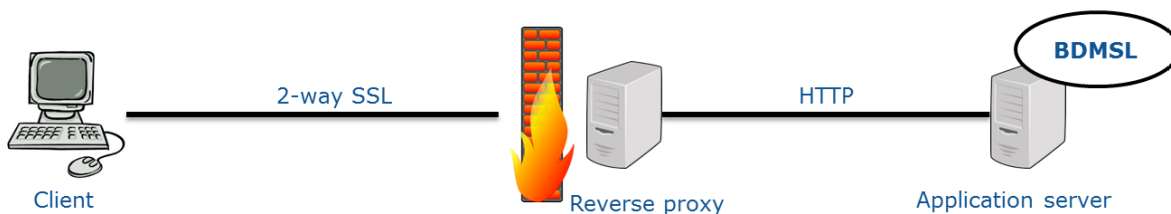
The 2-way SSL configuration can be directly set up on the application server hosting the application:



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

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:



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`.

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.

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`

Enable/disable BlueCoat Authentication Flag

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).

1.11.5. Authorizations

Roles

There are three 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: <code>authorization.smp.certSubjectRegex</code> . See Technical Requirements . For a Non Root Certificate Authority, the CN (Common Name) must contain <code>SMP</code> . See Granting the ROLE_SMP role for more information.
ROLE_MONITOR	The role only for invoking <code>isAlive</code> function	No certificate needed, the right credentials must be sent via the HTTP header attribute <code>Admin-Pwd</code> or <code>Monitor-Token</code>
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 <code>is_admin</code> set to true.

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, an `UnauthorizedFault` SOAP fault is thrown.

Granting `ROLE_SMP`

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

- One is the *Domain Certificate-issuer-based authorization* (also known as *Root Certificate Authority*) method that automatically authorizes 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.

Certificate-issuer-based Authorization

This 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_REGEX`.

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).*$`

NOTE

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.

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**.

1.11.6. 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>
```

1.12. Technical requirements

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

1.12.1. Hardware

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

1.12.2. Software

Recommended stack

- Ubuntu 24.04 LTS 64 bits
- Eclipse Temurin OpenJDK 21 TLS
- Tomcat 10.1
- Oracle Database 19c+

Operating Systems

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

Java Virtual Machines

- Eclipse Temurin OpenJDK 21 TLS

Java Application Servers

- Apache Tomcat 10.1

Databases

- Oracle Database 19c
- MySQL 8.0

Web Browsers

- Microsoft Edge
- Mozilla Firefox
- Google Chrome

1.13. Configuration

1.13.1. Application Configuration

Property	Description	Enc.
<i>Properties listed with (*) are mandatory</i>		

Property	Description	Enc.
adminPassword	BCrypt Hashed password to access admin services	FALSE
	Example: \$2a\$10\$...Bi	
(*) authentication.bluecoat.enabled	Is blue coat enabled.	FALSE
	Possible values: TRUE/FALSE. Example: FALSE	
	NOTE The property should be enabled only if is protected by the reverse proxy.	
(*) authentication.sslclientcert.enabled	Enable/Disable SSLClientCert header authentication.	FALSE
	Possible values: TRUE/FALSE. Example: FALSE	
	NOTE The property should be enabled only if is protected by the reverse proxy.	
(*) authorization.smp.certSubjectRegex	User with ROOT-CA is granted SMP_ROLE only if its certificates Subject matches configured regexp.	FALSE
	Example: ^.(CN=SMP_ OU=PEPPOL TEST SMP).\$	

Property	Description	Enc.
(*) authorization.domain.legacy.enabled	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 truststore. In case of false value Clien-Cert header cannot be used. Example: TRUE	FALSE
(*) cert.revocation.validation.graceful	In case of authorization.domain.legacy.enabled is set 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!. Example: TRUE	FALSE
(*) cert.revocation.validation.crl.protocols	In case of authorization.domain.legacy.enabled is set to false. All certificates in the truststore chain are validated and CRL URL is retrieved from the certificates directly. Comma separated list of allowed CRL and/or OCSP transport protocols for fetching the CRL list or OCSP validation. Example: http://,https://	FALSE

Property	Description	Enc.
<code>cert.revocation.validation.strategy</code>	Certificate validation strategy. Possible Values: OCSP_CRL, CRL_OCSP, OCSP_ONLY, CRL_ONLY, NO_VALIDATION. Default: OCSP_CRL (OCSP first, CRL second if OCSP fails). Example: CRL_ONLY	-
(*) <code>unsecureLoginAllowed</code>	True if the use of HTTPS is not required. If the VALUES is set to true, then the user <code>unsecure-http-client</code> is automatically created. Possible Values: TRUE/FALSE. Example: FALSE	FALSE
(*) <code>configurationDir</code>	The path to the folder containing all the configuration files (keystore and sig0 key). Example: <code>./</code>	FALSE
(*) <code>sm1.property.refresh.cronJobExpression</code>	Property refresh CRON expression (def 7 minutes to each hour)! Example: <code>0 53 */1 * * *</code>	FALSE
(*) <code>certificateChangeCronExpression</code>	CRON expression for the <code>changeCertificate</code> job. Example: <code>0 0 2 ? * *</code> .Expression which expresses the schedule: everyday at 2:00 am.	FALSE

Property	Description	Enc.
<code>smp.update.max.part.size</code>	Maximum number of participants on SMP which are automatically updated/deleted when calling services: <code>ManageServiceMetadataService/Update</code> <code>ManageServiceMetadataService/Delete</code> 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. Example: 1000	FALSE
<i>Properties listed with (*) are mandatory</i>		
(*) <code>dataInconsistencyAnalyzer.cronJobExpression</code>	CRON expression for dataInconsistencyChecker job. Example: 0 0 3 ? * *, which expresses the schedule: "every day at 3:00 am".	FALSE
(*) <code>dataInconsistencyAnalyzer.recipientEmail</code>	Email address to receive Data Inconsistency Checker results. Example: email@domain.com	FALSE
(*) <code>dataInconsistencyAnalyzer.senderEmail</code>	Sender email address for reporting Data Inconsistency Analyzer. Example: automated-notifications@some-mail.eu	FALSE
(*) <code>dataInconsistencyAnalyzer.serverInstance</code>	Server instance (hostname) to generate report. Example: localhost	FALSE
<i>Properties listed with (*) are mandatory</i>		

Property	Description	Enc.
(*) mail.smtp.host	Email server - configuration for submitting the emails. Example: mail.server.com	FALSE
(*) mail.smtp.port	Smtp mail port - configuration for submitting the emails. Example: 25	FALSE
(*)mail.smtp.protocol	smtp mail protocol- configuration for submitting the emails. Example: smtp	FALSE
mail.smtp.username	smtp mail protocol- username for submitting the emails.	FALSE
mail.smtp.password	smtp mail protocol - encrypted password for submitting the emails.	TRUE
mail.smtp.properties	smtp mail semicolon (;) separated properties. Example: mail.smtp.auth:true;mail.smtp.starttls.enable:true;mail.smtp.quitwait:'FALSE'	FALSE
<i>Properties listed with (*) are mandatory</i>		
(*) dnsClient.SIG0Enabled	true if the SIG0 signing is enabled. Required for DNSSEC. Possible values: TRUE/FALSE. Example: FALSE	FALSE
dnsClient.show.entries	If true than service ListDNS transfer and show the DNS entries (not recommended for large zones). Possible values: TRUE/FALSE. Example: TRUE	FALSE

Property	Description	Enc.
<code>dnsClient.tcp.timeout</code>	DNS TCP timeout in seconds. If the value is not given then tcp timeout is set to default value 60 seconds.	FALSE
	Example: TRUE	
(*) <code>dnsClient.SIG0KeyFileName</code>	The actual SIG0 key file. Should be just the filename if the file is in the classpath or in the configurationDir.	FALSE
	Example: <code>SIG0.private</code>	
(*) <code>dnsClient.SIG0PublicKeyName</code>	The public key name of the SIG0 key.	FALSE
	Example: <code>sig0.acc...ec.test.eu</code>	
<code>dnsClient.TSIGEnabled</code>	Property enables/disables use of TSIG signing. Values: true, false	FALSE
	Example: <code>false</code>	
<code>dnsClient.TSIGKeyName</code>	The name of the shared key e.g. acc.edelivery.tech.ec.europa.eu used for TSIG signing."	FALSE
	Example: <code>acc.edelivery.tech.ec.europa.eu</code>	
<code>dnsClient.TSIGKeyValue</code>	Base64 encoded and encrypted shared secret for TSIG signing. The value is encrypted with the same key as the keystore password.	TRUE
	Example: <code>vXA7JjCyEN1Qwg==</code>	
<code>dnsClient.TSIGAlgorithm</code>	The RFC8945 algorithm name of the shared key. Possible values: <code>hmac-md5</code> , <code>hmac-sha1</code> , <code>hmac-sha224</code> , <code>hmac-sha256</code> , <code>hmac-sha384</code> , <code>hmac-sha512</code> .	TRUE
	Example: <code>hmac-sha256</code>	

Property	Description	Enc.
(*) <code>dnsClient.enabled</code>	True if registration of DNS records is required. Must be true in production. Possible values: TRUE/FALSE. Example: FALSE	FALSE
(*) <code>dnsClient.publisherPrefix</code>	This is the prefix for the publishers (SMP). This is to be concatenated with the associated DNS domain in the table <code>bdmsl_certificate_domain</code>. Example: <code>publisher</code>	FALSE
(*) <code>dnsClient.server</code>	The DNS server. Example: <code>ddnsext.tech.ec.europa.eu</code>	FALSE
(*) <code>encryptionPrivateKey</code>	Name of the 256 bit AES secret key to encrypt or decrypt passwords. Example: <code>encryptionPrivateKey.private</code>	FALSE
<code>signResponseAlgorithm</code>	The signature algorithm to use when signing responses. Examples: http://www.w3.org/2001/04/xmlsig-more#rsa-sha256 http://www.w3.org/2021/04/xmlsig-more#eddsa-ed25519	FALSE
<code>signResponseDigestAlgorithm</code>	The signature digest algorithm to use when signing responses. Examples: http://www.w3.org/2001/04/xmlenc#sha256 http://www.w3.org/2001/04/xmlenc#sha512	FALSE
(*) <code>useProxy</code>	True if a proxy is required to connect to the internet. Possible values: TRUE/FALSE. Example: FALSE	FALSE

Property	Description	Enc.
(*) httpProxyHost	The http proxy host. Example: localhost	FALSE
(*) httpProxyPassword	Base64 encrypted password for Proxy. Example: vXA7JjCyEN1Qwg==	TRUE
(*) httpProxyPort	The http proxy port. Example: 8012	FALSE
(*) httpProxyUser	The proxy user. Example: user	FALSE
<i>Properties listed with (*) are mandatory</i>		
(*) signResponse	True if the responses must be signed. Possible values: TRUE/FALSE. Example: FALSE	FALSE
keystoreType	The keystore type. Possible values: JKS/PKCS12. Example: JKS	FALSE
(*) keystoreAlias	The alias in the keystore for signing responses. Example: senderalias	FALSE
(*) keystoreFileName	The (JKS or P12) keystore file. Should be just the filename if the file is in the classpath or in the configurationDir. Example: keystore.jks	FALSE
<i>Properties listed with (*) are mandatory</i>		
(*) keystorePassword	Base64 encrypted password for Keystore. Example: vXA7JjCy0EN1Qwg==	TRUE
(*) truststoreFileName	The truststore file (JKS or p12) should be just the filename if the file is in the classpath or in the configurationDir. Example: truststore.p12	FALSE

Property	Description	Enc.
truststoreType	The truststore type. Possible values: JKS/PKCS12. Example: PKCS12	FALSE
(*) truststorePassword	Base64 encrypted password for Truststore. Example: vXA7JjCy0EN1Qwg==	TRUE
partyIdentifier.splitPattern	Regular expression with groups <code><scheme></code> and <code><identifier></code> for splitting the URN identifiers to scheme and identifier part. Example: ^(?i)\\s*?(?<scheme>urn:ehelth:partyid-type)::?(?<identifier>.+)?\\s*\$'!	FALSE
partyIdentifier.template	Experimental template party configuration to format identifiers and DNS query. Example: MATCH_SCHEME_REGEXP=^(?i)(iso6523-actorid-upis); ID_TMPL=\${scheme}::\${identifier}; ID_SPLIT_REGEXP=^(?i)\\s*(::)?(?<scheme>(iso6523-actorid-upis))::?(?<identifier>.)?\\s*\$; DNS_SUFFIX_TMPL=\${icd}.\${scheme}; DNS_SUFFIX_SPLIT_REGEXP=^(?i)\\s*(::)?(?<scheme>(iso6523-actorid-upis))::?(?<icd>\\d{4})?:?(?<identifier>.)?\\s*\$	FALSE
partyIdentifier.scheme.mandatory	Property defines if scheme for participant identifier is mandatory. Example: FALSE	FALSE

Property	Description	Enc.
<code>partyIdentifier.scheme.caseSensitive</code>	Specifies separated scheme list of participant identifiers that must be considered case-sensitive. Example: <code>sensitive-participant-sc1 sensitive-participant-sc2</code>	FALSE
(*) <code>report.expiredSMPCertificates.cron</code>	CRON expression for triggering the report generation of expired SMP certificates job. Example: <code>0 22 6 ? * *</code> . Expression defining the schedule: every day at 3:00 am.	FALSE
<code>report.expiredSMPCertificates.recipientEmail</code>	Email address to receive expired SMP certificates report. Example: <code>email@domain.com</code>	FALSE
<code>report.expiredSMPCertificates.senderEmail</code>	Sender email address for expired SMP certificates report. Example: <code>notifications@some-mail.eu</code>	FALSE
<code>vault.enabled</code>	Property enables/disables the external vault usage. When enabled encrypted data is stored in the vault instead of the database. Example: <code>TRUE</code>	FALSE
<code>vault.implementation.classname</code>	The full class name of the vault implementation e.g. <code>eu.europa.ec.edelivery.vault.MyVault</code> or the vault implementation code/identifier if vault is registered as Java SPI (Service Provider Interface). Example: <code>eu.europa.ec.edelivery.vault.hashicorp.HashicorpVault</code> or <code>hashicorp-vault</code>	FALSE

Property	Description	Enc.
<code>vault.configuration</code>	The list of vault properties separated by <code>;</code> (semicolon). Properties are vault implementation specific.) Example: <code>hashicorp-vault.url:http://vault-service:8200/;hashicorp-vault.mount.path:secret;hashicorp-vault.value.key.prefix:domisml/test-</code>	<code>FALSE</code>
<code>vault.authentication.type</code>	The authentication type. e.g. token, username. The value depends on the vault implementation Example: <code>token</code>	<code>FALSE</code>
<code>vault.authentication.value</code>	The authentication value and format depends on the vault implementation and authentication type defined in <code>vault.authentication.type</code> . Example: <code>my-root-token</code>	<code>FALSE</code>
<code>report.expiredSMPCertificates.serverInstance</code>	If <code>sml.cluster.enabled</code> is set to true then then instance (hostname) to generate report. Example: <code>localhost</code>	<code>FALSE</code>

1.13.2. 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`

To configure a subdomain please follow the steps below:

1. Create a subdomain in the table BDMSL_Subdomain:

	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

Fig. 13. Example

NOTE

It's mandatory to define the new subdomain as `NON_ROOT_CA` or `ROOT_CA` in the column `IS_ROOT_CA`.

See [Granting the `ROLE\_SMP` role](#).

2. 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.isaibb.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

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.isaithb.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

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

PARTICIPANT_ID_REGEX = 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..

NOTE

The values of the properties aforementioned are case insensitive.

1.13.3. Application Server 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:

Error Codes

Property	Example	Description
<code>sml.hibernate.dialect</code>	<code>org.hibernate.dialect.Oracle10gDialect</code>	<i>Hibernate database dialect for accessing the database.</i>
<code>sml.datasource.jndi</code>	<code>jdbc/cipaeDeliveryDs</code>	<i>Datasource JNDI name configured on application server.</i>
<code>sml.jsp.servlet.class</code>	<code>org.apache.jasper.servlet.JspServlet</code>	<i>Application server implementation of JSP framework</i>
<code>sml.log.folder</code>	<code>./logs/</code>	<i>Logging folder.</i>
<code>sml.libraries.folder</code>	<code>./domisml-libs/</code>	<i>Path to the library folder. Absolute or relative path to the application server's work directory.</i>

Example:

Name for this Code Sample

```
# *****
# 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
# *****

# tomcat datasource JNDI example
sml.datasource.jndi=java:comp/env/jdbc/edelivery

# *****
# JSP implementation configuration
```

```
# *****
# tomcat, jboss
sml.jsp.servlet.class=org.apache.jasper.servlet.JspServlet

# *****
# Logging implementation
# *****

sml.log.folder=./logs/
```

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 two purposes:

- Defining the context root of the application;
- Linking the datasource to the globally defined JNDI datasource.

Chapter 2. Quick Start Guide

DomiSML was previously named BDMSL, which stands for Business Document Metadata Service Location.

DomiSML is the sample implementation of the SML maintained by DG DIGIT.

This guide refers to DomiSML/BDMSL 4.x versions.

Version 4 implements the [eDelivery BDXL profile](#).

2.1. Guide Overview

In this guide you can find a different steps to install the DomiSML on a:

- Tomcat 10.x server with a MySQL database;
- Tomcat 10.x server with an Oracle database.

2.1.1. Software Requirements

Install the following supported software on the target system:

Java Runtime Environment (JRE)	Database	Server
	One of the supported Database Management Systems	One of the supported Application Servers
• Java 21 Temurin OpenJDK 21	• MySQL 8.0.x ◦ tested version, future versions might also work • Oracle 19c and Oracle 21c express edition ◦ tested versions, future versions might also work	• Tomcat 10.1.x ◦ tested with Temurin OpenJDK 21

2.1.2. Binaries Repository

The eDelivery DomiSML artefacts can be downloaded from the [Digital portal](#).

2.1.3. Source Code Repository

The source code of eDelivery DomiSML is available from the following Git repository → <https://ec.europa.eu/digital-building-blocks/code/projects/EDELIVERY/repos/bdmsl/browse> .

IMPORTANT

Note as stated in the [software requirements](#), SML deployments has only been tested on Tomcat 10.1 application server.

2.2. Installation

Installation Overview

- DomiSML 4.x and 5.x does not use Liquibase as a database management tool, contrary to the previous version.
- Before installing, you need to create a database using the SQL scripts bundled in the `sml-5.x-setup.zip` archive file.
- Business properties are stored in the database table `BDMSL_CONFIGURATION`.
- Properties such as datasource JNDI, log folder, etc., are located in the `smp.config.properties` file which must be located in the server's classpath.

The deployment of the eDelivery DomiSML is made of the following mandatory steps:

- Database configuration
- Application Server preparation
- DomiSML initial configuration
- DomiSML file deployment

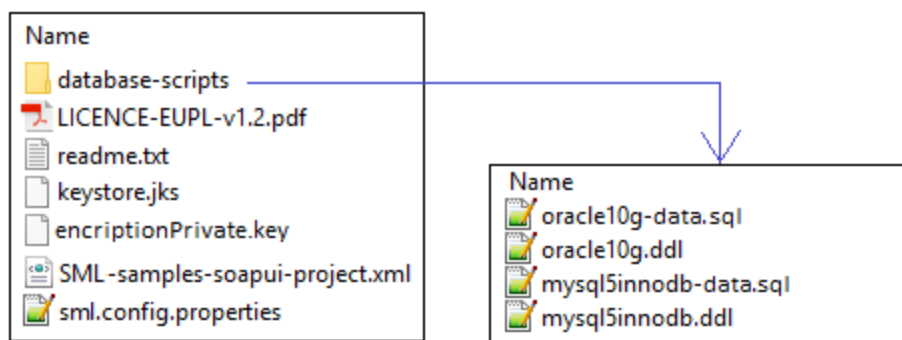
NOTE

The environment variable, `edelivery_path`, is a placeholder for your system's location, you have installed the DomiSML package, i.e. `CATALINA_HOME` for Tomcat.

2.2.1. Database

Database Scripts

The scripts to create (or migrate) the Oracle or MySQL databases are included in the `sml-5.x-setup.zip` available for download from the Digital portal.



Database Creation

Here you can find the necessary steps to create the database, the tables and the DomiSML database user (`dbuser`, used for database connection purposes).

This step is performed using the script included in the `sml-5.x-setup.zip` archive mentioned in the previous section ([Database Scripts](#)).

MySQL database

1. Download and copy the `mysql.ddl` script to `edelivery_path/database-scripts`.
2. Open a command prompt and navigate to the `edelivery_path/database-scripts` folder.
3. Execute the following MySQL commands.

WARNING This step will delete the user schema if it already exists in the database.

```
mysql -h localhost -u <root_user> -password=<root_password> -e \  
"DROP SCHEMA IF EXISTS bdmsl_schema; CREATE SCHEMA  
bdmsl_schema; ALTER DATABASE bdmsl_schema charset=utf8; CREATE  
USER sml_dbuser IDENTIFIED BY 'sml_password'; GRANT ALL ON  
bdmsl_schema.* TO sml_dbuser;"
```

This creates the `bdmsl_schema` and a `bdmsl` database user `sml_dbuser` with all privileges on this schema.

4. Create the required objects (tables, etc.) in the database, by executing:

```
mysql -h localhost -u <root_user> -p<root_password> bdmsl_schema <  
https://ec.europa.eu/digital-building-  
blocks/code/projects/EDELIVERY/repos/smp/browse/smp-webapp/src/main/smp-  
setup/database-scripts/mysql-5.0.ddl?at=refs%2Fheads%2Fdevelopment[mysql.ddl]
```

5. Set up the initial data, by executing:

```
mysql -h localhost -u root_user -p<root_password> bdmsl_schema <  
mysql-data.sql
```

Oracle database

1. Download and copy the `oracle.ddl` script to `edelivery_path/sql-scripts`.
2. Navigate to `edelivery_path/sql-scripts` directory and execute:

```
sqlplus sys as sysdba
```

Where:

- The expected password is the one assigned during Oracle's installation.

3. Once logged in Oracle:


```
CREATE USER sml_dbuser IDENTIFIED BY sml_dbpassword;  
GRANT ALL PRIVILEGES TO sml_dbuser;  
CONNECT sml_dbuser;  
SHOW USER
```

4. Run the scripts with the @ sign from their location:

```
@oracle.ddl ①  
@oracle-data.sql ②  
  
exit
```

① Script for Oracle's database creation.

② Script for Oracle's database initialization.

2.3. Configuration

2.3.1. Tomcat Configuration

To deploy the DomiSML on Tomcat, complete the steps in the next sections.

Configuring the Extra CLASSPATH for Tomcat

In the root path of the Tomcat's installation (**CATALINA_HOME**), the following directories:

- **keystores** - all security artifacts such as Keystore, Truststore, and encryption key.
- **logs**
- **classes** - the DomiSML configuration file **sml.config.properties**.

are created in this Tomcat deployment example.

NOTE

The **classes** folder must be added to a **CLASSPATH** variable in the Tomcat batch file, **CATALINA_HOME/bin/setenv.sh** (or **CATALINA_HOME/bin/setenv.bat**).

▼ For Linux

- Open (or create) and edit the **\$CATALINA_HOME/bin/setenv.sh** file as in the example below:

```
#!/bin/sh  
  
# Set CLASSPATH to include sml environment property file:  
# sml.config.properties  
  
export CLASSPATH=$CATALINA_HOME/classes
```

▼ For Windows

1. Open (or create) and edit the `%CATALINA_HOME%/bin/setenv.bat` file.

REM Set CLASSPATH to include sml environment property file:

REM sml.config.properties

set classpath=%classpath%;%catalina_home%\classes

2. Place the `sml.config.properties` (DomiSML's environment property file) in the `classes` folder.
3. Download an example, `sml-5.x-setup.zip`, from the [Digital portal](#).
For a description of environment properties see [Environment Parameters](#).

For tomcat/mysql configuration the file must have following properties and values:

```
###
# *****
# 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
# *****
# tomcat datasource JNDI example
sml.datasource.jndi=java:comp/env/jdbc/edelivery

# *****
# JSP implementation configuration
# *****
# tomcat
sml.jsp.servlet.class=org.apache.jasper.servlet.JspServlet

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

# *****
# Additional Library folders which are loaded
# at domisml startup
# *****
sml.libraries.folder=./domisml-libs/
# *****
# wsdl services schema validation
# *****
sml.ws.servicemetadataservices.schema-validation-enabled=true
sml.ws.participantservices.schema-validation-enabled=true
```

```
sml.ws.bdmslservices.schema-validation-enabled=true
sml.ws.bdmsladminservices.schema-validation-enabled=true
```

Configuring the Datasource for Tomcat

1. Create a [new data source in Tomcat](#) named: `java:comp/env/jdbc/edelivery`.
2. Go to `TOMCAT_HOME/conf/context.xml` and add the block:

```
<Resource name="jdbc/edelivery" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000"
    username="root" password="root" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/bdmsl"/>
```

JDBC Driver

The JDBC driver needs to be downloaded from the manufacturer website:

- For Mysql: <https://www.mysql.com/products/connector/>

The JDBC driver (.jar file) must be copied to the following directory: `edelivery_path/lib`.

2.4. Deployment

Copy the `bdmsl-webapp-5.X.war` file to the Tomcat `edelivery/webapps`.

2.4.1. Verification of the Installation

1. In a browser, to go to the following address:
`http://<hostname>:<port>/bdmsl-webapp-5.x/`.

NOTE

URL context path must match the `war` file's name. If the deployment filename is `edelivery-sml.war`, then the DomisSML URL is:

`http://<hostname>:<port>/edelivery-sml/`

If the deployment is successful, the following page is displayed:



eDelivery BDMSL is waiting for you

- Version: 4.3
- [List DNS](#)
- [Search participant/domain](#)
- [Services](#)

IMPORTANT

The context path (see example above: `/edelivery-sml/`) needs to be the same as is deployment `war` file. If the `war` file's name is `sml.war`, then the

application's URL will be `http:// <hostname>: <port>/sml/`.

2.4.2. General Configuration

Environment Parameters

The DomiSML application's environment parameters are stored in the properties file `sml.config.properties`. This configuration is in a properties file because these parameters are required before database connection.

You can find a configuration preset for the Tomcat/MySql installation scenario in the setup bundle, `sml-4.x-setup.zip` (see preset for the Tomcat/MySql installation scenario [Database Scripts](#)).

```
###
# *****
# 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
# *****
# tomcat datasource JNDI example
sml.datasource.jndi=java:comp/env/jdbc/edelivery

# *****
# JSP implementation configuration
# *****
# tomcat
sml.jsp.servlet.class=org.apache.jasper.servlet.JspServlet

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

# *****
# Additional Library folders which are loaded
# at domisml startup
# *****
sml.libraries.folder=./domisml-libs/
# *****
# wsdl services schema validation
# *****
sml.ws.servicemetadataseservices.schema-validation-enabled=true
sml.ws.participantservices.schema-validation-enabled=true
sml.ws.bdmslservices.schema-validation-enabled=true
```

```
sml.ws.bdmsladminservices.schema-validation-enabled=true
```

The configuration file has the following parameters:

- `sml.hibernate.dialect` - Hibernate dialect used for accessing the database.
- `sml.datasource.jndi` - JNDI Datasource's name configured in [Configuring the Datasource for Tomcat](#).
- `sml.jsp.servlet.class` - Application server implementation of the JSP framework.
- `sml.log.folder` - Logs folder.
- `sml.libraries.folder` - Additional libraries folder e.g. vault implementation library.

DomiSML Parameters

The DomiSML application contains its parameters in the database table `BDMSL_CONFIGURATION`.

Parameters can be updated:

- using the sql script
- by calling a webservice operation

Updating parameters using a sql script:

```
mysql -h localhost -u <root_user> -p<root_password> bdmsl_schema -e \  
"UPDATE bdmsl_configuration SET value='true', last_updated_on=NOW()  
WHERE property='unsecureLoginAllowed';"
```

Updating parameters by calling a webservice operation

Call `BDMSLAdminServices/SetProperty()`.

For more details, see the [Interface Description](#).

Property refresh

All properties are refreshed without server restart, except the CRON schedule definitions:

- `sml.property.refresh.cronJobExpression`
- `certificateChangeCronExpression`
- `dataInconsistencyAnalyzer.cronJobExpression`

The properties mentioned above are refreshed as defined in the cron property `sml.property.refresh.cronJobExpression`.

By default, properties are refreshed (if updated) every hour.

IMPORTANT

If a property is changed using the SQL script, make sure that the value `last_updated` is also changed, otherwise the properties are not updated.

For the list of properties and their description, see the [DomiSML's Architecture Overview](#).

Generating a Private Key File

DomiSML uses a private key for encrypting/decrypting passwords.

If the key is not present in the folder defined in property `configurationDir` at startup, it will automatically create and store it.

When deploying DomiSML (especially in Production), make sure its unique encryption key is generated for the deployment.

Below is an example of how to manually create the key.

Creating an encryption key

1. To create a private key, please follow the steps below:
2. Download one of the latest DomiSML war files (eg: `bdmsl-webapp-5.x.war`) from the repository <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/SML>
3. Extract the war file using any extracting tool
4. Run the following commands to create a private key:

Example - Generating a private key

```
cd bdmsl-webapp-5.x
java -cp "WEB-INF/lib/*" eu.europa.ec.bdmsl.common.util.PrivateKeyGenerator c:\temp\encryptionPrivateKey.private
```

The `<key_path>` is a required parameter and stands for the full directory path where the generated private key is to be stored.

In the example above `<key_path> = c:\temp\encryptionPrivateKey.private`.

5. Once the private key is generated,
 - Set the `encryptionPrivateKey` property's value (located in the `BDMSL_Configuration` table) as the name of the private key file you have just generated.
 - Copy the private file into the location configured in the property `configurationDir`.

Encrypting a password

DomiSML encrypts passwords automatically when setting the password property using the WebService `SetProperty`.

Setting a password

After [generating a private key at item](#), if needed, configure the proxy or keystore (used to sign responses) password as follows:

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

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

```
\privateKey.private Password123
```

Where two parameters are expected:

- 1) private key location, in this example: `c:\temp\privateKey.private`
- 2) plain text password, in this example: `Password123`.

Proxy password

To configure the proxy password, set the printed encrypted and base64 encoded password as the value of the `httpProxyPassword` property (found in the `BDMSL_CONFIGURATION` table).

```
httpProxyPassword= vXA7JjCy0iDQmX1UEN1Qwg==
```

Keystore password

To configure the keystore password, copy the printed encrypted and base64 encoded password as the value of the `keystorePassword` property (found in the `BDMSL_CONFIGURATION` table).

```
keystorePassword= vXA7JjCy0iDQmX1UEN1Qwg==
```

Certificate revocation list (CRL) Truststore

Truststore for the certificate revocation list (CRL) download over the HTTPS

The DomiSML establishes the HTTPS trust to the server hosting the CLR list by using the application server system truststore, which is defined by system variables, such as: `javax.net.ssl.trustStore`, `javax.net.ssl.trustStoreType`, etc.

If the truststore is not set, the default java truststore is used in the following location: `${JAVA_HOME}/jre/lib/security/cacerts`

To enable the download of the CRL files over HTTPS, ensure the appropriate certificates are registered in the application server truststore.

Example of how to add/register the `crl-server` certificate to default java cacerts truststore:

```
$JAVA_HOME/bin/keytool -importcert -alias crl-server -keystore  
$JAVA_HOME/jre/lib/security/cacerts -storepass changeit -file  
/opt/smlconf/init-configuration/sml_crl_crl-server.cer -noprompt
```

Sign Responses Certificate

If the flag `signResponse` is TRUE in the `BDMSL_CONFIGURATION` table, a keystore file name, its alias and password must be provided in the same table.

For testing purposes only, the provided `keystore.p12` (with password: test123) can be used. The keystore contains RSA, EC, ED25519 and ED448 key examples.

The keystore is located in the configuration bundle `sml-setup-${VERSION}.zip`.

Add files to Application Server

In the configuration directory that you specified in the `configurationDir` property, you need to add the following files:

- `keystore.p12` - this keystore must contain your private key with the alias and password defined in the `keystoreAlias` and `keystorePassword` properties.

NOTE | This name can be customized in the `keystoreFileName` property.

- `sig0.private` - this file is only required if you use DNSSEC (i.e. property `dnsClient.SIG0Enabled` set to true).

NOTE | This name can be changed in the `dnsClient.SIG0KeyFileName` property.

- `encryptionPrivateKey.private` - This private key file is only required if you use Proxy or Sign Response.

NOTE | This name can be changed in the `encryptionPrivateKey` property.

Once the needed files have been copied, restart the server(s).

2.5. DNS integration

DomiSML was developed and tested with using a BIND9 DNS server. The DNS integration can be switched on/off by setting attribute `dnsClient.enabled` to TRUE or FALSE.

If the property is set to TRUE, the parameter `dnsClient.server` must contain the hostname/IP address of the DNS server.

To secure the DNS integration, DomiSML has implemented SIG(0). This option can be enabled/disabled by the following parameter:

`dnsClient.SIG0Enabled`, with values TRUE or FALSE.

If the option is set to FALSE, the DNS should allow updates to **any** IP address

IMPORTANT | This is NOT advised in production environments.

or restrict the update permission to the requester **IP address**.

Below is example of configuration for BIND9 zone `example.edelivery.eu.local` without the use of SIG(0) (in this case the DomiSML should have `dnsClient.SIG0Enabled= false`):

```
zone "example.edelivery.eu.local" \{
  type master;
  file "/var/lib/bind/db.example.edelivery.eu.local ";
  allow-update \{ 10.22.1.3;}
  allow-transfer \{ 10.22.0.0/16; };
```



```
};
```

2.5.1. Securing DNS integration with SIG(0)

SIG0 are asymmetric key-pairs, usually with a filename ending with **.key** for a public key, and a filename ending with **.private** for a private key.

SIG(0) key pair can be created with the `dnssec-keygen` utility (which is provided as part of a BIND9 DNS server).

Examples of commands for generating keys:

DSA key

```
dnssec-keygen -a DSA -b 1024 -n HOST -T KEY sig0.example.edelivery.eu.local
```

RSA key - Example 1

```
dnssec-keygen -a RSASHA256 -b 4096 -T KEY -n HOST domism-l-  
rsasha256.test.edelivery.local
```

RSA key - Example 2

```
dnssec-keygen -a RSASHA512 -b 4096 -T KEY -n HOST domism-l-  
rsasha512.test.edelivery.local
```

EC key - Example 1

```
dnssec-keygen -a ECDSAP256SHA256 -T KEY -n HOST domism-l-  
ecdsap256sha256.test.edelivery.local
```

EC key - Example 2

```
dnssec-keygen -a ECDSAP384SHA384 -T KEY -n HOST domism-l-  
ecdsap384sha384.test.edelivery.local
```

Edward curve key - Example 1

```
dnssec-keygen -a ED25519 -T KEY -n HOST domism-l-ed25519.test.edelivery.local
```

Edward curve key - Example 1

```
dnssec-keygen -a ED448 -T KEY -n HOST domism-l-ed448.test.edelivery.local
```

NOTE

DSA algorithm is obsolete on the newer version of the Bind9 server. Older versions

of Bind9 do not support ed25519 and ed448 keys.

The command produces the following files:

- `Ksig0.example.edelivery.eu.local.+003+03054.key`
- `Ksig0.example.edelivery.eu.local.+003+03054.private`

The content of the file is as follows:

`Ksig0.example.edelivery.eu.+003+03054.key`

It is the DNS Key entry, that should be put in the DNS zone as in the example below:

```
sig0.example.edelivery.eu.local.      604800      IN      KEY      512      3      3
CLC4l6DtbtzWAIJIMkYrv4MClWvj2BUclxqCd86vzX/f0ka+oS73dFCp
tb9Yv9oYjGmG1JLnv4EKuPiGPa80/CQWrbJ5I7Yts3GDMgZNRswxMije
H6OoYkZ6ywRpjv8nommw6JMzDaDhcU5/tLQXhvx3U/c7W5QepAXfHb6Z
gGwL4TkqR/RGp5xcxayID4b/+DJvqi04BjN09WR3XGRHWZ5a00pRcRjx
imDtlnIjpsykE59o03UyQ+YT1CYNPjNlmOoT1JVgBEFGgouAm7yEZq3A
HwsqZEHCeucvQKBADmIk5rHwfZJwv7dzXrZR2U5AqE/AxqhrWyTpItRg
oGEkc+piGciuPRtwRZPkD6+GcFn/2knJ3YuRBOiog0+5mtbqaIP0ew+B
+BtQk6X5E5tNnEuQJeRjjxzN6YdzN7hTDFPvtwGEQvDUoU4SP/6YHoAd
AaH5Vs+YTRHjdISvnJIV6VRxIbQFJWaf3Z+UT4ns0+4pIGXm7C0ADA2a
1wGpj4QF8A37VAofcFWLUertNv9YmVHQC2l
```

When the public key is correctly registered on the DNS server, it can be tested with the dig util as in the example below:

```
$dig sig0.example.edelivery.eu.local @localhost KEY
```

```
ANSWER
; <<>> DiG 9.10.3-P4-Ubuntu <<>> sig0.example.edelivery.eu.local
@localhost KEY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36443
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL:
2
+
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
; sig0.example.edelivery.eu.local. IN KEY
+
;; ANSWER SECTION:
sig0.example.edelivery.eu.local. 604800 IN KEY 512 3 3
CLC4l6DtbtzWAIJIMkYrv4MClWvj2BUclxqCd86vzX/f0ka+oS73dFCp
tb9Yv9oYjGmG1JLnv4EKuPiGPa80/CQWrbJ5I7Yts3GDMgZNRswxMije
H6OoYkZ6ywRpjv8nommw6JMzDaDhcU5/tLQXhvx3U/c7W5QepAXfHb6Z
gGwL4TkqR/RGp5xcxayID4b/+DJvqi04BjN09WR3XGRHWZ5a00pRcRjx
imDtlnIjpsykE59o03UyQ+YT1CYNPjNlmOoT1JVgBEFGgouAm7yEZq3A
HwsqZEHCeucvQKBADmIk5rHwfZJwv7dzXrZR2U5AqE/AxqhrWyTpItRg
```

```
oGEkc+piGciuPRtwRZPKD6+GcFn/2knJ3YuRB0iog0+5mtbqaIP0ew+B
+BtQk6X5E5tNnEuQJeRjxxznGYdzN7hTDFPvtwGEQvDUoU4SP/6YHoAd
AaH5Vs+YTRHjdISvnJIV6VRxIbQFJWaf3Z+UT4ns0+4pIGXm7C0ADA2a
1wGpj4QF8A37VAofcFWLUERtNv9YmVHQcA2l
```

```
+
;; AUTHORITY SECTION:
example.edelivery.eu.local. 604800 IN NS ns.
example.edelivery.eu.local.
+
;; ADDITIONAL SECTION:
ns.example.com.local. 604800 IN A 192.168.56.3
```

If we want to allow DNS updates on the zone `example.edelivery.eu.local` only by requests signed by private key of the `sig0.example.edelivery.eu.local`, we have to update the DNS zone configuration as below:

```
zone "example.edelivery.eu.local" \{
    type master;
    file "/var/lib/bind/db.example.edelivery.eu.local ";
    allow-update \{ key "sig0.example.edelivery.eu.local."; \}
    allow-transfer \{ 10.22.0.0/16; \};
};
```

2.5.2. Configuration of the SIG(0) in DomiSML

To configure DomiSML to use SIG(0), the following parameters must be set:

- **dnsClient.SIG0PublicKeyName**: must be DNS name of the DNS KEY entry. In the example above this value is:
`dnsClient.SIG0PublicKeyName= sig0.example.edelivery.eu.local`
- **dnsClient.SIG0KeyFileName**: the private key must be put into to the BDMS configuration folder and Value of the parameter **dnsClient.SIG0KeyFileName** must be the name of the SIG(0) private key filename.
As example: `dnsClient.SIG0KeyFileName= Ksig0.example.edelivery.eu.local.+003+03054.private`
- **dnsClient.SIG0Enabled**: to enable SIG(0) the configuration parameter must be set to true:
`dnsClient.SIG0Enabled= true`.

NOTE

DomiSML does not use the SIG(0) to transfer the DNS records. DNS records are retrieved from DBS server when generating inconsistency reports and when calling the resource web `/listDNS`.

Above is an example of how to secure a transfer to a network: `10.22.0.0/16`.

2.5.3. Securing DNS integration with TSIG

TSIG (Transaction Signatures), is an alternative to SIG(0), a security mechanism used in BIND9 to authenticate DNS transactions between servers. TSIG ensures that DNS messages, such as dynamic updates from DomiSML, are securely exchanged using shared secret keys and cryptographic hashing.

How TSIG Works

A shared secret key is generated and configured on both the DomiSML and slave DNS server. Each DNS message includes a TSIG record that signs the message using a cryptographic hash.

The receiving DNS server verifies the signature to ensure the message's authenticity and integrity. If the signature is invalid or missing, the transaction is rejected.

Configuring TSIG in BIND9

The TSIG key is a shared secret key used to authenticate DNS messages. It is typically generated using the `tsig-keygen` command as in the example below.

```
tsig-keygen -a hmac-shaxxx key-tsig-hmac-shaxxx
```

▼ Examples for the key algorithms:

sha256 key

```
tsig-keygen -a hmac-sha256 key-tsig-hmac-sha256
```

sha384 key

```
tsig-keygen -a hmac-sha384 key-tsig-hmac-sha384
```

sha512 key

```
tsig-keygen -a hmac-sha512 key-tsig-hmac-sha512
```

▼ To configure a bind9 DNS server:

1. Add the Key to `named.conf` as in the example below:

```
key key-tsig-hmac-sha256 {  
    algorithm hmac-sha256;  
    secret "dMXmygRu7aCTK00cjP3dSk9KY90p2LKX12VmC78Z3Xc=";  
};
```

2. Add the key to the configuration zone:

```
zone "test.edelivery.local" {  
    type master;  
    file "/etc/bind/db.test.edelivery.local";  
    allow-query { any; };  
    allow-transfer { any; };  
    allow-update {  
        key key-tsig-hmac-sha256;  
    } ;
```

```
};
```

Configuring TSIG in DomiSML

To configure DomiSML to use TSIG, the following parameters must be set:

- `dnsClient.TSIGKeyName` – The name of the shared key for TSIG signing (`dnsClient.SIG0PublicKeyName= key-tsig-hmac-sha256`).
- `dnsClient.TSIGKeyValue` – The encrypted shared secret.
 - Example: `dMXmygRu7aCTK00cjP3dSk9KY90p2LKXl2VmC78Z3Xc=`
- `dnsClient.TSIGAlgorithm` – The RFC8945 algorithm name of the shared key.
 - Possible values: `hmac-md5`, `hmac-sha1`, `hmac-sha224`, `hmac-sha256`, `hmac-sha384`, `hmac-sha512`.
 - Example: `hmac-sha256`.
- `dnsClient.TSIGEnabled` – To enable TSIG the configuration parameter must be set to true (`dnsClient.TSIGEnabled= true`).

NOTE

The DomiSML does not use TSIG to transfer the DNS records. DNS records are retrieved from DBS server when generating inconsistency reports and when calling the resource web `/listDNS`.

Above is an example of how to secure a transfer to a network: `10.22.0.0/16`. If TSIG is used, disable the SIG(0): `dnsClient.SIG0Enabled=false`.

Chapter 3. Interface Description

BDMSL stands for Business Document Metadata Service Location. DomiSML is the sample implementation of the SML maintained by DG DIGIT. The DomiSML version described in this document implements the eDelivery BDXL profile.

▼ About this Guide

This guide defines the:

- participant's interface to the BDMSL.
- WSDL and the observable behaviour of the interface(s).

In this guide we describe the following interfaces:

Interface	Description	V.	Use Cases Link
ManageServiceMetadataService-1.0.wsdl	Definition of the service to Manage Service Metadata. See also, → this page and; → this page .	1.0	Use Cases
ManageBusinessIdentifierService-1.0.wsdl	Definition of the service to Manage Participant Identifier's. → See also this page .	1.0	Use Cases
BDMSLService-1.0.wsdl	Definition of the services to Manage Service Metadata and to monitor SML specific to this implementation. → See also this page .	1.0	Use Cases
BDMSLAdminService-1.0.wsdl	Definition of the services for administration of the SML application. Services are specific to this implementation. → See also this page .	1.0	Use Cases

▼ Scope of the document

This document covers the service interface of the BDMSL. It includes information regarding the description of the services available, the list of use cases, the information model and the sequence of message exchanges for the services provided. This specification is limited to the service interface of the BDMSL. All other aspects of its implementation are not covered by this document. The ICD specification provides both the provider (i.e. the implementer) of the services and their consumers with a complete specification of the following aspects:

- *Interface Functional Specification*, this specifies the set of services and the operations provided by each service and this is represented by the flows explained in the use cases.
- *Interface Behavioural Specification*, this specifies the expected sequence of steps to be

respected by the participants in the implementation when calling a service or a set of services and this is represented by the sequence diagrams presented in the use cases.

- *Interface Message standards*, this specifies the syntax and semantics of the data.

▼ *Target Audience*

This document is intended for the Directorate Generals and Services of the European Commission, Member States (MS) and also companies of the private sector wanting access DomiSML services.

In particular:

- **Architects** - useful for determining how to best exploit BDMSL services to create a fully-fledged solution integrating other components like the SMP, the DNS and the administration application;
- **Analysts** - useful to understand the communication between the BDMSL and its major peer component the SMP which will enable them to have an holistic and detailed view of the operations and data involved in the use cases;
- **Developers** - essential as a basis of their development concerning the interaction mainly between the DOMISML and the SMP, and also with the DNS and the administration application;
- **Testers** - useful as a basis to test the interface by following the use cases described; in particular the communications of the BDMSL with the SMP.

▼ *Relevant Resources*

Here are some relevant sources for further reading:

- [Business Document Metadata Service Location Version 1.0](#)

This specification defines service discovery methods. A method is first specified to query and retrieve an 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 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.

- [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.

- [PEPPOL Transport Infrastructure](#)

- [Service Metadata Locator \(SML\)](#)

This document defines the profiles for the discovery and management interfaces for the Business Document Exchange Network (BUSDOX) Service Metadata Locator service.

The Service Metadata Locator service exposes three interfaces: Service Metadata discovery, Manage participant identifiers and Manage service metadata interfaces.

- [Service Metadata Publishing \(SMP\) Version 1.0](#)

This document describes a protocol for publishing service metadata within a 4-corner

network. In a 4-corner network, entities are exchanging business documents through intermediary gateway services (sometimes called Access Points). To successfully send a business document in a 4-corner network, an entity must be able to discover critical metadata about the recipient (endpoint) of the business document, such as types of documents the endpoint is capable of receiving and methods of transport supported. The recipient makes this metadata available to other entities in the network through a Service Metadata Publisher service. This specification describes the request/response exchanges between a Service Metadata Publisher and a client wishing to discover endpoint information. A client can either be an end-user business application or a gateway/access point in the 4-corner network. It also defines the request processing that must happen at the client side.

- [eDelivery BDXL profile](#)
Specifications of eDelivery BDXL profile.
- [Policy for use of Identifiers \(PEPPOL Transport Infrastructure\)](#)
This document describes a PEPPOL policy and guidelines for use of identifiers within the PEPPOL network.

3.1. Functional Specification

3.1.1. Purpose of the DomiSML Component

The DomiSML component enables Access Points to dynamically discover the IP address of the destination Access Point. Instead of looking at a static list of IP addresses, the Access Point consults a Service Metadata Publisher (SMP) where information about every participant in the document/data exchange network is kept up to date, including the IP addresses of their Access Point. As at any point in time there can be one or several active SMPs in a same network.

For dynamic discovery to work, every participant must be given a unique ID in the form of a website's URL which must be known on the internet's Domain Name System (DNS) thanks to the Service Metadata Locator (SML).

By knowing this URL, the Access Point is able to dynamically locate the right SMP and therefore the right Access Point.

▼ More...

By combining the SMP services with a Service Location solution building block like the [DomiSML component](#), the participants of a document/data exchange network can benefit from **dynamic discovery**. In such a configuration, a participant about to send a document or data will first use the service location service (e.g. the DomiSML) to retrieve the endpoint address of the SMP. As a second step, he will query the SMP to obtain "**Metadata**" of the receiving participant, i.e. its capabilities and settings, which includes the endpoint address of the receiver's access point. The sender has then enough information and can send the message to the receiver using the adequate transport protocol, security settings, etc.

The eDelivery Service Metadata Locator (SML) enables Access Points to dynamically discover the IP address of the destination Access Point. Instead of looking at a static list of IP addresses, the Access Point consults a Service Metadata Publisher (SMP) where information about every participant in the document/ data exchange network is kept up to date, including the IP

addresses of their Access Point. As at any point in time there can be one or several active SMPs in a same network. For dynamic discovery to work, every participant must be given a unique ID in the form of a website's URL which must be known on the internet's Domain Name System (DNS) thanks to the Service Metadata Locator (SML). By knowing this URL, the Access Point is able to dynamically locate the right SMP and therefore the right Access Point.

The current SML software component maintained by the European Commission implements the PEPPOL Transport Infrastructure SML specifications.

3.1.2. Use Cases Overview

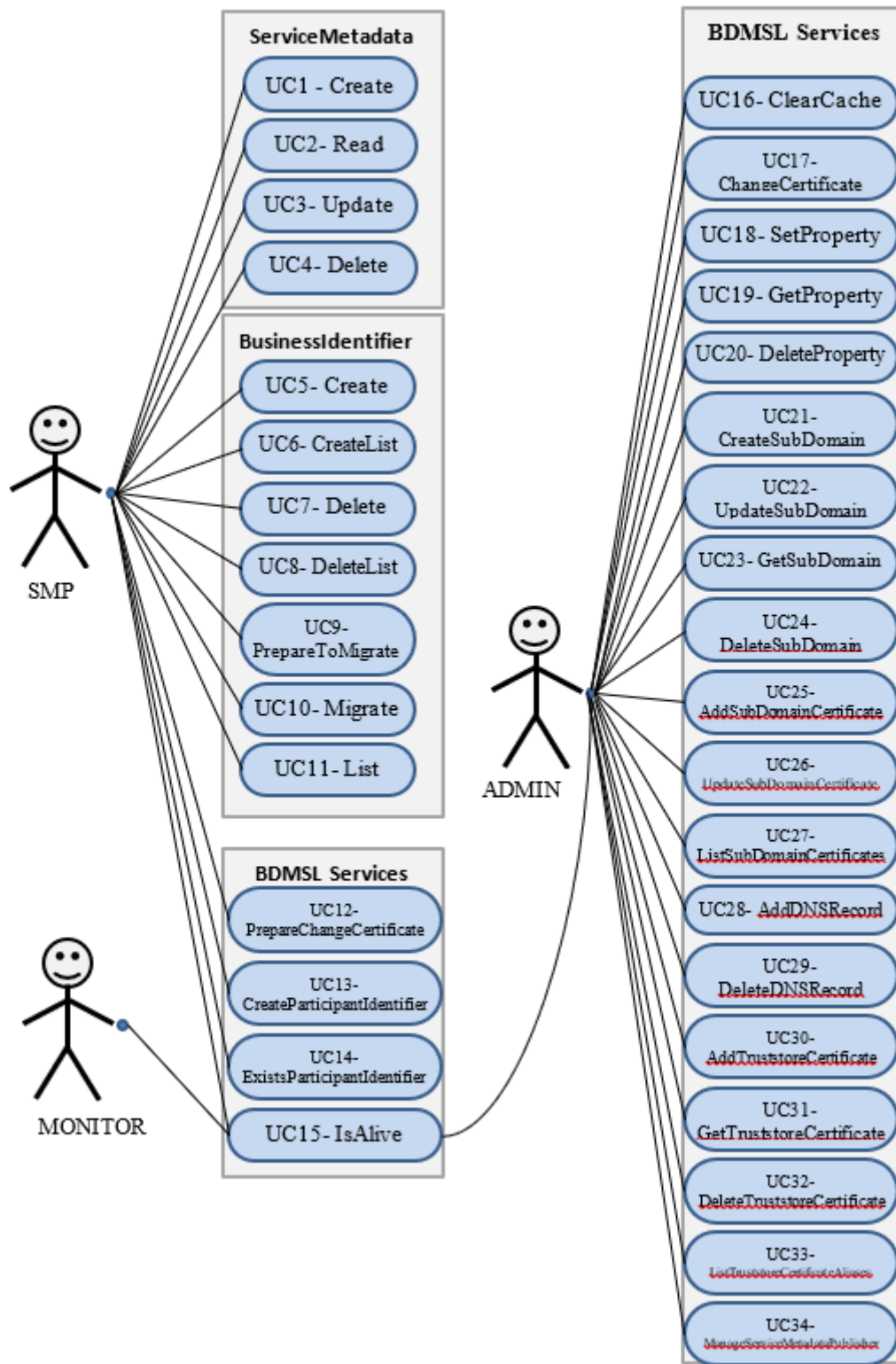


Figure. Use Cases Diagram

Actors

Actor	Definition
SMP	Holds the service metadata information about participants in the network.
SML	Provides controlled access to the creation and updating of entries in the DNS.
ADMIN	Application user with administrative privileges.

Actor	Definition
Monitor user	The Monitor user who is allowed to call the isAlive service for monitoring health of SML application.

List of Use Cases per Interface

▼ ManageServiceMetadataService-1.0.wsdl

ManageServiceMetadataService-1.0.wsdl use cases

Use Case	Actors	Description
UC1 Create	SMP	Establishes a Service Metadata Publisher metadata record, containing the metadata about the Service Metadata Publisher-information as outlined in the ServiceMetadataPublisherService data type.
UC2 Read	SMP	Retrieves the Service Metadata Publisher record for the service metadata publisher.
UC3 Update	SMP	Updates the Service Metadata Publisher record for the service metadata publisher.
UC4 Delete	SMP	Deletes the Service Metadata Publisher record for the service metadata publisher.

▼ ManageBusinessIdentifierService-1.0.wsdl

ManageBusinessIdentifierService-1.0.wsdl use cases

Use Case	Actors	Description
UC5 Create	SMP	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.
UC6 CreateList	SMP	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.
UC7 Delete	SMP	Deletes the information that the Service Metadata Locator service holds for a specific Participant Identifier.

Use Case	Actors	Description
UC8 DeleteList	SMP	Deletes the information that the Service Metadata Locator service holds for a list of Participant Identifiers.
UC9 PrepareToMigrate	SMP	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.
UC10 Migrate	SMP	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 replaced Service Metadata Publisher. The PrepareToMigrate operation MUST have been previously invoked for the supplied Participant Identifier, using the same <i>MigrationCode</i>, 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.

Use Case	Actors	Description
UC11 List	SMP	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.

▼ **BDMSLService-1.0.wsdl**

BDMSLService-1.0.wsdl use cases

Use Case	Actors	Description
UC12 PrepareChangeCertificate	SMP	Allows a SMP to prepare a change of its certificate when the current one is about to expire, and the future one is available.
UC13 CreateParticipantIdentifier	SMP	This service has the same behaviour as the Create() operation in the ManageParticipantIdentifier (UC5) interface but it has one additional and optional() Input: the serviceName element.
UC14 ExistsParticipantIdentifier	SMP	The method enables an SMP to verify if the SMP was already registered the participant into the DomiSML.
UC15 IsAlive	SMP, ADMIN, Monitor user	Confirms that the application is up and running (monitoring purposes)

▼ **BDMSLAdminService-1.0.wsdl**

BDMSLAdminService-1.0.wsdl use cases

Use Case	Actors	Description
UC16 ClearCache	ADMIN	Clears all the caches managed by the application.
UC17 ChangeCertificate	ADMIN	This operation allows the admin team to change the SMP certificate. It is called by the admin team in case the SMP certificate has expired and the new one needs to be applied.
UC18 SetProperty	ADMIN	This operation allows the admin team to change DomiSML property in database. New property is taken into account when CRON task refresh the properties
UC19 GetProperty	ADMIN	This operation allows the admin team to verify DomiSML property in database.
UC20 DeleteProperty	ADMIN	This operation allows the admin team to delete DomiSML property from database.

Use Case	Actors	Description
UC21 CreateSubDomain	ADMIN	This operation allows the admin team to create new DomiSML SubDomain.
UC22 UpdateSubDomain	ADMIN	This operation allows the admin team to update DomiSML SubDomain properties.
UC23 GetSubDomain	ADMIN	This operation allows the admin team to read DomiSML SubDomain properties.
UC24 DeleteSubDomain	ADMIN	This operation allows the admin team to delete empty DomiSML SubDomain.
UC25 AddSubDomainCertificate	ADMIN	This operation allows the admin team to add new Domain certificate to DomiSML SubDomain.
UC26 UpdateSubDomainCertificate	ADMIN	This operation allows the admin team to update Domain certificate properties.
UC27 ListSubDomainCertificates	ADMIN	This operation allows the admin team to search for domain certificate by partial certificate DN and by the Subdomain.
UC28 AddDNSRecord	ADMIN	This operation allows the admin team to add new record to DNS for DNS RecordType: A, CNAME and NAPTR.
UC29 DeleteDNSRecord	ADMIN	This operation allows the admin team to delete record from DNS by the DNS name.
UC30 AddTruststoreCertificate	ADMIN	This operation allows the admin team to add certificate to the truststore. Service is needed for adding complete certificate chain to the truststore.
UC31 GetTruststoreCertificate	ADMIN	This operation allows the admin team to retrieve the certificate from the truststore by the alias.
UC32 DeleteTruststoreCertificate	ADMIN	This operation allows the admin team to delete the certificate from the truststore by the alias.
UC33 ListTruststoreCertificateAliases	ADMIN	This operation allows the admin team to retrieve all aliases for the certificates registered in the truststore.
UC34 ManageServiceMetadataPublisher	ADMIN	This operation allows the admin team to Read, Enable, Disable, Delete or Update ServiceMetadataPublisher instances.

3.1.3. Use Cases Detail

Jump to Use Case Details

UC1 Create	UC14	UC27 ListSubDomainCertificates
UC2 Read	ExistsParticipantIdentifier	UC28 AddDNSRecord
UC3 Update	UC15 IsAlive	UC29 DeleteDNSRecord
UC4 Delete	UC16 ClearCache	UC30 AddTruststoreCertificate
UC5 Create	UC17 ChangeCertificate	UC31 GetTruststoreCertificate
UC6 CreateList	UC18 SetProperty	UC32
UC7 Delete	UC19 GetProperty	DeleteTruststoreCertificate
UC8 DeleteList	UC20 DeleteProperty	UC33
UC9 PrepareToMigrate	UC21 CreateSubDomain	ListTruststoreCertificateAliases
UC10 Migrate	UC22 UpdateSubDomain	UC34
UC11 List	UC23 GetSubDomain	ManageServiceMetadataPublisher
UC12	UC24 DeleteSubDomain	
PrepareChangeCertificate	UC25	
UC13	AddSubDomainCertificate	
CreateParticipantIdentifier	UC26	
	UpdateSubDomainCertificate	

Sample Requests and Responses

Besides in the use cases detailed description found in this guide, you can find sample requests and responses in the [SoapUI project](#).

These examples are additional to the ones structure found in the interface's definition (see [Data Model \(WSDL\)](#)) based on their WSDL files and related XSDs.

ManageServiceMetadataService Use Cases

The **ManageServiceMetadataService** interface allows Service Metadata Publishers to manage the metadata held in the Service Metadata Locator service about their service metadata publisher services, such as binding, interface profile and key information.

This interface requires user authentication.

The user's identity is derived from the authentication process and identifies the Service Metadata Publisher associated with the service metadata which is managed via this interface.

Nav to:

→ [UC1 Create](#) → [UC2 Read](#) → [UC3 Update](#) → [UC4 Delete](#)

▼ UC1 Create

UC1 Create

Description

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is ROLE_SMP
C3	The SMP does not already exist in the SML
C4	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.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the Create() operation.
SML	2	Authenticates the user, validates the request, and adds the metadata record into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the creation confirmation.
SMP	5	Use case ends

Alternative Flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response
2.1.3	-	Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	Any other error occurred that prevented the SML to process the request	
2.3.1	SML	Returns an HTTP error 500 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends

Post conditions

-

Successful conditions

The Metadata record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation (E1)
<code>badRequestFault</code> (400)	Returned when the supplied <code>CreateServiceMetadataPublisherService</code> does not contain consistent data (E2)
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request for any reason (E3)

▼ UC2 Read

UC2 Read

Description

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is <code>ROLE_SMP</code>
C3	The SMP already exists in the SML
C4	Input <code>ReadServiceMetadataPublisherService</code> , <code>ServiceMetadataPublisherID</code> : the unique ID of the Service Metadata Publisher for which the record is required.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the <code>Read()</code> operation.
SML	2	Authenticates the user, validates the request, and reads the requested SMP metadata from its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the metadata.

Actor	Step	Description
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response
2.1.3		Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	Any other occurred error preventing SML from processing the request	
2.3.1	SML	Returns an HTTP error 500 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends
E4	The identifier of the SMP could not be found	
2.4.1	SML	Returns an HTTP error 404 as response to the requester
2.4.2	SMP	Receives the error response
2.4.3	-	Use case ends

Post conditions

-

Successful conditions

The Metadata has been provided to the requester.

Output: `ServiceMetadataPublisherService`, the service metadata publisher record, in the form of a `ServiceMetadataPublisherService` data type.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the Create operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied parameter does not contain consistent data(E2).

Error	Description
<code>internalErrorFault</code> (500)	Returned when the SML service was unable to process the request(E3).
<code>notFoundFault</code> (404)	Returned when SMP's identifier was not found(E4).

▼ UC3 Update

UC3 Update

Description

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is <code>ROLE_SMP</code>
C3	The SMP already exists in the SML
C4	Input <code>UpdateServiceMetadataPublishService</code> , <code>ServiceMetadataPublisherService</code> - contains the service metadata for the service metadata publisher, which includes the logical and physical addresses for the SMP (Domain name and IP address).

Basic Flow

Actor	Step	Description
SMP	1	Invokes the Update() operation
SML	2	Authenticates the user, validates the request, and updates the requested metadata record from its configuration database.
SML	3	Returns a positive response to the requester
SMP	4	Receives the update confirmation
-	5	Use case ends

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response

Flow	Actor	Description
2.1.3	-	Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	The identifier of the SMP could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester
2.3.2	SMP	Receives the error response
2.2.3	-	Use case ends
E4	Any other error occurred that prevented the SML to process the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester
2.4.2	SMP	Receives the error response
2.4.3	-	Use case ends

Post conditions

None

Successful conditions

The SMP record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation (E1).
<code>badRequestFault</code> (400)	Returned when the supplied <code>UpdateServiceMetadataPublishService</code> does not contain consistent data (E2).
<code>notFoundFault</code> (404)	Returned when SMP's identifier was not found (E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request(E4).

▼ UC4 Delete

UC4 Delete

Description UC04 Delete

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is <code>ROLE_SMP</code>
C3	The SMP already exists in the SML
C4	Input <code>DeleteServiceMetadataPublisherService</code> , <code>ServiceMetadataPublisherID</code> : the unique ID of the Service Metadata Publisher to delete

Basic Flow

Actor	Step	Description
SMP	1	Invokes the Delete() operation
SML	2	Authenticates the user, validates the request, and deletes the requested metadata record from its configuration database.
SML	3	Returns a positive response to the requester
SMP	4	Receives the deletion confirmation
-	5	Use case ends

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response
2.1.3		Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3		Use case ends
E3	The identifier of the SMP could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester

Flow	Actor	Description
2.4.2	SMP	Receives the error response
2.4.3	-	Use case ends

Post conditions

None

Successful conditions

The SMP record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation (E1).
<code>badRequestFault</code> (400)	Returned when the <code>DeleteServiceMetadataPublisherService</code> supplied does not contain consistent data(E2).
<code>notFoundFault</code> (404)	Returned when the identifier of the SMP was not found(E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request for any reason(E4).

ManageBusinessIdentifierService Use Cases

The `ManageParticipantIdentifier` interface allows Service Metadata Publishers to manage the information in the Service Metadata Locator service relating to individual participant identifiers for which they hold metadata.

This interface requires authentication of the Service Metadata Publisher. The identity of the Service Metadata Publisher derived from the authentication process identifies the Service Metadata Publisher associated with the Participant Identifier(s) which is (are) managed via this interface.

Nav to:

→ [UC5 Create](#) → [UC6 CreateList](#) → [UC7 Delete](#) → [UC8 DeleteList](#) → [UC9 PrepareToMigrate](#) → [UC10 Migrate](#) → [UC11 List](#)

▼ UC5 Create

UC5 Create

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.

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is ROLE_SMP
C3	The SMP does already exist in the SML
C4	The participant does not already exist
C5	Input CreateParticipantIdentifier , + ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP which holds its data.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the Create() operation
SML	2	Authenticates the user, validates the request, and adds the SMP record into its configuration database.
SML	3	Returns a positive response to the requester
SMP	4	Receives the creation confirmation
-	5	Use case ends

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response
2.1.3	-	Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	The SMP could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends
E4	Any other error occurred that prevented the SML to process the request	

Flow	Actor	Description
2.4.1	SML	Returns an HTTP error 500 as response to the requester
2.4.2	SMP	Receives the error response
2.4.3	-	Use case ends

Post conditions

None

Successful conditions

The SMP record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied CreateParticipantIdentifier does not contain consistent data (E2).
<code>notFoundFault</code> (404)	Returned when the identifier of the SMP could not be found (E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request for any reason(E4).

▼ UC6 CreateList

UC6 CreateList

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.

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate.
C2	The role associated to the user is <code>ROLE_SMP</code> .
C3	SMP already exists in the SML.
C4	The participants don't already exist.

Precondition	Description
C5	Input CreateList: ParticipantIdentifierPage , contains the list of Participant Identifiers for the participants which are added to the Service Metadata Locator service. The NextPageIdentifier is absent.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the CreateList() operation
SML	2	Authenticates the user, validates the request, and adds the SMP records into its configuration database.
SML	3	Returns a positive response to the requester
SMP	4	Receives the creation confirmation
-	5	Use case ends

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Receives the error response
2.1.3	-	Use case ends
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	One or several SMP or participants could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends
E4	Any other error occurred that prevented the SML to process the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester
2.4.2	SMP	Receives the error response
2.4.3	-	Use case ends

Post conditions

None

Successful conditions

The SMP records have been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the: • supplied CreateList does not contain consistent data; • number of participants in the list is greater than 100(E2).
<code>notFoundFault</code> (404)	Returned when SMP's or a participants' identifier was not found(E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request(E4).

▼ UC7 Delete

UC7 Delete

Description

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is <code>ROLE_SMP</code> .
C3	SMP already exists in the SML.
C4	The participant already exists.
C5	Input <code>DeleteParticipantIdentifier</code> , <code>ServiceMetadataPublisherServiceForParticipantType</code> - contains the Participant Identifier for a given participant and the identifier of the SMP that publishes its metadata.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the Delete() operation

Actor	Step	Description
SML	2	Authenticates the user, validates the request, and adds the SMP records into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the deletion confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	SMP or participant could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

The SMP record has been deleted from the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1).

Error	Description
badRequestFault (400)	Returned when the supplied DeleteParticipantIdentifier does not contain consistent data (E2).
notFoundFault (404)	Returned when SMP's or a participants' identifier was not found(E3).
internalErrorFault (500)	Returned when the SML service is unable to process the request for any reason(E4).

▼ UC8 DeleteList

UC8 DeleteList

Description

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

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate
C2	The role associated to the user is ROLE_SMP .
C3	SMP already exists in the SML.
C4	The participant already exists.
C5	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.
C6	The number of participants in the list is less than 100.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the DeleteList() operation
SML	2	Authenticates the user, validates the request, and adds the SMP records into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the deletion confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	SMP or participant could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

The SMP records have been deleted from the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the: * supplied DeleteList does not contain consistent data; * number of participants in the list is greater than 100(E2).
<code>notFoundFault</code> (404)	Returned when SMP's or a participants' identifier was not found(E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request(E4).

▼ UC9 PrepareToMigrate

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.

Actors

SMP

Preconditions

Precondition	Description
C1	The user has a valid certificate.
C2	The role associated to the user is ROLE_SMP .
C3	SMP already exists in the SML.
C4	The participant already exists.
C5	Input PrepareMigrationRecord , MigrationRecordType : contains the Migration Key and the Participant Identifier which is about to be migrated from one Service Metadata Publisher to another.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the PrepareToMigrate() operation
SML	2	Authenticates the user, validates the request, and adds the SMP records into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the prepared to migrate confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.

Flow	Actor	Description
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	SMP or participant could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

The SMP record is ready to be migrated into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied <code>PrepateMigrationRecord</code> does not contain consistent data (E2).
<code>notFoundFault</code> (404)	Returned when SMP's or a participants' identifier was not found(E3).
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request(E4).

▼ UC10 Migrate

UC10 Migrate

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.

Actors

SMP

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The role associated to the user is <code>ROLE_SMP</code> .
C3	SMP already exists in the SML.
C4	Participant already exists.
C5	The operation was previously invoked for the supplied Participant Identifier using the same <code>MigrationCode</code> .
C6	Input <code>CompleteMigrationRecord</code> , <code>MigrationRecordType</code> : contains the Migration Key and the Participant Identifier which is to be migrated from one Service Metadata Publisher to another.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the <code>Migrate()</code> operation
SML	2	Authenticates the user, validates the request, and migrates the SMP record into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the migration confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.

Flow	Actor	Description
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	The SMP or migration key could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

The participant identifier has been migrated into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied <code>CompleteMigrationRecord</code> does not contain consistent data (E2).
<code>notFoundFault</code> (404)	Returned when the migration key or the identifier of the SMP could not be found (E3).
<code>internalErrorFault</code> (500)	Returned when the SML service was unable to process the request(E4).

▼ UC11 List

UC11 List

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.

Actors

SMP

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The role associated to the user is ROLE_SMP .
C3	SMP already exists in the SML.
C4	Participant already exists.
C5	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.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the List() operation
SML	2	Authenticates the user, validates the request, and builds the list of SMP from its configuration database.
SML	3	Returns the requested list to the requester.
SMP	4	Receives the requested list.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	The SMP or migration key could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.

Flow	Actor	Description
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

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.

NOTE

The underlying data may be updated between two sequential invocations of **List()**, this means the pages of participant identifiers retrieved can result in a inconsistent set of data.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke the operation(E1).
badRequestFault (400)	Returned when the supplied NextPage does not contain consistent data (E2).
notFoundFault (404)	Returned when the NextPage or the SMP's identifiers were not found(E3).
internalErrorFault (500)	Returned when the SML service was unable to process the request(E4).

BDMSLService Use Cases

This interface describes non-core services which are not defined both in the SML and BDX specifications.

The following paragraphs define the use cases related to the BDMSLService service.

Nav to:

→ [UC12](#) [PrepareChangeCertificate](#) → [UC13](#) [CreateParticipantIdentifier](#) → [UC14](#)
[ExistsParticipantIdentifier](#) → [UC15](#) [IsAlive](#)

▼ UC12 PrepareChangeCertificate

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 DomiSML 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.

Actors

SMP

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The role associated to the user is ROLE_SMP .
C3	The user has the new certificate for the SMP(s).
C4	Input: PrepareChangeCertificate containing the new certificate and the validity start date.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the PrepareChangeCertificate() operation.
SML	2	Authenticates the user, validates the request, and stores the future certificate into its configuration database.
SML	3	Returns the requested list to the requester.
SMP	4	Receives the creation confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	

Flow	Actor	Description
2.2.1	SML	Returns an HTTP error 400 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends
E3	The SMP or migration key could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.

Post conditions

None

Successful conditions

The Metadata record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the: • supplied request does not contain consistent data • new certificate is not valid at the date provided in the migrationDate element • migrationDate is not in the future. • migrationDate is not provided and the "Not Before" date of the new certificate is not in the future • migrationDate is not provided and the "Valid From" is in the past(E2).
<code>internalErrorFault</code> (500)	Returned when the SML service was unable to process the request(E3).

▼ UC13 CreateParticipantIdentifier

UC13 CreateParticipantIdentifier

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.

- `CreateParticipantIdentifier()` operation, this service name can be customized.

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

Actors

SMP

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The role associated to the user is <code>ROLE_SMP</code> .
C3	SMP already exists in the SML.
C4	Participant doesn't exist yet.
C5	Input <code>CreateParticipantIdentifier</code> , <code>ServiceMetadataPublisherServiceForParticipantType</code> : contains the participant's identifier for a given participant and the identifier of the SMP which holds its data. Additional parameter: <code>serviceName</code> .

Basic Flow

Actor	Step	Description
SMP	1	Invokes the <code>CreateParticipantIdentifier()</code> operation.
SML	2	Authenticates the user, validates the request, and adds the SMP record into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the creation confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester.
2.2.2	SMP	Receives the error response.

Flow	Actor	Description
2.2.3	-	Use case ends.
E3	The SMP could not be found	
2.3.1	SML	Returns an HTTP error 404 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.
E4	Any other occurred error preventing SML from processing the request	
2.4.1	SML	Returns an HTTP error 500 as response to the requester.
2.4.2	SMP	Receives the error response.
2.4.3	-	Use case ends.

Post conditions

None

Successful conditions

The SMP record has been created into the SML.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied <code>CreateParticipantIdentifier</code> does not contain consistent data (E2).
<code>notFoundFault</code> (404)	Returned when the SMP's identifier was not found(E3).
<code>internalErrorFault</code> (500)	Returned when the SML service was unable to process the request(E4).

▼ UC14 *ExistsParticipantIdentifier*

UC14 ExistsParticipantIdentifier

Description

The method enables an SMP to verify if it was already registered the participant into the DomiSML.

Actors

SMP

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The role associated to the user is <code>ROLE_SMP</code> .

Precondition	Description
C3	SMP already exists in the SML.
C4	Participant doesn't exist yet.
C5	Input, <code>ServiceMetadataPublisherServiceForParticipantType</code> : contains the Participant Identifier for a given participant and the identifier of the SMP which holds its data.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the <code>ExistsParticipantIdentifier()</code> operation.
SML	2	Authenticates the user, validates the request, and adds the SMP record into its configuration database.
SML	3	Returns participant's query data and exists parameter set to: • TRUE if the entry already exists; • FALSE if the entry doesn't exist.
SMP	4	Receives the response.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	
2.2.1	SML	Returns an HTTP error 400 as response to the requester.
2.2.2	SMP	Receives the error response.
2.2.3	-	Use case ends.
E3	Any other occurring error preventing SML from processing the request	
2.3.1	SML	Returns an HTTP error 500 as response to the requester.
2.3.2	SMP	Receives the error response.
2.3.3	-	Use case ends.

Post conditions

None

Successful conditions

The SML successfully read the database status for the participant.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when the supplied <code>CreateParticipantIdentifier</code> does not contain consistent data (E2).
<code>internalErrorFault</code> (500)	Returned when the SML service was unable to process the request(E3).

▼ UC15 IsAlive

UC15 IsAlive

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.

Actors

SMP
ADMIN
Monitor user

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role <code>ROLE_SMP</code> , <code>ROLE_ADMIN</code> OR <code>ROLE_MONITOR</code> .
C4	Input: None.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the <code>ExistsParticipantIdentifier()</code> operation.
SML	2	Authenticates the user.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the alive confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns no response to the requester
2.1.3	-	Use case ends.
E2	Any other occurring error preventing SML from processing the request	
2.2.1	SML	Returns no response to the requester.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

HTTP 200 OK response sent to the requester.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke the operation (E1).
internalServerError (500)	Returned when the SML service was unable to process the request(E2).

BDMSLSERVICE Use Cases

This interface describes non-core services that are not defined in the SML or BDX specifications.

The following paragraphs define the use cases related to the BDMSLAdminService service.

Nav to:

→ UC16 ClearCache	→ UC19 GetProperty	→ UC22	→ UC24 DeleteSubDomain
→ UC17	→ UC20	UpdateSubDomain	→ UC25
ChangeCertificate	DeleteProperty	→ UC23	AddSubDomainCertificate
→ UC18 SetProperty	→ UC21	GetSubDomain	
	CreateSubDomain		

▼ UC16 ClearCache

UC16 ClearCache

Description

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, to avoid the cost of downloading each time the CRLM for each certificate.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: None.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the ClearCache() operation.
SML	2	Authenticates the user, validates the request, and clears the in-memory cache.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the alive confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other occurring error preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester
2.2.2	SMP	Receives the error response
2.2.3	-	Use case ends

Post conditions

None

Successful conditions

The cache is reset. HTTP 200 OK response sent to the requester.

Failure Conditions (HTTP errors)

No or unspecified type of response (E1, E2).

▼ UC17 ChangeCertificate

UC17 ChangeCertificate

Description

This operation allows the admin team to change the SMP certificate. It is called by the admin team in case the SMP 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.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	The user has the new certificate for the SMP(s).
C4	Input : SMP ID, New certificate public key.

Basic Flow

Actor	Step	Description
SMP	1	Invokes the ChangeCertificate() operation.
SML	2	Authenticates the user, validates the request, and stores the new certificate into its configuration database.
SML	3	Returns a positive response to the requester.
SMP	4	Receives the alive confirmation.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	SMP	Receives the error response.
2.1.3	-	Use case ends.
E2	Request is not valid	

Flow	Actor	Description
2.2.1	SML	Returns an HTTP error 400 as response to the requester.
2.2.2	SMP	Receives the error response.
2.2.3		Use case ends.
E3	Any other occurring error preventing SML from processing the request	
2.3.1	SML	Returns an HTTP error 500 as response to the requester
2.3.2	SMP	Receives the error response
2.3.3	-	Use case ends

Post conditions

None

Successful conditions

New Certificate is stored.

- Output: none.
- HTTP 200 OK expected.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke the operation(E1).
<code>badRequestFault</code> (400)	Returned when(E2): * supplied request does not contain consistent data * new certificate is not valid at the date provided in the <code>migrationDate</code> element * <code>migrationDate</code> is not in the future. * <code>migrationDate</code> is not provided and the "Not Before" date of the new certificate is not in the future * <code>migrationDate</code> is not provided and the "Valid From" is in the past.
<code>internalErrorFault</code> (500)	Returned when the SML service is unable to process the request(E3).

▼ UC18 SetProperty

UC18 SetProperty

Description

This operation allows the admin team to change DomiSML property in database: as passwords, DNS URL, etc. New property is taken into account when CRON task refresh the properties at the same time on all nodes in cluster. CRON tab properties are refreshed only with restart of DomiSML server.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Property name and property value.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the SetProperty () operation.
SML	2	Authenticates the user, validates the request, and ,if property is password, the password is encrypted and stored.
SML	3	Returns returns stored property from database. In case of password property the '***' is returned.
ADMIN	4	Receives the stored property.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3		Use case ends.

Post conditions

None

Successful conditions

- Output, **PropertyType**: the property stored in DomiSML.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke the operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC19 GetProperty

UC19 GetProperty

Description

This operation allows the admin team to retrieve DomiSML property from database, DNS URL, smtp configuration, etc.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Property name.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the GetProperty () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Returns returns stored property from database. In case of password property, the '***' is returned.
ADMIN	4	Receives the stored property.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.

Flow	Actor	Description
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output, **PropertyType**: the property from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC20 DeleteProperty

UC20 DeleteProperty

Description::This operation allows the admin team to delete DomiSML non mandatory properties from database.

Actors

ADMIN

Preconditions

Precon dition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Property name.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the DeleteProperty () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Returns returns stored property from database. In case of password property, the '***' is returned.
ADMIN	4	Receives the stored property.

Actor	Step	Description
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other occurring error preventing the SML to process the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: PropertyType : the deleted property from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1)
<code>internalErrorFault</code> (500)	Returned when the DomiSML service is unable to process the request(E2)

▼ UC21 CreateSubDomain

UC21 CreateSubDomain

Description

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

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Domain name, DNS Zone, DNS Record type, allowed SMP Url scheme, Participant regular expression. Regular expression for validating the Certificate Subject DN, List of allowed Certificate Policy OIDs, Max. allowed number of participants for the domain and for the SMP.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the CreateSubDomain () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Stores the domain in the database.
ADMIN	4	Receives the stored and normalized values for the SubDomain.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester
2.2.2	ADMIN	Receives the error response
2.2.3	-	Use case ends

Post conditions

None

Successful conditions

- Output: SubDomainType: the stored SubDomainData from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation (E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC22 UpdateSubDomain

UC22 UpdateSubDomain

Description

This operation allows the admin team to update DomiSML 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.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Domain name (optional): DNS Record type, allowed SMP Url scheme, Participant regular expression. Regular expression for validationg the Certificate Subject DN, List of allowed Certificate Policy OIDs, Max. allowed number of participants for the domain and for the SMP.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the UpdateSubDomain () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Updates the subdomain data in the database.
ADMIN	4	Receives the stored and normalized values for the SubDomain.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	

Flow	Actor	Description
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: SubDomainType : the stored SubDomainData from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC23 GetSubDomain

UC23 GetSubDomain

Description

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

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Domain name.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the GetSubDomain() operation.
SML	2	Authenticates the user, validates the request.

Actor	Step	Description
SML	3	Gets the subdomain data from the database.
ADMIN	4	Receives the stored values for the SubDomain.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3		Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3		Use case ends.

Post conditions

None

Successful conditions

- Output, **SubDomainType**: the stored SubDomainData from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC24 DeleteSubDomain

UC24 DeleteSubDomain

Description

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

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Domain name.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the DeleteSubDomain () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Delete the subdomain data from database.
ADMIN	4	Receives the deleted SubDomain data.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurred that prevented the SML to process the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output, **SubDomainType**: the deleted SubDomainData from DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).

Error	Description
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC25 AddSubDomainCertificate

UC25 AddSubDomainCertificate

Description

This operation allows the admin team to add new Domain certificate to DomiSML SubDomain. Certificate can be flagged as RootPKI certificate and as Admin certificate. Admin certificate can be only the certificate which is not flagged as RootPKI certificate. If truststore authentication is enabled, then the certificate is automatically added to the truststore. For the certificate to be fully trusted, the whole PKI chain also has to be added to the truststore using the operation: AddTruststoreCertificate.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Certificate (PEM/DER), SubDomain name, IsRootPKI certificate, Is Admin certificate, CRL distribution URL.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the AddSubDomainCertificate () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Adds the new SubDomain certificate to the subdomain.
ADMIN	4	Receives added Certificate SubDomain data.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.

Flow	Actor	Description
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: SubDomainCertificateType: the new SubDomain Certificate Data stored to DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation (E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC26 UpdateSubDomainCertificate

UC26 UpdateSubDomainCertificate

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.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Certificate ID, Optional: SubDomain name, Is Admin certificate, CRL distribution URL.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the UpdateSubDomainCertificate () operation.

Actor	Step	Description
SML	2	Authenticates the user, validates the request.
SML	3	Update data to SubDomain certificate.
ADMIN	4	Receives stored Certificate SubDomain data.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: SubDomainCertificateType : the new SubDomain Certificate Data stored to DomiSML database.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC27 ListSubDomainCertificates

UC27 ListSubDomainCertificates

Description

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

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: Optional: Partial Certificate ID, SubDomain name.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the ListSubDomainCertificates () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Returns list of SubDomain certificate data according to search parameters.
ADMIN	4	Receives list of SubDomain certificate data.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: List of SubDomainCertificateTypes which match search criteria.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC28 AddDNSRecord

UC28 AddDNSRecord

Description

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

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: DNS Record Type, DNS Record Name, DNS Zone, Value and service name for NAPTR Type.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the AddDNSRecordType () operation.
SML	2	Authenticates the user, validates the request.
SML	3	Stores DNS record to SML database and if DNS integration is on inserts record to DNS server.
ADMIN	4	Receives stored DNS data.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.

Flow	Actor	Description
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: DNSRecord data which are stored to database and optionally to DNS.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation (E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC29 DeleteDNSRecord

UC29 DeleteDNSRecord

Description

This operation allows the admin team to delete record from DNS by the DNS name. If there are multiple DNS records with the same name in database and DNS server, all of them are deleted.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: DNS Record Name, DNS Zone.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the DeleteDNSRecordType () operation.
SML	2	Authenticates the user, validates the request.

Actor	Step	Description
SML	3	Deleted DNS records from SML database and DNS server if DNS integration is on.
ADMIN	4	Receives list of deleted DNS data from database.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: List of DNSRecord data, which are deleted from database.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1).
<code>internalErrorFault</code> (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC30 AddTruststoreCertificate

UC30 AddTruststoreCertificate

Description

This operation allows the admin team to add X509 certificate to the truststore.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: X509Certificate.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the AddTruststoreCertificate operation.
SML	2	Authenticates the user, validates the request.
SML	3	Adds the certificate in the Truststore and generates a new alias if not provided in the request.
ADMIN	4	Receives inserted certificate with alias.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: Certificate with alias.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).

▼ UC31 GetTruststoreCertificate

UC31 GetTruststoreCertificate

Description

This operation allows the admin team to retrieve X509 certificate from the truststore by the certificate alias.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role ROLE_ADMIN .
C3	Input: certificate alias.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the GetTruststoreCertificate operation.
SML	2	Authenticates the user, validates the request.
SML	3	Retrieve certificate from Truststore.
ADMIN	4	Receives certificate with alias.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: certificate with alias.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1).
<code>internalErrorFault</code> (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC32 DeleteTruststoreCertificate

UC32 DeleteTruststoreCertificate

Description

This operation allows the admin team to delete X509 certificate from the truststore.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role <code>ROLE_ADMIN</code> .
C3	Input: certificate alias.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the DeleteTruststoreCertificate operation.
SML	2	Authenticates the user, validates the request.
SML	3	Deletes the certificate from the truststore.
ADMIN	4	Receives deleted certificate with alias.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.

Flow	Actor	Description
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: Deleted X509 Certificate and alias.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation(E1).
<code>internalErrorFault</code> (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC33 ListTruststoreCertificateAliases

UC33 ListTruststoreCertificateAliases

Description

This operation allows the admin team to list all X509 certificates in the truststore.

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.
C2	The user has the role <code>ROLE_ADMIN</code> .
C3	Input: search parameter: null or partial certificate alias.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the ListTruststoreCertificate operation.
SML	2	Authenticates the user, validates the request.
SML	3	Retrieves the aliases which match the search parameter. If the search parameter is not given, then the operation returns all certificates.

Actor	Step	Description
ADMIN	4	Receives truststore aliases which matched search parameter.
-	5	Use case ends.

Alternative flows

None

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: List of truststore aliases.

Failure Conditions (HTTP errors)

Error	Description
unauthorizedFault (401)	Returned when the caller is not authorized to invoke this operation(E1).
internalErrorFault (500)	Returned when the DomiSML service is unable to process the request(E2).

▼ UC34 ManageServiceMetadataPublisher

UC34 ManageServiceMetadataPublisher

Description

Actors

ADMIN

Preconditions

Precondition	Description
C1	User has a valid certificate.

Precondition	Description
C2	The user has the role ROLE_ADMIN .
C3	Input: Action, SMP Id, Email address, Domain Zone, Certificate Id of the owner, Optional: Physical address a Logical address of SMP in case of Update action.

Basic Flow

Actor	Step	Description
ADMIN	1	Invokes the ManageServiceMetadataPublisher operation.
SML	2	Authenticates and Authorize the user, validates the request.
SML	3	Creates task for action (Enable, Disable, Delete or Update) executes action asynchronously.
ADMIN	5	Receives SOAP response with the task description and status.
ADMIN	4	Receives email with tasks status when the task is completed.
-	5	Use case ends.

Alternative flows

Alternative Flow

Actor	Step	Description
ADMIN	1	Invokes the ManageServiceMetadataPublisher operation.
SML	2	Authenticates and Authorize the user, validates the request.
SML	3	If action is READ it retrieves the SMP the data from the database.
ADMIN	4	Receives the current SMP data from the database.
-	5	Use case ends.

Exception flows

Flow	Actor	Description
E1	The user is not authorized	
2.1.1	SML	Returns an HTTP error 401 as response to the requester.
2.1.2	ADMIN	Receives the error response.
2.1.3	-	Use case ends.
E2	Any other error occurring preventing SML from processing the request	
2.2.1	SML	Returns an HTTP error 500 as response to the requester.
2.2.2	ADMIN	Receives the error response.
2.2.3	-	Use case ends.

Post conditions

None

Successful conditions

- Output: The SMP is enabled/disabled/deleted/updated.

Failure Conditions (HTTP errors)

Error	Description
<code>unauthorizedFault</code> (401)	Returned when the caller is not authorized to invoke this operation (E1).
<code>internalErrorFault</code> (500)	Returned when the DomiSML service is unable to process the request(E2).

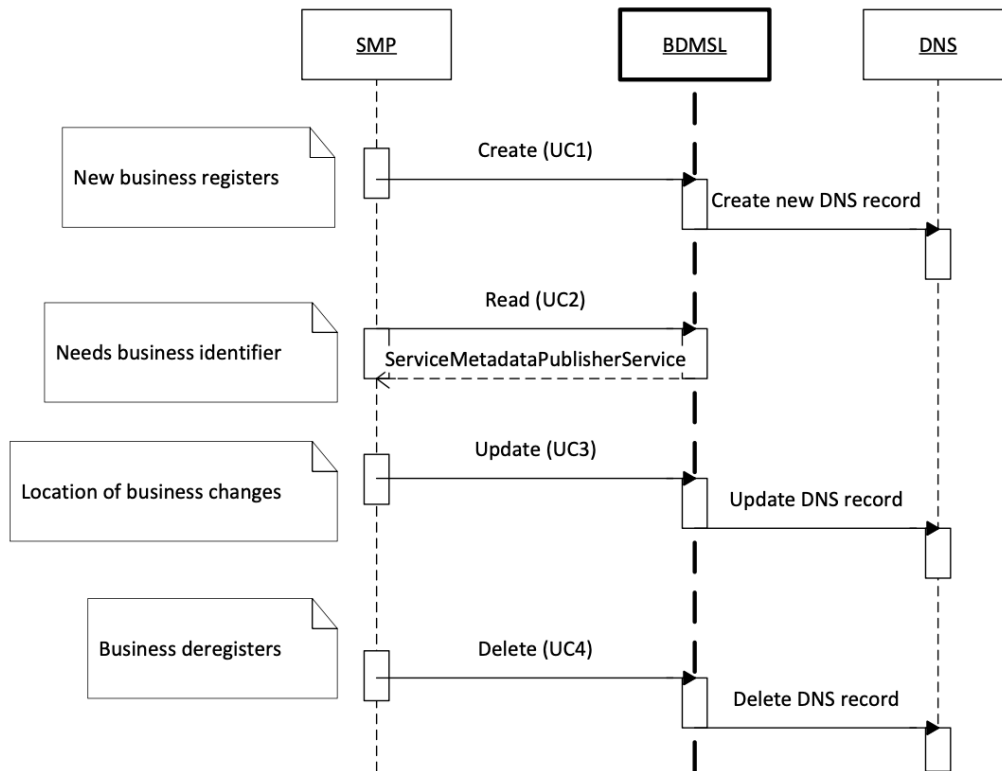
3.2. Interface Behavioural Specification

3.2.1. Sequence diagrams

ManageServiceMetadataService

The ManageServiceMetadata interface allows Service Metadata Publishers to manage the metadata held in the Service Metadata Locator service about their service metadata publisher services, e.g., binding, interface profile and key information. This interface requires authentication of the user. The identity of the user derived from the authentication process identifies the Service Metadata Publisher associated with the service metadata which is managed via this interface. The ManageServiceMetadata interface has the following operations:

- Create
- Read
- Update
- Delete



PEPPOL Transport Infrastructure - Service Metadata Locator (SML)

SEE ALSO

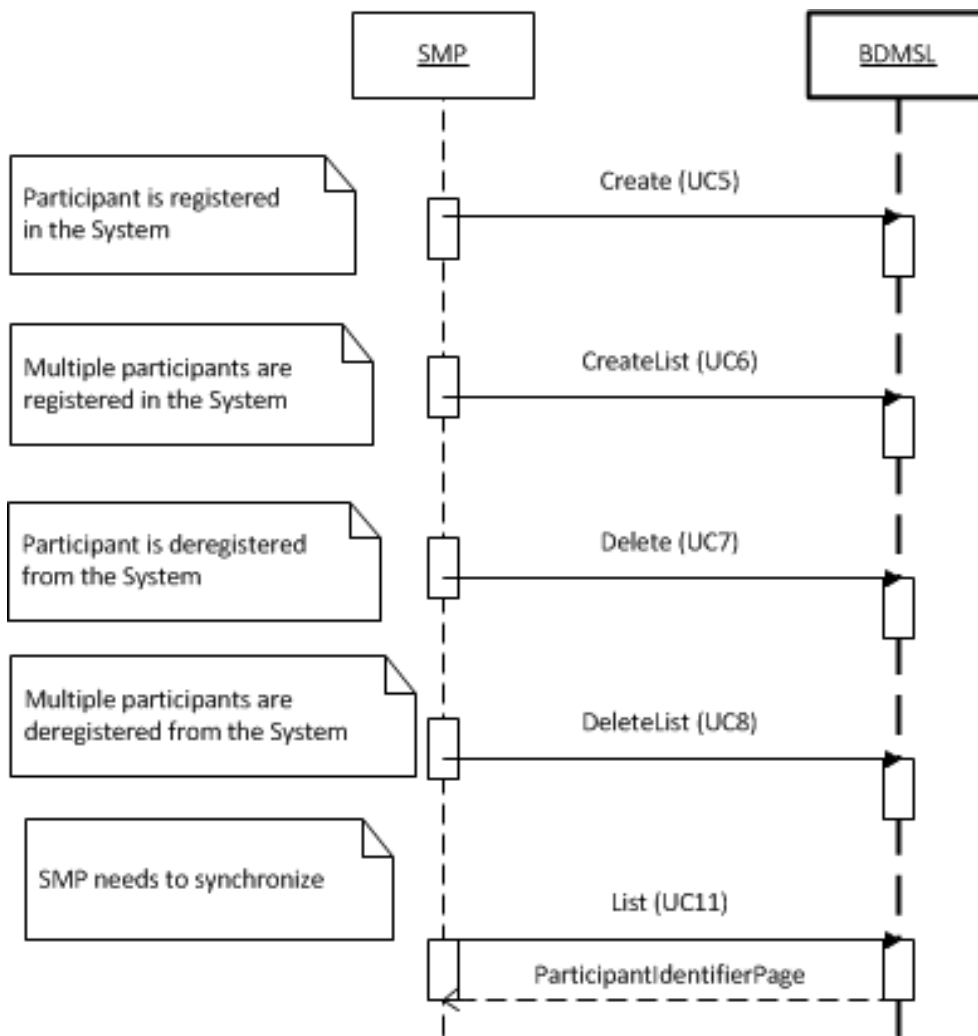
This document defines the profiles for the discovery and management interfaces for the Business Document Exchange Network (BUSDOX) Service Metadata Locator service. The Service Metadata Locator service exposes three interfaces: Service Metadata discovery, Manage participant identifiers and Manage service metadata interfaces.

ManageBusinessIdentifierService

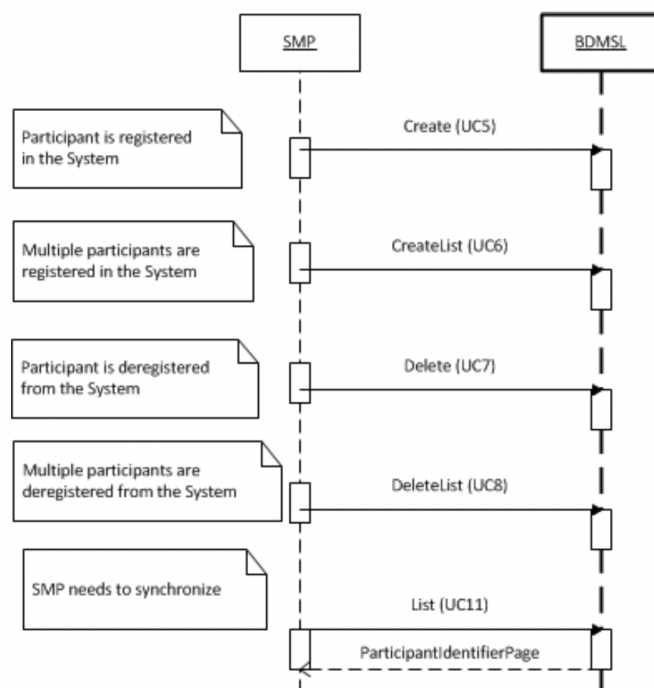
The ManageParticipantIdentifier interface has the following operations:

- Create
- CreateList
- Delete
- DeleteList
- PrepareToMigrate
- Migrate
- List

These services are listed in the sequence diagram below:

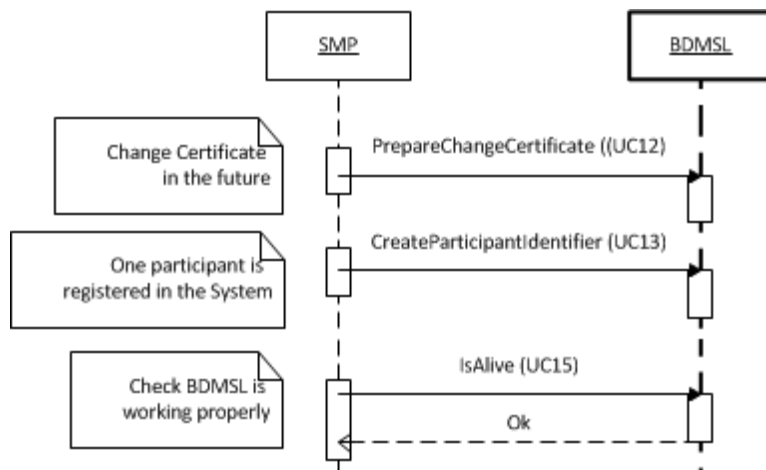


The usage of the services related to the SMP migration process – involving more than a single step like the others above – are shown in the sequence diagram below:



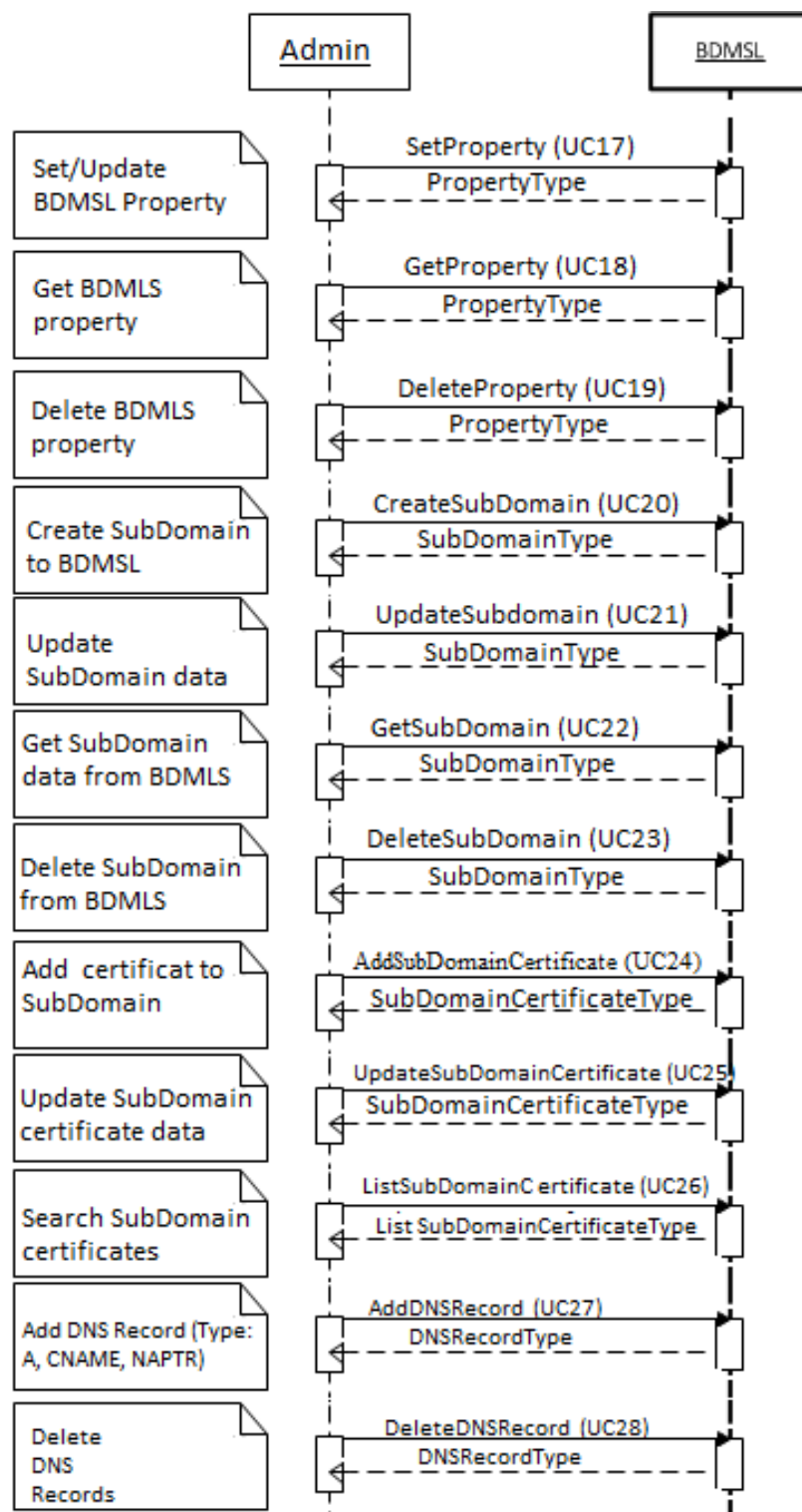
BDMSLSERVICE

This interface describes non-core services that are not defined in the SML or BDX specifications.



BDMSLAdminService

This interface describes non-core administration services that are not defined in the SML or BDX specifications.



3.2.2. Data Model (WSDL)

The interface data model of the DomiSML is described using some conventions.

▼ Model Description Conventions

1

One paragraph for each of the 3 web services will introduce all their operations.

2

One paragraph for each operation will specify their Input and Output structures and the fault

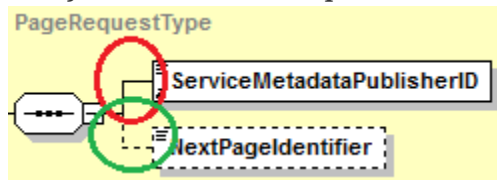
that these operations may return.

In some cases, there is no argument, in which case there is no related structure (the text below will mention "none" in those cases).

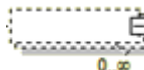
3

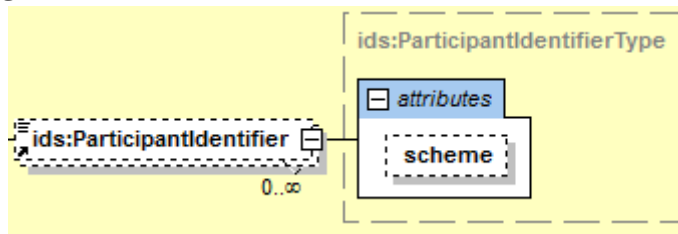
For each input or output structure, the related XSD structure is detailed in a graphically, specifying:

- The arborescence structure;
- The mandatory (solid lines) and optional (dashed line) fields.



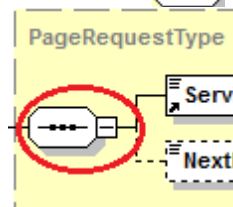
Example:

- The repeated fields and their cardinality (icon  with min/max indication on the bottom right).



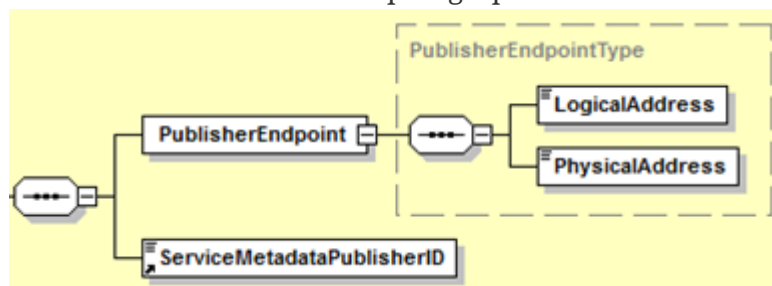
Example:

- The sequences (icon .



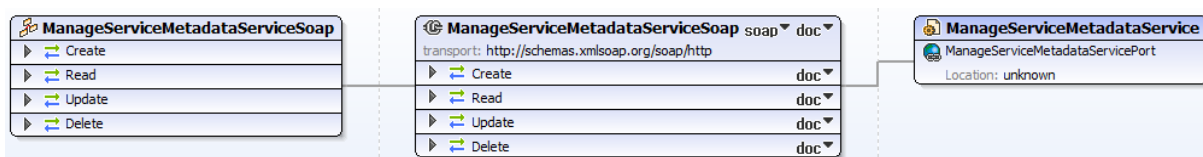
Example:

- The leave attributes as defined in the first paragraph.



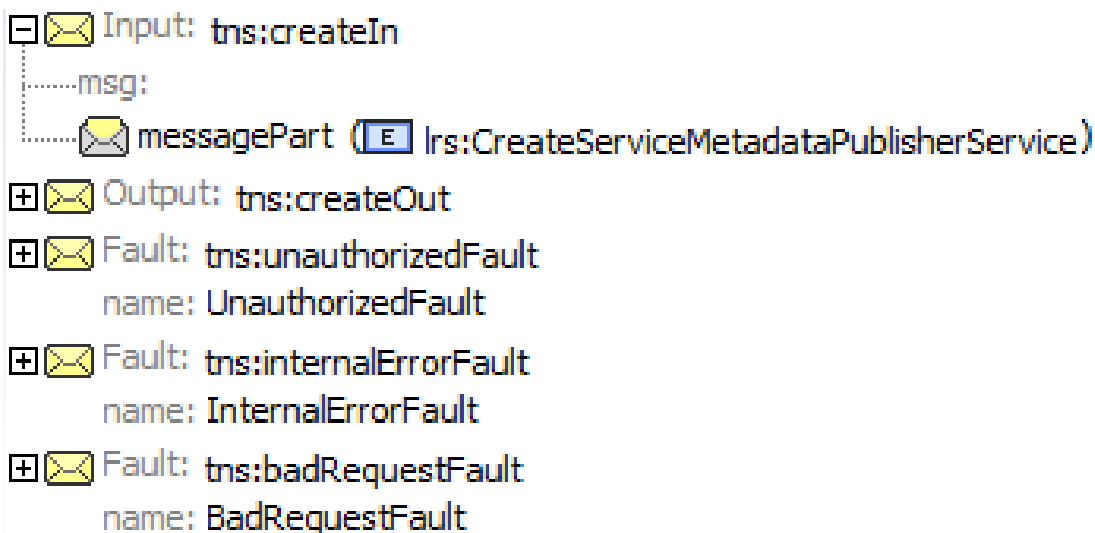
Examples (*):

ManageServiceMetadataService WSDL Model

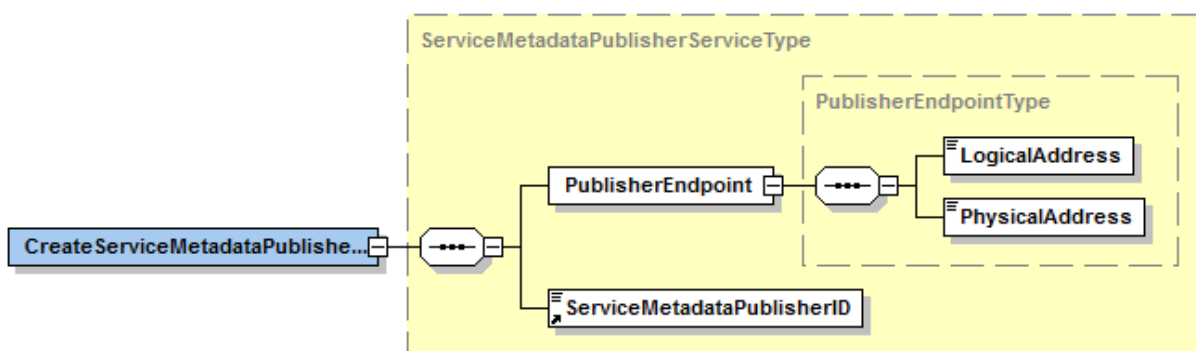


Operations Signatures

▼ Create() Signature



▼ Create() Input



Argument: `PublisherEndpoint.LogicalAddress`

Description

The logical address of the endpoint (Domain name).

Format/XSD/Xpath

xs:anyURI
ServiceMetadataLocatorTypes-1.0.xsd
/[**local-name()**='schema']/[**local-name()**='complexType' and
@name='PublisherEndpointType']/[**local-name()**='sequence']/[**local-name()**='element'
and @name='LogicalAddress']

Constraint

- Must not be null or empty.
- Must be valid and well formatted.

Argument: `PublisherEndpoint.PhysicalAddress`

Description

IP Address of the endpoint.

Format/XSD/Xpath

Constraint

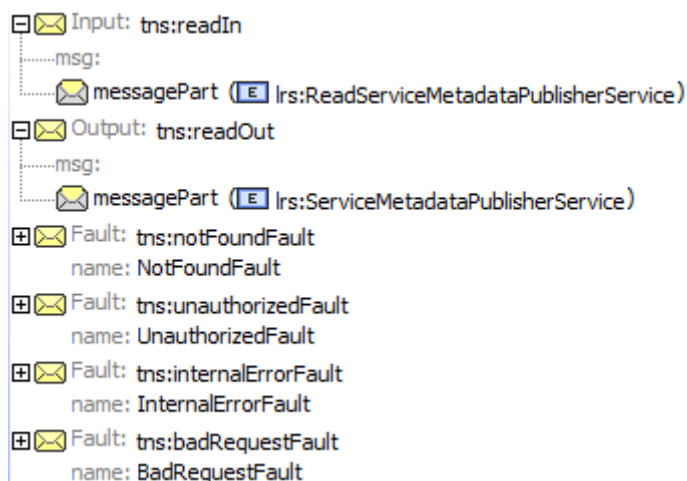
Argument: <code>PublisherEndpoint.PhysicalAddress</code>	
xs:string ServiceMetadataLocatorTypes-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType' and @name='PublisherEndpointType']/[local-name()='sequence']/[local-name()='element' and @name='PhysicalAddress']</code>	This physical address is used as the ALIAS on the CNAME DNS record by default. However a NAPTR DNS record is also provided in order to give the possibility to process regular expressions for accessing the domain if necessary. NAPTR records are 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. • Must not be null or empty. • Must be valid according to IPv4 and well formatted.

Argument: <code>ServiceMetadataPublisherID</code>	
Description Unique identifier of the SMP.	
Format/XSD/Xpath	Constraint
xs:string ServiceMetadataLocatorTypes-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element' and @ref='ServiceMetadataPublisherID']</code>	In <code>ManageServiceMetadata</code> service, this establishes the link with the managing SMP of the participant as defined by in the <code>ManageBusinessIdentifier</code> service.

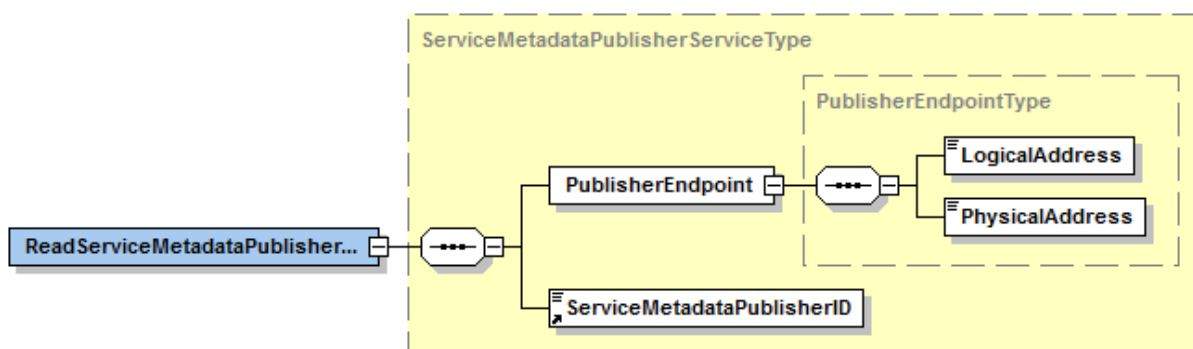
▼ *Create() Output*

None

▼ *Read() Signature*



▼ Read() Input



Argument: `PublisherEndpoint.LogicalAddress`

Description

The logical address of the endpoint(Domain name).

Format/XSD/Xpath	Constraint
xs:anyURI ServiceMetadataLocatorTypes-1.0.xsd / [local-name()='schema'] / [local-name()='complexType'] and @name='PublisherEndpointType'// [local-name()='sequence'] / * [local-name()='element'] and @name='LogicalAddress'	• Must not be null or empty. • Must be valid and well formatted.

Argument: `PublisherEndpoint.PhysicalAddress`

Description

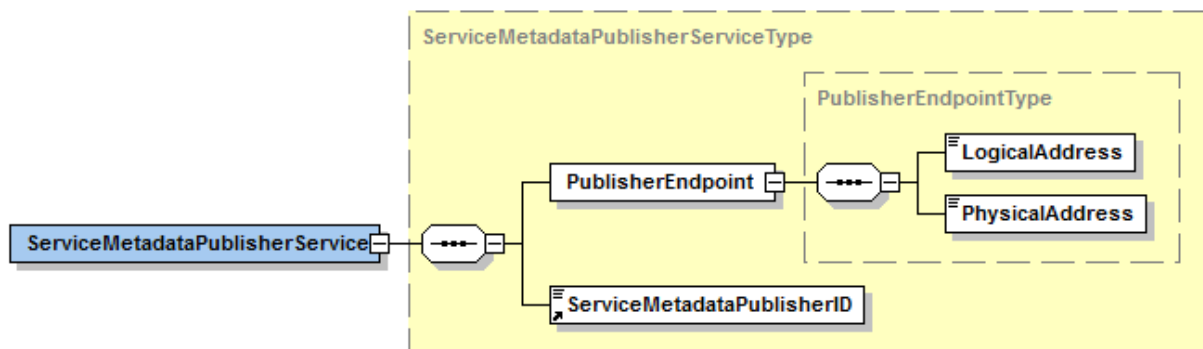
IP Address of the endpoint

Format/XSD/Xpath	Constraint
------------------	------------

Argument: <i>PublisherEndpoint.PhysicalAddress</i>	
xs:string ServiceMetadataLocatorTypes-1.0.xsd / [local-name()='schema'] /[local-name()='complexType' and @name='PublisherEndpointType']/*[local-name()='sequence']/*[local-name()='element' and @name='PhysicalAddress']	This physical address is used as the ALIAS on the CNAME DNS record by default. However a NAPTR DNS record is also provided in order to give the possibility to process regular expressions for accessing the domain if necessary. NAPTR records are 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. • Must not be null or empty. • Must be valid according to IPv4 and well formatted.

Argument: <i>ServiceMetadataPublisherID</i>	
Description Unique identifier of the SMP.	
Format/XSD/Xpath	Constraint
xs:string ServiceMetadataLocatorTypes-1.0.xsd / [local-name()='schema'] /[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']// [local-name()='sequence'] /[local-name()='element' and @ref='ServiceMetadataPublisherID']	In ManageServiceMetadata service, this establishes the link with the managing SMP of the participant as defined by in the ManageBusinessIdentifier service.

▼ *Read() Output*



Argument: PublisherEndpoint.LogicalAddress**Description**

The logical address of the endpoint (Domain name).

Format/XSD/Xpath

xs:anyURI
ServiceMetadataLocatorTypes-1.0.xsd
/[**local-name()**='schema']/[local-name()='complexType' and
@name='PublisherEndpointType']/[**local-name()**='sequence']/[local-name()='element'
and @name='LogicalAddress']

Constraint

Must not be null or empty.
Must be valid and well formatted.

Argument: PublisherEndpoint.PhysicalAddress**Description**

IP Address of the endpoint.

Format/XSD/Xpath

xs:string
ServiceMetadataLocatorTypes-1.0.xsd
/[**local-name()**='schema']/[local-name()='complexType' and
@name='PublisherEndpointType']/*[local-name()='sequence']/ *[local-name()='element'
and @name='PhysicalAddress']

Constraint

This physical address is used as the ALIAS on the CNAME DNS record by default. However a NAPTR DNS record is also provided in order to give the possibility to process regular expressions for accessing the domain if necessary.
NAPTR records are 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.

- Must not be null or empty.
- Must be valid according to IPv4 and well formatted.

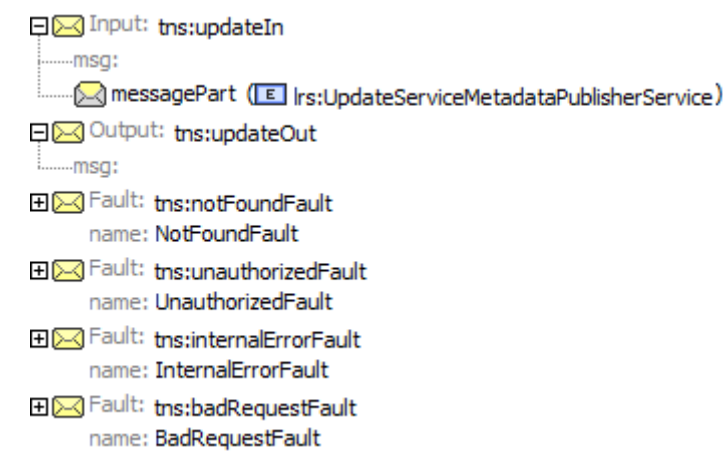
Argument: ServiceMetadataPublisherID**Description**

Unique identifier of the SMP.

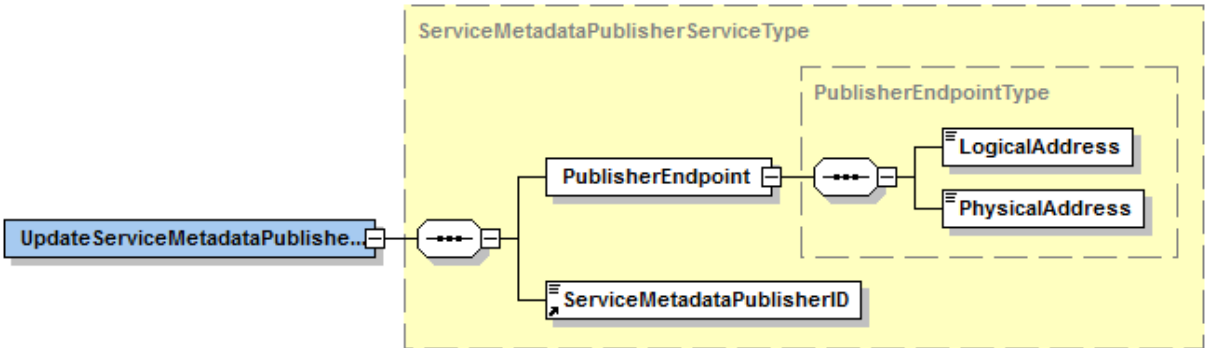
Format/XSD/Xpath**Constraint**

Argument: <i>ServiceMetadataPublisherID</i>	
xs:string ServiceMetadataLocatorTypes-1.0.xsd / [local-name()='schema'] /[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']// [local-name()='sequence'] /[local-name()='element' and @ref='ServiceMetadataPublisherID']	In <i>ManageServiceMetadata</i> service, this establishes the link with the managing SMP of the participant as defined by in the ManageBusinessIdentifier service.

▼ *Update()* Signature



▼ *Update()* Input



Argument: <i>ServiceMetadataPublisherID</i>	
Description The logical address of the endpoint (Domain name).	
Format/XSD/Xpath	Constraint
xs:anyURI ServiceMetadataLocatorTypes-1.0.xsd / [local-name()='schema'] /[local-name()='complexType' and @name='PublisherEndpointType']// [local-name()='sequence'] /	• Must not be null or empty. • Must be valid and well formatted.

Argument: `PublisherEndpoint.PhysicalAddress`

Description

IP Address of the endpoint.

Format/XSD/Xpath

xs:string
ServiceMetadataLocatorTypes-1.0.xsd
/[**local-name()='schema'**]/[local-name()='complexType' and
@name='PublisherEndpointType']/*[local-name()='sequence']/ *[local-name()='element'
and @name='PhysicalAddress']

Constraint

This physical address is used as the ALIAS on the CNAME DNS record by default. However a NAPTR DNS record is also provided in order to give the possibility to process regular expressions for accessing the domain if necessary.

NAPTR records are 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.

- Must not be null or empty.
- Must be valid according to IPv4 and well formatted.

Argument: `ServiceMetadataPublisherID`

Description:Unique identifier of the SMP.

Format/XSD/Xpath

xs:string
ServiceMetadataLocatorTypes-1.0.xsd
/[**local-name()='schema'**]/[local-name()='complexType' and
@name='ServiceMetadataPublisherServiceForParticipantType']//[**local-name()='sequence'**]/[local-name()='element'
and @ref='ServiceMetadataPublisherID']

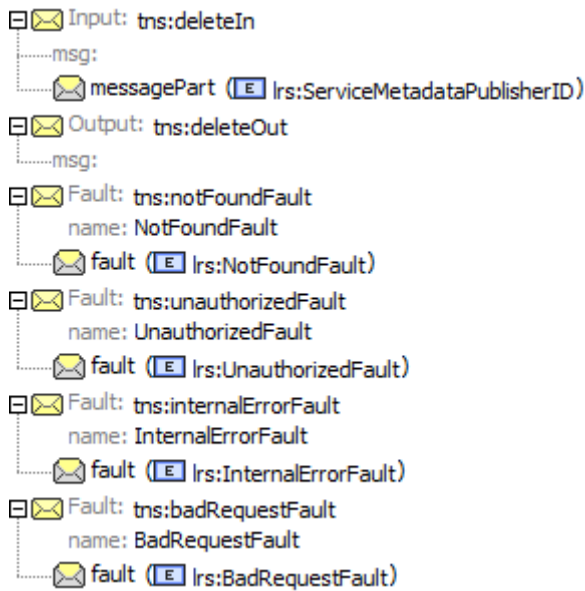
Constraint

In `ManageServiceMetadata` service, this establishes the link with the managing SMP of the participant as defined by in the `ManageBusinessIdentifier` service.

▼ *Update() Output*

None

▼ *Delete() Signature*



▼ Delete() Input

ServiceMetadataPublisherID

Argument: ServiceMetadataPublisherID

Description

Unique identifier of the SMP.

Format/XSD/Xpath

xs:string
ServiceMetadataLocatorTypes-1.0.xsd
/[local-name()='schema']/[local-name()='complexType' and
@name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element' and @ref='ServiceMetadataPublisherID']

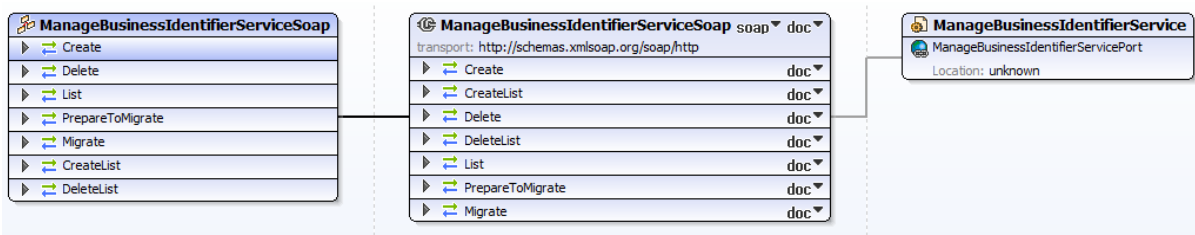
Constraint

In ManageServiceMetadata service, this establishes the link with the managing SMP of the participant as defined by in the ManageBusinessIdentifier service.

▼ Delete() Output

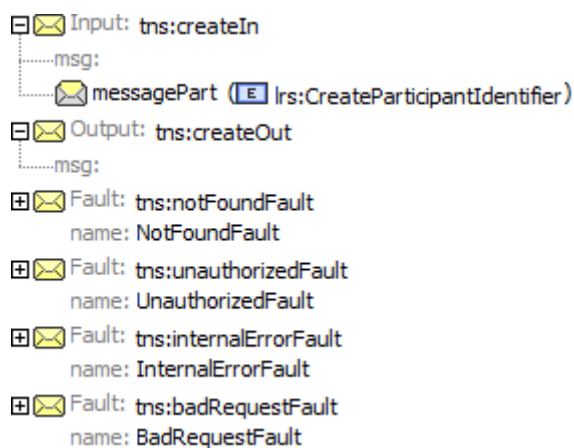
None

ManageBusinessIdentifierService WSDL Model

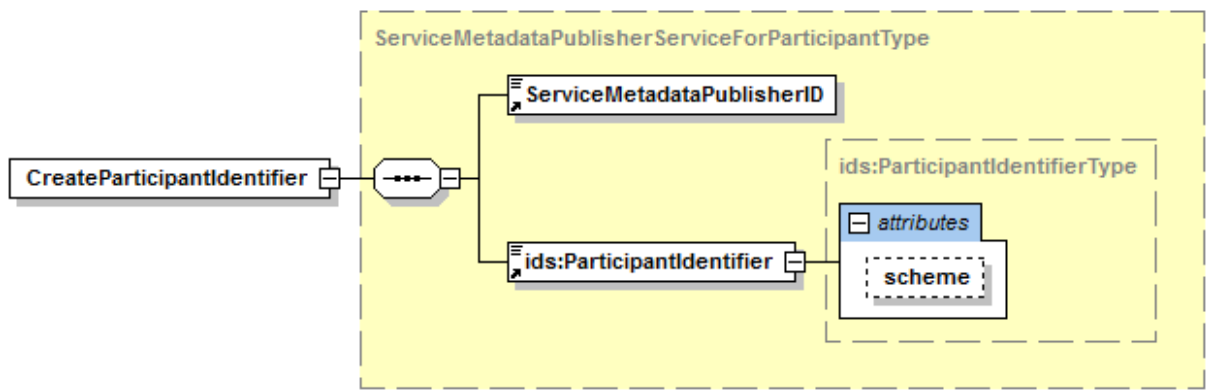


Operations Signatures

▼ Create() Signature



▼ Create() Input



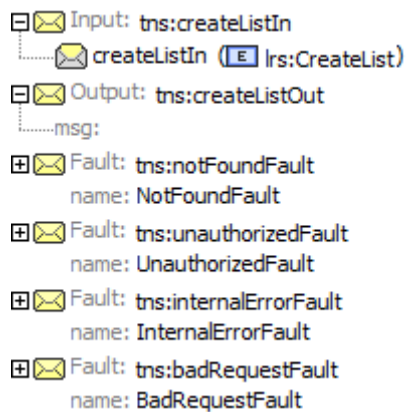
Argument: <i>ServiceMetadataPublisherID</i>	
Description Unique identifier of the SMP.	
Format/XSD/Xpath	Constraint
xs:string ServiceMetadataLocatorTypes-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>/[local-name()='sequence']/[local-name()='element']</code> and <code>@name='ServiceMetadataPublisherServiceForParticipantType'//</code> <code>name()='sequence']/[local-name()='element']</code> and <code>@ref='ServiceMetadataPublisherID']</code>	In <code>ManageServiceMetadata</code> service, this establishes the link with the managing SMP of the participant as defined by in the <code>ManageBusinessIdentifier</code> service.

Argument: <i>ParticipantIdentifier</i>	
Description Business unique identifier of the Participant.	
Format/XSD/Xpath	Constraint

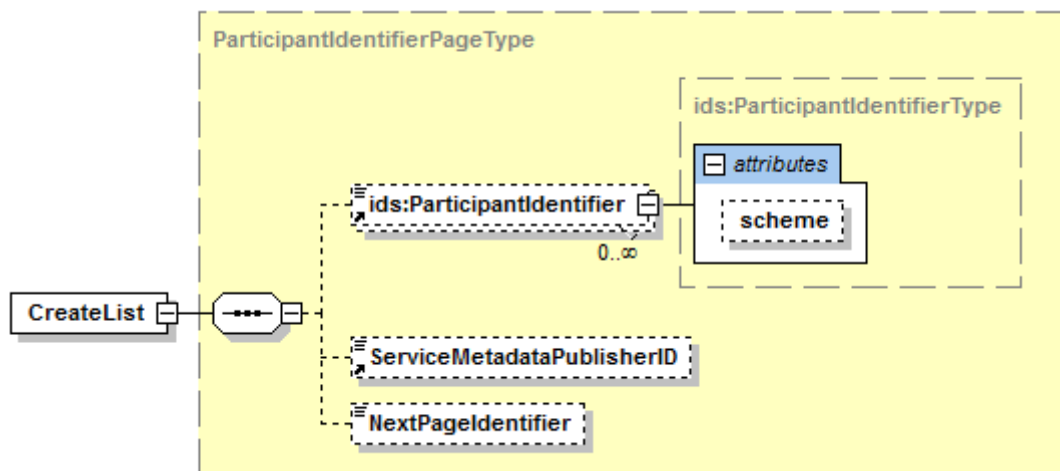
Argument: <code>ParticipantIdentifier</code>	
Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc...	The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ParticipantIdentifierType']</code>

Argument: <code>ParticipantIdentifier.scheme</code>	
Description The scheme of the participant identifier.	
Format/XSD/Xpath	Constraint
Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. + Examples: iso6523-actorid-upis, busdox-actorid-upis Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute</code>	• May not exceed 25 characters. • Must match the two following pattern: <code>-[a-zA-Z0-9]-[a-zA-Z0-9]</code> Which defines the following parts: <code><domain>-<identifier_area>-<identifier_type></code>

- ▼ *Create() Output*
None
- ▼ *CreateList() Signature*



▼ CreateList() Input



NOTE the **NextPageIdentifier** is absent.

Argument: **ParticipantIdentifier** (0..n)

Business unique identifier of the Participant. Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.

Format/XSD/Xpath

Constraint

Argument: ParticipantIdentifier (0..n)	
The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd <pre> /[@local-name()='schema']/ [local- name()='complexType' and @name='ParticipantIdentifierType'] </pre>	- Must be unique. - May not exceed 50 characters and must be at least 1 character long. - May only contain ASCII characters. - Participant identifier must be trimmed. If the scheme refers to the (default) PEPPOL participant identifier scheme (iso6523-actorid-upis) and the participant identifier is not a wildcard (*), then the issuing agency specified in the ParticipantIdentifier (as its leading part before ':') must be included in the official list (see List of Valid PEPPOL Issuing Agencies). NOTE The participant identifier is case insensitive.

Argument: ParticipantIdentifier.scheme (0..n)	
The scheme of the participant identifier.	
Format/XSD/Xpath	Constraints
Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. Examples: iso6523-actorid-upis, busdox-actorid-upis Identifiers-1.0.xsd <pre> /[@local-name()='schema']/ [local- name()='complexType' and @name='ProcessIdentifierType'] /xs:simpleContent/xs:extension/xs:attribute </pre>	May not exceed 25 characters. Must match the two following pattern: -[a-zA-Z0-9]-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier_area>-<identifier_type>.

Argument: ServiceMetadataPublisherID
Description Unique identifier of the SMP.

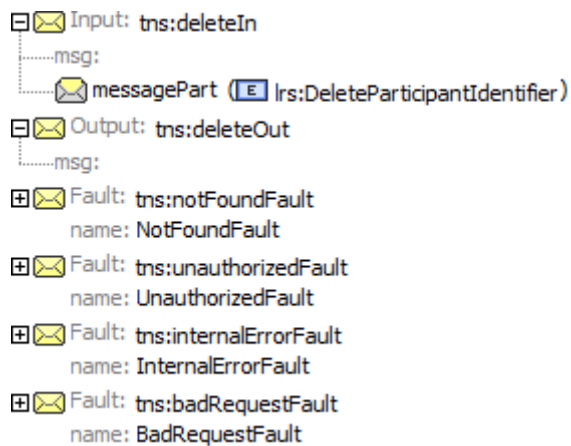
Argument: <i>ServiceMetadataPublisherID</i>	
Format/XSD/Xpath	Constraints
<i>xs:string</i> <i>ServiceMetadataLocatorTypes-1.0.xsd</i> <code>/[local-name()='schema']/</code> [local-name()='complexType' and <code>@name='ServiceMetadataPublisherServiceForParticipantType']//</code> [local-name()='sequence']/ <code>[local-name()='element' and</code> <code>@ref='ServiceMetadataPublisherID']</code>	In <i>ManageServiceMetadata</i> service, this establishes the link with the managing SMP of the participant as defined by in the <i>ManageBusinessIdentifier</i> service.

Argument: <i>NextPageIdentifier</i>	
Description This argument is not used in this context.	
Format/XSD/Xpath	Constraints
n/aa	* Must be null.

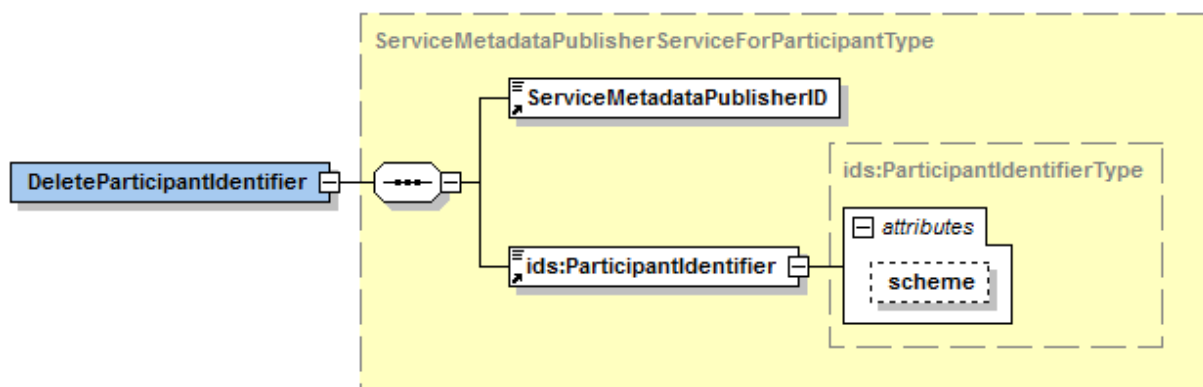
▼ *CreateList()* Output

None

▼ *Delete()* Signature



▼ *Delete()* Input



Argument: <i>ServiceMetadataPublisherID</i>	
Description Unique identifier of the SMP.	
Format/XSD/Xpath	Constraint
xs:string ServiceMetadataLocatorTypes-1.0.xsd /[local-name()='schema']/[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element' and @ref='ServiceMetadataPublisherID']	In <i>ManageServiceMetadata</i> service, this establishes the link with the managing SMP of the participant as defined by in the ManageBusinessIdentifier service.

Argument: <i>ParticipantIdentifier</i>	
Description Business unique identifier of the Participant.	
Format/XSD/Xpath	Constraint
Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.	The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure . Example: 0088:4035811991014 Identifiers-1.0.xsd /[local-name()='schema']/[local-name()='complexType' and @name='ParticipantIdentifierType']

Argument: <i>ParticipantIdentifier.scheme</i>	
Description The scheme of the participant identifier.	

Argument: `ParticipantIdentifier.scheme`

Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance.

Examples: iso6523-actorid-upis, busdox-actorid-upis

Identifiers-1.0.xsd

```
/[local-name()='schema']/[local-
```

```
name()='complexType' and
```

```
@name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute
```

May not exceed 25 characters.

Must match the two following pattern:

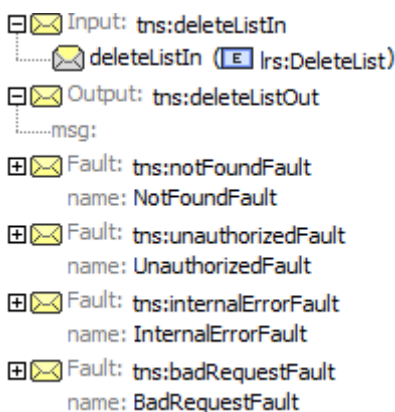
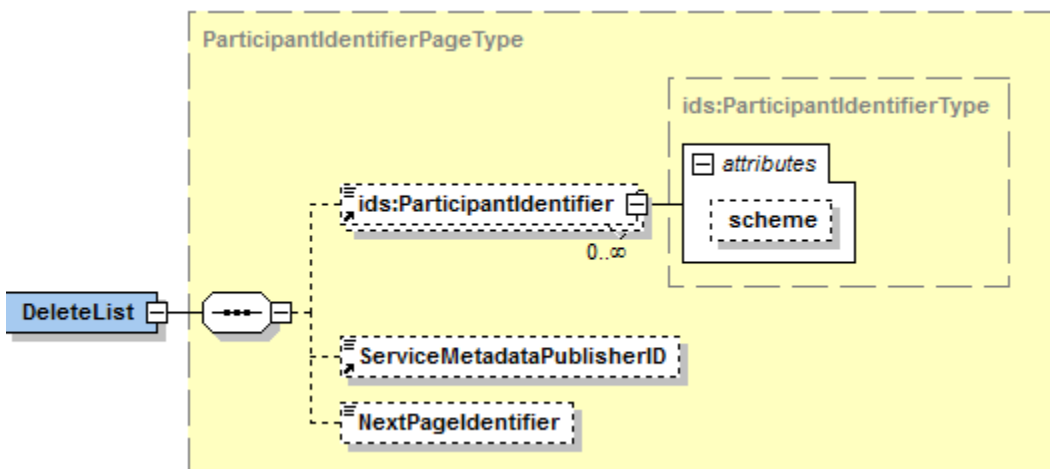
`-[a-zA-Z0-9]-[a-zA-Z0-9]`

+ Which defines the following parts:

`<domain>-<identifier_area>-<identifier_type>`

▼ Delete() Output

None

▼ DeleteList() Signature**▼ DeleteList() Input**

NOTE the `NextPageIdentifier` is absent.

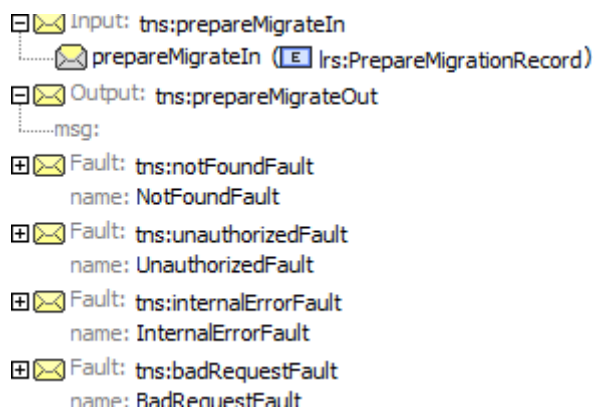
Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier (0..n)	Business unique identifier of the Participant. Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.	The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> <code>@name='ParticipantIdentifierType'</code> and	Must be unique. May not exceed 50 characters and must be at least 1 character long. + May only contain ASCII characters. + Participant identifier must be trimmed. + If the scheme refers to the (default) PEPPOL participant identifier scheme (iso6523-actorid-upis) and the participant identifier is not a wildcard (*), then the issuing agency specified in the ParticipantIdentifier (as its leading part before ':') must be included in the official list (see List of Valid PEPPOL Issuing Agencies). NOTE: The participant identifier is case insensitive.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme (0..n)	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. Examples: iso6523-actorid-upis, busdox-actorid-upis + Identifiers-1.0.xsd + /[local-name()='schema']/[local-name()='complexType' and @name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute	May not exceed 25 characters. Must match the two following pattern: -[a-zA-Z0-9]-[a-zA-Z0-9] + Which defines the following parts: <domain>-<identifier Area>-<identifier type>
ServiceMetadataPublisherID	Unique identifier of the SMP	xs:string + ServiceMetadataLocatorTypes-1.0.xsd + /[local-name()='schema']/[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element' and @ref='ServiceMetadataPublisherID']	In ManageServiceMetadata service, this establishes the link with the managing SMP of the participant as defined by in the ManageBusinessIdentifier service.
NextPageIdentifier	This argument is not used in this context.	n/a	Must be null

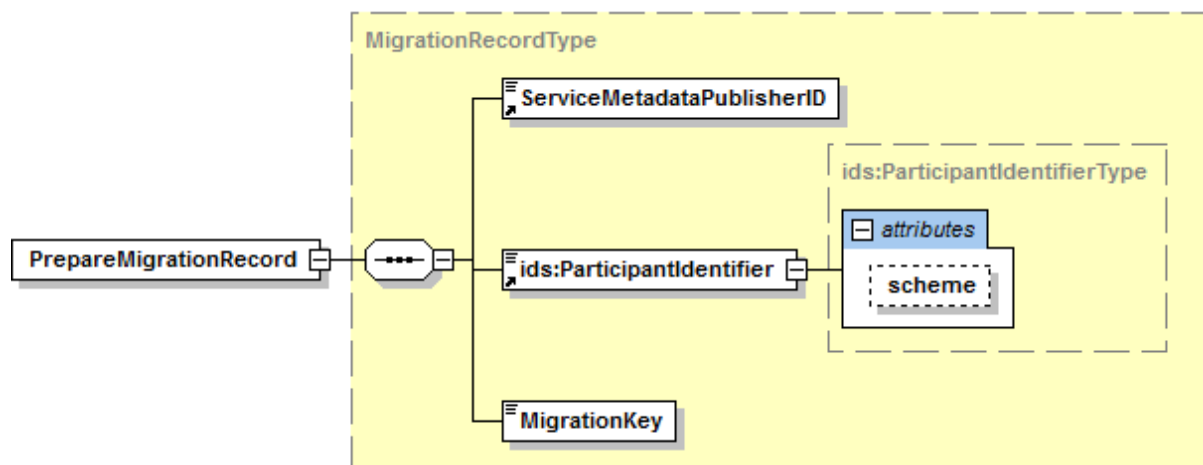
▼ *DeleteList() Output*

None

▼ *PrepareToMigrate() Signature*



▼ PrepareToMigrate() Input



Argument: *ServiceMetadataPublisherID*

Description

Unique identifier of the SMP.

Format/XSD/Xpath

xs:string

+ ServiceMetadataLocatorTypes-1.0.xsd

+ `/[local-name()='schema']/[local-name()='complexType'] and @name='ServiceMetadataPublisherServiceForParticipantType'//[local-name()='sequence']/[local-name()='element'] and @ref='ServiceMetadataPublisherID'`

Constraint

In *ManageServiceMetadata* service, this establishes the link with the managing SMP of the participant as defined by in the *ManageBusinessIdentifier* service.

Argument: *ParticipantIdentifier*

Description

Business unique identifier of the Participant.

Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc...

Format/XSD/Xpath

Constraint

Argument: <i>ParticipantIdentifier</i>	
The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ParticipantIdentifierType'</code>	• Must be unique. • Must be at least 1 character long • Must not exceed 50 characters. • May only contain ASCII characters. • Participant identifier must be trimmed. • If the scheme refers to the (default) PEPPOL participant identifier scheme (iso6523-actorid-upis) and the participant identifier is not a wildcard (*), then the issuing agency specified in the ParticipantIdentifier (as its leading part before ':') must be included in the official list (see List of Valid PEPPOL Issuing Agencies). NOTE The participant identifier is case insensitive.

Argument: <i>ParticipantIdentifier.scheme</i>	
Description The scheme of the participant identifier.	
Format/XSD/Xpath	Constraint
Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. + Examples: iso6523-actorid-upis, busdox-actorid-upis + Identifiers-1.0.xsd + <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute</code>	• May not exceed 25 characters. • Must match the two following pattern: Which defines the following parts: <domain>-<identifier_area>-<identifier_type>

Argument: *MigrationKey*

Description

String which is a unique key controlling the migration of the metadata for a given ParticipantIdentifier from one Service Metadata Publisher to another.

Format/XSD/Xpath

xs:string

+ ServiceMetadataLocatorTypes-1.0.xsd

+ **/[local-name()='schema']/[local-name()='complexType']**
and **@name='MigrationRecordType']/[local-name()='sequence']** **/[local-name()='element']**
and **@name='MigrationKey'**

Constraint

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.

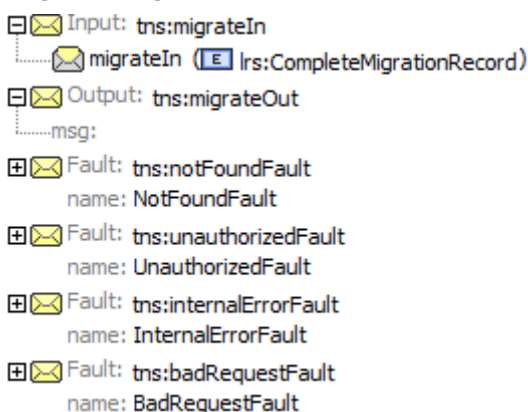
This code:

- must be unique;
- must not be null nor empty;
- 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

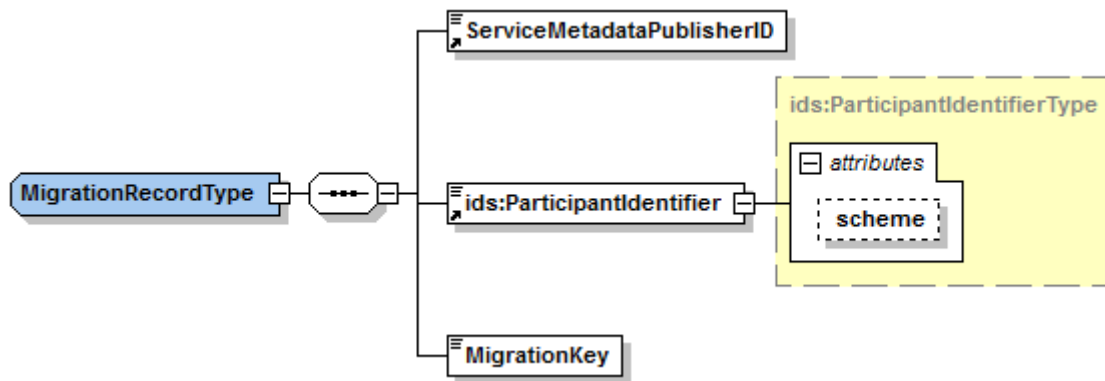
▼ PrepareToMigrate() Output

None

▼ Migrate() Signature



▼ Migrate() Input



Argument: *ServiceMetadataPublisherID*

Description

Unique identifier of the SMP.

Format/XSD/Xpath

xs:string

+ ServiceMetadataLocatorTypes-1.0.xsd

+ `[/[local-name()='schema']/[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element' and @ref='ServiceMetadataPublisherID']`

Constraint

In *ManageServiceMetadata* service, this establishes the link with the managing SMP of the participant as defined by in the *ManageBusinessIdentifier* service.

Argument: *ParticipantIdentifier*

Description

Business unique identifier of the Participant.

Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.

Format/XSD/Xpath

Constraint

Argument: <i>ParticipantIdentifier</i>	
The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd /[local-name()='schema']/[local-name()='complexType'] @name='ParticipantIdentifierType']	• Must be unique. • Must be at least 1 character long. • May not exceed 50 characters. • May only contain ASCII characters. • Participant identifier must be trimmed. • If the scheme refers to the (default) PEPPOL participant identifier scheme (iso6523-actorid-upis) and the participant identifier is not a wildcard (*), then the issuing agency specified in the ParticipantIdentifier (as its leading part before ':') must be included in the official list (see List of Valid PEPPOL Issuing Agencies). NOTE The participant identifier is case insensitive.

Argument: <i>ParticipantIdentifier.scheme</i>	
Description The scheme of the participant identifier.	
Format/XSD/Xpath	Constraint
Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. + Examples: iso6523-actorid-upis, busdox-actorid-upis + Identifiers-1.0.xsd + /[local-name()='schema']/[local-name()='complexType'] @name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute	• May not exceed 25 characters. • Must match the two following pattern: -[a-zA-Z0-9]-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>

Argument: *MigrationKey*

Description

The scheme of the participant identifier.

String which is a unique key controlling the migration of the metadata for a given ParticipantIdentifier from one Service Metadata Publisher to another.

Format/XSD/Xpath

xs:string

+ ServiceMetadataLocatorTypes-1.0.xsd

+ **/[local-name()='schema']/[local-name()='complexType']**
and **@name='MigrationRecordType']/[local-name()='sequence']/[local-name()='element']**

+ and @name='MigrationKey']

Constraint

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.

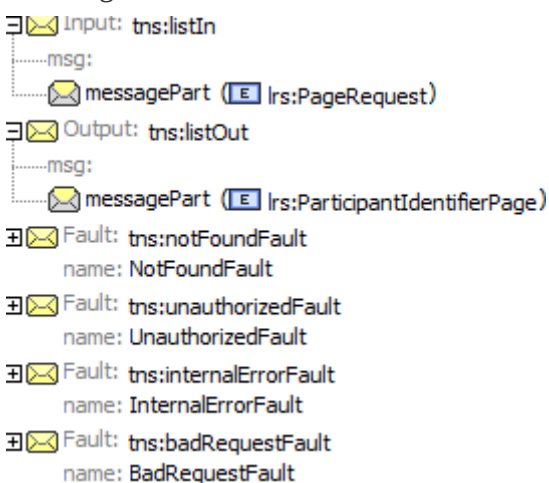
This code:

- must be unique;
- must not be null nor empty;
- 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

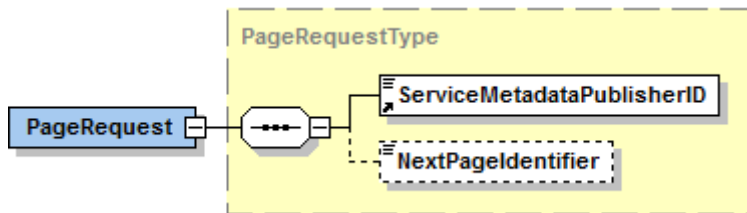
▼ *Migrate() Output*

None

▼ *List() Signature*



▼ *List() Input*



Argument: *ServiceMetadataPublisherID*

Description

Unique identifier of the SMP

Format/XSD/Xpath

xs:string

ServiceMetadataLocatorTypes-1.0.xsd

`/[local-name()='schema']/[local-name()='complexType']` and `@name='ServiceMetadataPublisherServiceForParticipantType']`
`/[local-name()='sequence']/[local-name()='element']` and `@ref='ServiceMetadataPublisherID']`

Constraint

In *ManageServiceMetadata* service, this establishes the link with the managing SMP of the participant as defined by in the *ManageBusinessIdentifier* service.

Argument: *NextPageIdentifier*

Description

Identifier that controls the navigation between pages of a long list.

As Input parameter this identifier represents the page of data to retrieve. If the *NextPageIdentifier* is absent, the first page is returned.

As Output parameter, this value can be used as() Input at the next call to the same operation to retrieve the next page of data (to navigate forward).

This parameter is used only for 'read' operations returning list of values.

Format/XSD/Xpath

xs:string

ServiceMetadataLocatorTypes-1.0.xsd

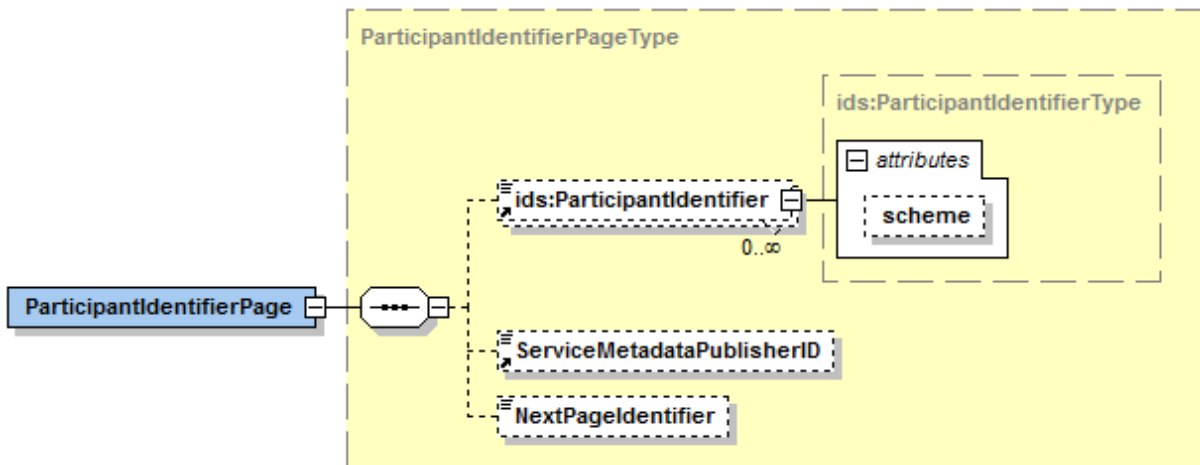
`/[local-name()='schema']/[local-name()='complexType']` and `@name='PageRequestType']`
`/[local-name()='sequence']/[local-name()='element']` and `@name='NextPageIdentifier']`

Constraint

Must be a positive number.

Must be null or empty for create or update operations (since the full list of values has to be provided at once).

▼ *List() Output*



Argument: ParticipantIdentifier (0..n)	
Description Business unique identifier of the Participant. Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.	
Format/XSD/Xpath	Constraint
The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure . Example: 0088:4035811991014 Identifiers-1.0.xsd /[local-name()='schema']/[local-name()='complexType'] @name='ParticipantIdentifierType'] and	• Must be unique. • Must be at least 1 character long. • May not exceed 50 characters. • May only contain ASCII characters. • Participant identifier must be trimmed. • If the scheme refers to the (default) PEPPOL participant identifier scheme (iso6523-actorid-upis) and the participant identifier is not a wildcard (*), then the issuing agency specified in the ParticipantIdentifier (as its leading part before ':') must be included in the official list (see List of Valid PEPPOL Issuing Agencies). NOTE The participant identifier is case insensitive.

Argument: `ParticipantIdentifier.scheme (0..n)`

Description

The scheme of the participant identifier.

+ Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.

Format/XSD/Xpath

Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance.

+ Examples: iso6523-actorid-upis, busdix-actorid-upis

+ Identifiers-1.0.xsd

+ `/[local-name()='schema']/[local-name()='complexType']` and `@name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute`

Constraint

- May not exceed 25 characters.
- Must match the two following pattern:

Which defines the following parts:

`<domain>-<identifier Area>-<identifier type>`

Argument: `ServiceMetadataPublisherID`

Description

Unique identifier of the SMP.

+ Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.

Format/XSD/Xpath

`xs:string`

+ `ServiceMetadataLocatorTypes-1.0.xsd`

+ `/[local-name()='schema']/[local-name()='complexType']` and `@name='ServiceMetadataPublisherServiceForParticipantType']/[local-name()='sequence']/[local-name()='element']` and `@ref='ServiceMetadataPublisherID']`

Constraint

In `ManageServiceMetadata` service, this establishes the link with the managing SMP of the participant as defined by in the `ManageBusinessIdentifier` service.

Argument: *NextPageIdentifier*

Description

Identifier that controls the navigation between pages of a long list.

+ As Input parameter this identifier represents the page of data to retrieve. If the *NextPageIdentifier* is absent, the first page is returned.

+ As Output parameter, this value can be used as() Input at the next call to the same operation to retrieve the next page of data (to navigate forward).

+ This parameter is used only for 'read' operations returning list of values.

Format/XSD/Xpath

xs:string

+ ServiceMetadataLocatorTypes-1.0.xsd

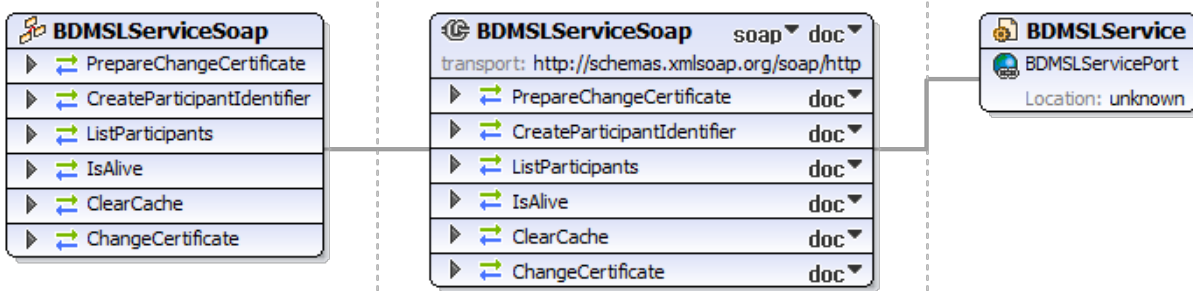
+ `//*[@local-name()='schema']/[local-name()='complexType']` and
`@name='PageRequestType']/[local-name()='sequence']/[local-name()='element']`
and `@name='NextPageIdentifier']`

Constraint

Must be a positive number.

Must be null or empty for create or update operations (since the full list of values has to be provided at once).

WSDL model for BDMSLSERVICE

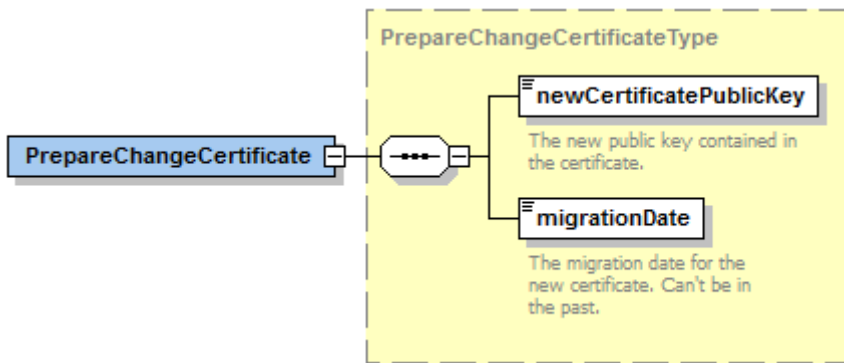


Operations Signatures

▼ PrepareChangeCertificate() Signature

- Input: `tns:prepareChangeCertificateIn`
 - msg:
 - messagePart (E) `tns:prepareChangeCertificateIn`
- Output: `tns:prepareChangeCertificateOut`
 - msg:
- Fault: `tns:notFoundFault`
 - name: `NotFoundFault`
- Fault: `tns:unauthorizedFault`
 - name: `UnauthorizedFault`
- Fault: `tns:internalErrorFault`
 - name: `InternalErrorFault`
- Fault: `tns:badRequestFault`
 - name: `BadRequestFault`

▼ PrepareChangeCertificate() Input



Argument: `newCertificatePublicKey`

Description

The new public key contained in the certificate

Format/XSD/Xpath

base64Binary
 BDMSLSERVICE-1.0.xsd
`/[local-name()='schema']/[local-name()='complexType'] and
 @name='PrepareChangeCertificateType']/[local-name()='sequence']/[local-name()='element']
 and @name='newCertificatePublicKey']`

Constraint

Must be valid and belong to the list of authorized root certificate aliases.

Argument: `migrationDate`

Description

The migration date for the new certificate.

xs:date
 BDMSLSERVICE-1.0.xsd
`/[local-name()='schema']/[local-name()='complexType'] and
 @name='PrepareChangeCertificateType']/[local-name()='sequence']/[local-name()='element']
 and @name='migrationDate']`

- May not be in the past.
- Must be in the validity period of the related new certificate (i.e. within `NotBeforeCertificateDate` and `NotAfterCertificateDate` attribute of the certificate).
 If `migrationDate` is empty, then the "Valid From" date is extracted from the certificate and is used as the `migrationDate`.

▼ *PrepareChangeCertificate() Output*

None

▼ *CreateParticipantIdentifier() Signature*

Argument: <code>ParticipantIdentifier</code>	
Business unique identifier of the Participant. Represents a business level endpoint key that uniquely identifies an end-user entity in the network. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses, etc.	The format must comply with ISO 15459 constraints as defined in Policy for the use of Identifiers in PEPPOL Transport Infrastructure. Example: 0088:4035811991014 Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ParticipantIdentifierType']</code>

Argument: <code>ParticipantIdentifier.scheme</code>	
Description The scheme of the participant identifier.	
Format/XSD/Xpath	Constraint
Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure may choose to define identifier schemes that match the type of documents, participants or profiles that are relevant to support in that instance. Examples: iso6523-actorid-upis, busdox-actorid-upis Identifiers-1.0.xsd <code>/[local-name()='schema']/[local-name()='complexType']</code> and <code>@name='ProcessIdentifierType']/xs:simpleContent/xs:extension/xs:attribute</code>	- May not exceed 25 characters. - Must match the two following pattern: Which defines the following parts: <code><domain>-<identifier_area>-<identifier_type></code>

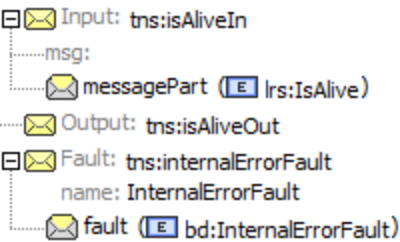
Argument: <code>serviceName</code>	
Description The name of the service for the NAPTR record.	
Format/XSD/Xpath	Constraint

Argument: <i>serviceName</i>	
xs:string BDMSLService-1.0.xsd / [local-name()='schema'] /[local-name()='complexType' and @name='SMPAdvancedServiceForParticipantT ype']/ [local-name()='sequence'] /[local- name()='element'and @name='serviceName']	No constraint is enforced by the DomiSML

▼ *CreateParticipantIdentifier() Output*

None

▼ *IsAlive() Signature*



▼ *IsAlive() Input*

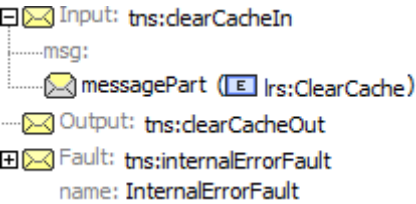


(empty)

▼ *IsAlive() Output*

None

▼ *ClearCache() Signature*



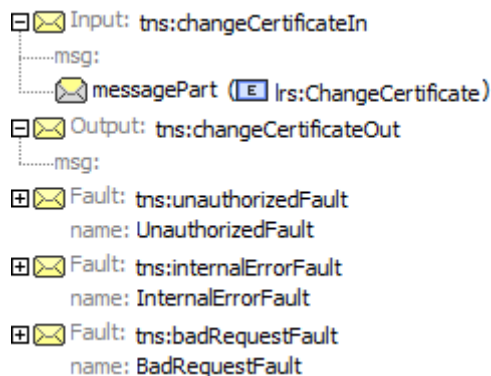
▼ *ClearCache() Input*

none

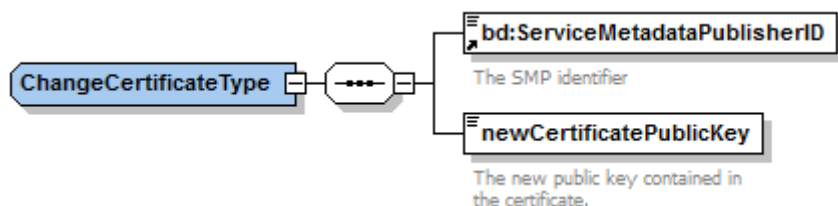
▼ *ClearCache() Output*

none

▼ *ChangeCertificate() Signature*



▼ ChangeCertificate() Input



Argument: *ServiceMetadataPublisherID*

Description

Unique identifier of the SMP.

Format/XSD/Xpath

xs:string

ServiceMetadataLocatorTypes-1.0.xsd

/[local-name()='schema']/[local-name()='complexType'] and @name='ServiceMetadataPublisherServiceForParticipantType']//[local-name()='sequence']/[local-name()='element'] + and @ref='ServiceMetadataPublisherID']

Constraint

In *ManageServiceMetadata* service, this establishes the link with the managing SMP of the participant as defined by in the *ManageBusinessIdentifier* service.

Argument: *newCertificatePublicKey*

Description

The new public key contained in the certificate.

Format/XSD/Xpath

base64Binary

BDMSLSERVICE-1.0.xsd

/[local-name()='schema']/[local-name()='complexType'] and @name='PrepareChangeCertificateType']//[local-name()='sequence']/[local-name()='element'] and @name='newCertificatePublicKey']

Constraint

Must be valid and belong to the list of authorized root certificate aliases.

▼ *ChangeCertificate()* Output

None

Faults

Faults generic specifications

All operations above may return all or some of the possible following faults:

- `notFoundFault`: the target element(s) (`MetadataPublisher`, `Participant` ...) on which the operation must be performed is not present into the configuration.
- `unauthorizedFault`: the user does not have the permission to execute that operation
- `badRequestFault`: the structure of the request is not well-formed
- `internalFault`: any other error occurred during the processing of the request

Error Codes

Whenever a fault occurs, more details on the source of the error will be provided in the SOAP fault with the applicable error code as listed in the table below:

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

Faults specific usages

The tables that follow show the applicability of each error per operation specified in this specification:

	ManageServiceMetadata			
Error/Operation	notFoundFault	unauthorizedFault	internalErrorFault	badRequestFault
Create		X	X	X
Read	X	X	X	X
Update	X	X	X	X
Delete	X	X	X	X

	ManageBusinessIdentifier			
Error/Operation	notFoundFault	unauthorizedFault	internalErrorFault	badRequestFault
Create	X	X	X	X
CreateList	X	X	X	X
Delete	X	X	X	X
DeleteList	X	X	X	X
PrepareToMigrate	X	X	X	X
Migrate	X	X	X	X
List	X	X	X	X

	BDMSLService			
Error/Operation	notFoundFault	unauthorizedFault	internalErrorFault	badRequestFault
PrepareChangeCertificate	X	X	X	X
CreateParticipantIdentifier	X	X	X	X
IsAlive		X	X	X

	BDMSLAdminService			
Error/Operation	notFoundFault	unauthorizedFault	internalErrorFault	badRequestFault
SetProperty	X	X	X	X
GetProperty	X	X	X	X
DeleteProperty	X	X	X	X
CreateSuDomain	X	X	X	X
UpdateSubDomain	X	X	X	X
GetSubDomain	X	X	X	X
DeleteSubDomain	X	X	X	X
AddSubDomainCertificate	X	X	X	X
UpdateSubDomainCertificate	X	X	X	X

	BDMSLAdminService			
ListSubDomainCertificates		X	X	X
AddDNSRecord	X	X	X	X
DeleteDNSRecord	X	X	X	X
ClearCache		X	X	X
ChangeCertificate	X	X	X	X
AddTruststoreCertificate		X	X	X
GetTruststoreCertificate	X	X	X	X
DeleteTruststoreCertificate	X	X	X	X
ListTruststoreCertificateAliases		X	X	X
ManageServiceMetadaPublisher	X	X	X	X

Sample SOAP errors

The following are SOAP faults samples as they are returned to the requester in case of error encountered by the DomiSML.

▼ Sample `NotFoundFault`

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>[ERR-100] The SMP 'testSMPPPrepareToMigrate520'
doesn't exist.
[14ojYP80p4vWs78XmcVtLBVFsbPat1mOE9h8HChQW0048Xb00ZXu!-
2114654990!1469472374542]</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
'testSMPPPrepareToMigrate520' doesn't exist.</FaultMessage>
        </NotFoundFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Sample "BadRequestFault":

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  <soap:Body>
```

```

    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>[ERR-106]Participant Identifier value "9999" is
illegal .
[NtsjYc25LzhHUNmQ4_Z6Bv8En5sB2e9WjFhMZrBTUcLHT5QlsR99!-
2114654990!1469472427449]</faultstring>
      <detail>
        <BadRequestFault
          xmlns:ns2="http://busdox.org/transport/identifiers/1.0/"
          xmlns="http://busdox.org/serviceMetadata/locator/1.0/">
          <FaultMessage>[ERR-106] Participant Identifier Value
contains the illegal issuing agency '0185'</FaultMessage>
        </BadRequestFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>

```

3.3. Annexes

3.3.1. Interface Message standards

The WSDL and XSD documents can all be downloaded from the eDelivery GIT repository at this location:

<https://ec.europa.eu/digital-building-blocks/code/projects/EDELIVERY/repos/bdmsl/browse/bdmsl-api/src/main/resources>

WSDL's

▼ *ManageBusinessIdentifierService*

```

<!--
(C) Copyright 2018 - European Commission | CEF eDelivery

Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://joinup.ec.europa.eu/sites/default/files/custom-
page/attachment/eupl_v1.2_en.pdf

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->

```

```

<wsdl:definitions
xmlns:tns="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/"
xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
name="ManageBusinessIdentifierService"
targetNamespace="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/Schema/">
      <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
schemaLocation="ServiceMetadataLocatorTypes-1.0.xsd"/>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="createIn">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:part name="messagePart" element="lrs>CreateParticipantIdentifier"/>
  </wsdl:message>
  <wsdl:message name="createOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  </wsdl:message>
  <wsdl:message name="deleteIn">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:part name="messagePart" element="lrs>DeleteParticipantIdentifier"/>
  </wsdl:message>
  <wsdl:message name="deleteOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  </wsdl:message>
  <wsdl:message name="listIn">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:part name="messagePart" element="lrs:PageRequest"/>
  </wsdl:message>
  <wsdl:message name="listOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:part name="messagePart" element="lrs:ParticipantIdentifierPage"/>
  </wsdl:message>
  <wsdl:message name="prepareMigrateIn">
    <wsdl:part name="prepareMigrateIn" element="lrs:PrepareMigrationRecord"/>
  </wsdl:message>
  <wsdl:message name="prepareMigrateOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  </wsdl:message>
  <wsdl:message name="migrateIn">
    <wsdl:part name="migrateIn" element="lrs:CompleteMigrationRecord"/>
  </wsdl:message>
  <wsdl:message name="migrateOut">

```

```

    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
</wsdl:message>
<wsdl:message name="createListIn">
    <wsdl:part name="createListIn" element="lrs:CreateList" />
</wsdl:message>
<wsdl:message name="createListOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
</wsdl:message>
<wsdl:message name="deleteListIn">
    <wsdl:part name="deleteListIn" element="lrs:DeleteList" />
</wsdl:message>
<wsdl:message name="deleteListOut">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
</wsdl:message>
<wsdl:message name="badRequestFault">
    <wsdl:part name="fault" element="lrs:BadRequestFault" />
</wsdl:message>
<wsdl:message name="internalErrorFault">
    <wsdl:part name="fault" element="lrs:InternalErrorFault" />
</wsdl:message>
<wsdl:message name="notFoundFault">
    <wsdl:part name="fault" element="lrs:NotFoundFault" />
</wsdl:message>
<wsdl:message name="unauthorizedFault">
    <wsdl:part name="fault" element="lrs:UnauthorizedFault" />
</wsdl:message>
<wsdl:portType name="ManageBusinessIdentifierServiceSoap">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
    <wsdl:operation name="Create">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
        <wsdl:input message="tns:createIn" />
        <wsdl:output message="tns:createOut" />
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault" />
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault" />
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault" />
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault" />
    </wsdl:operation>
    <wsdl:operation name="Delete">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
        <wsdl:input message="tns:deleteIn" />
        <wsdl:output message="tns:deleteOut" />
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault" />
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault" />
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault" />
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault" />
    </wsdl:operation>
    <wsdl:operation name="List">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
        <wsdl:input message="tns:listIn" />
        <wsdl:output message="tns:listOut" />
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault" />

```

```

        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    </wsdl:operation>
    <wsdl:operation name="PrepareToMigrate">
        <wsdl:input message="tns:prepareMigrateIn"/>
        <wsdl:output message="tns:prepareMigrateOut"/>
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    </wsdl:operation>
    <wsdl:operation name="Migrate">
        <wsdl:input message="tns:migrateIn"/>
        <wsdl:output message="tns:migrateOut"/>
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    </wsdl:operation>
    <wsdl:operation name="CreateList">
        <wsdl:input message="tns:createListIn"/>
        <wsdl:output message="tns:createListOut"/>
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    </wsdl:operation>
    <wsdl:operation name="DeleteList">
        <wsdl:input message="tns:deleteListIn"/>
        <wsdl:output message="tns:deleteListOut"/>
        <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
        <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
        <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
        <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ManageBusinessIdentifierServiceSoap"
type="tns:ManageBusinessIdentifierServiceSoap">
    <soap11:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Create">
        <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:createIn" style="document"/>
        <wsdl:input>
            <soap11:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap11:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NotFoundFault">

```



```

        <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InternalErrorFault">
        <soap:fault name="InternalErrorFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadRequestFault">
        <soap:fault name="BadRequestFault" use="literal"/>
    </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="CreateList">
    <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:createListIn" style="document"/>
    <wsdl:input>
        <soap11:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap11:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundFault">
        <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InternalErrorFault">
        <soap:fault name="InternalErrorFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadRequestFault">
        <soap:fault name="BadRequestFault" use="literal"/>
    </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="Delete">
    <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:deleteIn" style="document"/>
    <wsdl:input>
        <soap11:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap11:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NotFoundFault">
        <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>

```

```

        <wsdl:fault name="InternalServerErrorFault">
            <soap:fault name="InternalServerErrorFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="BadRequestFault">
            <soap:fault name="BadRequestFault" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="DeleteList">
        <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:deleteListIn" style="document"/>
        <wsdl:input>
            <soap11:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap11:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NotFoundFault">
            <soap:fault name="NotFoundFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnauthorizedFault">
            <soap:fault name="UnauthorizedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InternalServerErrorFault">
            <soap:fault name="InternalServerErrorFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="BadRequestFault">
            <soap:fault name="BadRequestFault" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="List">
        <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:listIn" style="document"/>
        <wsdl:input>
            <soap11:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap11:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NotFoundFault">
            <soap:fault name="NotFoundFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnauthorizedFault">
            <soap:fault name="UnauthorizedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InternalServerErrorFault">
            <soap:fault name="InternalServerErrorFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="BadRequestFault">
            <soap:fault name="BadRequestFault" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>

```

```

        </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="PrepareToMigrate">
        <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:prepareMigrateIn" style="document"/>
        <wsdl:input>
            <soap11:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap11:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NotFoundFault">
            <soap:fault name="NotFoundFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnauthorizedFault">
            <soap:fault name="UnauthorizedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InternalErrorFault">
            <soap:fault name="InternalErrorFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="BadRequestFault">
            <soap:fault name="BadRequestFault" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Migrate">
        <soap11:operation
soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
:migrateIn" style="document"/>
        <wsdl:input>
            <soap11:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap11:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="NotFoundFault">
            <soap:fault name="NotFoundFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnauthorizedFault">
            <soap:fault name="UnauthorizedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InternalErrorFault">
            <soap:fault name="InternalErrorFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="BadRequestFault">
            <soap:fault name="BadRequestFault" use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="ManageBusinessIdentifierService">
    <wsdl:port name="ManageBusinessIdentifierServicePort"

```

```

binding="tns:ManageBusinessIdentifierServiceSoap">
  <soap:address location="unknown"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

▼ ServiceGroupReferenceList.xsd reformed

```

?xml version="1.0" encoding="utf-8"?>
<!--
(C) Copyright 2018 - European Commission | CEF eDelivery

Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License.
You may obtain a copy of the License at

https://joinup.ec.europa.eu/sites/default/files/custom-
page/attachment/eupl_v1.2_en.pdf

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->

<xs:schema xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://busdox.org/serviceMetadata/publishing/1.0/"
elementFormDefault="qualified" id="ServiceGroupReferenceList">
  <xs:include schemaLocation="ServiceMetadataPublishingTypes-1.0.xsd"/>
  <xs:element name="ServiceGroupReferenceList"
type="ServiceGroupReferenceListType"/>
  <xs:complexType name="ServiceGroupReferenceListType">
    <xs:sequence>
      <xs:element name="ServiceGroupReference"
type="ServiceGroupReferenceType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ServiceGroupReferenceType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="href" type="xs:anyURI"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="CompleteServiceGroup" type="CompleteServiceGroupType"/>
  <xs:complexType name="CompleteServiceGroupType">
    <xs:sequence>

```

```

        <xs:element ref="ServiceGroup"/>
        <xs:element ref="ServiceMetadata" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>

```

▼ ServiceMetadataLocatorTypes-1.0.xsd

```

<?xml version="1.0" encoding="utf-8"?>
<!--
(C) Copyright 2018 - European Commission | CEF eDelivery
Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License. You may obtain a
copy of the License at
https://joinup.ec.europa.eu/sites/default/files/custom-
page/attachment/eupl_v1.2_en.pdf
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
License for the specific language governing permissions and
limitations under the License.
-->
<xs:schema xmlns="http://busdox.org/serviceMetadata/locator/1.0/"
xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://busdox.org/serviceMetadata/locator/1.0/"
elementFormDefault="qualified" id="ServiceMetadataPublisherService">
  <xs:import namespace="http://busdox.org/transport/identifiers/1.0/"
schemaLocation="Identifiers-1.0.xsd"/>
  <xs:element name="ServiceMetadataPublisherID" type="xs:string"/>
  <xs:element name="CreateServiceMetadataPublisherService"
type="ServiceMetadataPublisherServiceType"/> <xs:element
name="ReadServiceMetadataPublisherService"
type="ServiceMetadataPublisherServiceType"/> <xs:element
name="UpdateServiceMetadataPublisherService"
type="ServiceMetadataPublisherServiceType"/> <xs:element
name="ServiceMetadataPublisherService" type="ServiceMetadataPublisherServiceType"/>
  <xs:complexType name="ServiceMetadataPublisherServiceType">
    <xs:sequence>
      <xs:element name="PublisherEndpoint" type="PublisherEndpointType"/> <xs:element
ref="ServiceMetadataPublisherID"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PublisherEndpointType">
    <xs:sequence>
      <xs:element name="LogicalAddress" type="xs:anyURI"/> <xs:element
name="PhysicalAddress" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ServiceMetadataPublisherServiceForParticipantType">

```

```

<xs:sequence>
  <xs:element ref="ServiceMetadataPublisherID"/> <xs:element
  ref="ids:ParticipantIdentifier"/>
</xs:sequence>
</xs:complexType>
<xs:element name="CreateParticipantIdentifier"
  type="ServiceMetadataPublisherServiceForParticipantType"/> <xs:element
  name="DeleteParticipantIdentifier"
  type="ServiceMetadataPublisherServiceForParticipantType"/> <xs:element
  name="ParticipantIdentifierPage" type="ParticipantIdentifierPageType"/>
<xs:element name="CreateList" type="ParticipantIdentifierPageType"/>
<xs:element name="DeleteList" type="ParticipantIdentifierPageType"/>
<xs:complexType name="ParticipantIdentifierPageType">
  <xs:sequence>
    <xs:element ref="ids:ParticipantIdentifier" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="ServiceMetadataPublisherID" minOccurs="0"/>
    <xs:element name="NextPageIdentifier" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="PageRequest" type="PageRequestType"/> <xs:complexType
  name="PageRequestType">
  <xs:sequence>
    <xs:element ref="ServiceMetadataPublisherID"/>
    <xs:element name="NextPageIdentifier" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="PrepareMigrationRecord" type="MigrationRecordType"/> <xs:element
  name="CompleteMigrationRecord" type="MigrationRecordType"/> <xs:complexType
  name="MigrationRecordType">
  <xs:sequence>
    <xs:element ref="ServiceMetadataPublisherID"/> <xs:element
    ref="ids:ParticipantIdentifier"/> <xs:element name="MigrationKey" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="BadRequestFault" type="FaultType"/> <xs:element
  name="InternalServerError" type="FaultType"/> <xs:element name="NotFoundFault"
  type="FaultType"/> <xs:element name="UnauthorizedFault" type="FaultType"/>
<xs:complexType name="FaultType">
  <xs:sequence>
    <xs:element name="FaultMessage" type="xs:string" minOccurs="0"/>
  </xs:sequence> </xs:complexType>
</xs:schema>

```

▼ ServiceMetadataPublishingTypes-1.0.xsd

```

<?xml version="1.0" encoding="utf-8"?>
<!--
(C) Copyright 2018 - European Commission | CEF eDelivery
Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License. You may obtain a

```

copy of the License at

https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

-->

```
<xs:schema xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
targetNamespace="http://busdox.org/serviceMetadata/publishing/1.0/"
elementFormDefault="qualified" id="ServiceMetadataPublishing">
<xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="xmldsig-
core-schema.xsd"/> <xs:import
namespace="http://busdox.org/transport/identifiers/1.0/"
schemaLocation="Identifiers-1.0.xsd"/> <xs:import
namespace="http://www.w3.org/2005/08/addressing" schemaLocation="ws-addr.xsd"/>
<xs:element name="ServiceGroup" type="ServiceGroupType"/>
<xs:element name="ServiceMetadata" type="ServiceMetadataType"/>
<xs:element name="SignedServiceMetadata" type="SignedServiceMetadataType"/>
<xs:complexType name="SignedServiceMetadataType">
<xs:annotation>
<xs:documentation>The SignedServiceMetadata structure is a ServiceMetadata structure
that has been signed by the ServiceMetadataPublisher, according to governance
policies.</xs:documentation> </xs:annotation>
<xs:sequence>
<xs:element ref="ServiceMetadata">
<xs:annotation>
<xs:documentation>The ServiceMetadata element covered by the
Signature.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element ref="ds:Signature"> <xs:annotation>
<xs:documentation>Represents an enveloped XML() Signature over the
SignedServiceMetadata element.</xs:documentation>
</xs:annotation> </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceMetadataType">
<xs:annotation> <xs:documentation>
This data structure represents Metadata about a specific electronic service.
The role of the ServiceMetadata structure is to associate a participant identifier
with the ability to receive a specific document type over a specific transport. It
also describes which business processes a document can participate in, and various
operational data such as service activation and expiration times.
The ServiceMetadata resource contains all the metadata about a service that a sender
Access Point needs to know in order to send a message to that service.
</xs:documentation> </xs:annotation>
```



```

<xs:sequence> <xs:choice>
  <xs:element name="ServiceInformation" type="ServiceInformationType"> <xs:annotation>
    <xs:documentation>Contains service information for an actual service registration,
    rather than a redirect to another SMP</xs:documentation>
  </xs:annotation> </xs:element>
  <xs:element name="Redirect" type="RedirectType"> <xs:annotation>
    <xs:documentation>
      For recipients that want to associate more than one SMP with their participant
      identifier,
      they may redirect senders to an alternative SMP for specific document types. To
      achieve this, the ServiceMetadata element defines the optional element Redirect.
      This element holds the URL of the alternative SMP, as well as the Subject Unique
      Identifier of the destination SMPs certificate used to sign its resources.
      In the case where a client encounters such a redirection element, the client MUST
      follow the first redirect reference to the alternative SMP. If the
      SignedServiceMetadata resource at the alternative SMP also contains a redirection
      element, the client SHOULD NOT follow that redirect. It is the responsibility of the
      client to enforce this constraint.
    </xs:documentation> </xs:annotation>
  </xs:element>
</xs:choice> </xs:sequence> </xs:complexType>
<xs:complexType name="ServiceInformationType"> <xs:sequence>
  <xs:element ref="ids:ParticipantIdentifier"> <xs:annotation>
    <xs:documentation>The participant identifier. Comprises the identifier, and an
    identifier scheme. This identifier MUST have the same value of the {id} part of the
    URI of the enclosing ServiceMetadata resource.</xs:documentation>
  </xs:annotation> </xs:element>
  <xs:element ref="ids:DocumentIdentifier"> <xs:annotation>
    <xs:documentation>Represents the type of document that the recipient is able to
    handle.</xs:documentation> </xs:annotation>
  </xs:element>
  <xs:element name="ProcessList" type="ProcessListType">
    <xs:annotation>
      <xs:documentation>Represents the processes that a specific document type can
      participate in, and endpoint address
      and binding information. Each process element describes a specific business process
      that accepts this type of document as() Input and holds a list of endpoint addresses
      (in the case that the service supports multiple transports) of services that
      implement the business process, plus information about the transport used for each
      endpoint.</xs:documentation>
    </xs:annotation> </xs:element>
  <xs:element name="Extension" type="ExtensionType" minOccurs="0"> <xs:annotation>
    <xs:documentation>The extension element may contain any XML element. Clients MAY
    ignore this element.</xs:documentation>
  </xs:annotation> </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ProcessListType">
  <xs:annotation>
    <xs:documentation>List of processes</xs:documentation>
  </xs:annotation> <xs:sequence>

```



```

<xs:element name="Process" type="ProcessType" maxOccurs="unbounded"/> </xs:sequence>
</xs:complexType>
<xs:complexType name="ProcessType">
<xs:sequence>
<xs:element ref="ids:ProcessIdentifier">
<xs:annotation>
<xs:documentation>The identifier of the process.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="ServiceEndpointList" type="ServiceEndpointList"> <xs:annotation>
<xs:documentation>List of one or more endpoints that support this
process.</xs:documentation> </xs:annotation>
</xs:element>
<xs:element name="Extension" type="ExtensionType" minOccurs="0">
<xs:annotation>
<xs:documentation>
    The extension element may contain any XML element. Clients MAY ignore this
    element.
</xs:documentation>
</xs:annotation>
</xs:element> </xs:sequence> </xs:complexType>
<xs:complexType name="ServiceEndpointList"> <xs:annotation>
<xs:documentation>Contains a list of all endpoint</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:element name="Endpoint" type="EndpointType" maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>Endpoint represents the technical endpoint and address type of the
recipient, as an
URL.</xs:documentation> </xs:annotation>
</xs:element> </xs:sequence>
</xs:complexType>
<xs:complexType name="EndpointType">
<xs:sequence>
<xs:element ref="wsa:EndpointReference">
<xs:annotation>
<xs:documentation>The address of an endpoint, as an WS-Addressing Endpoint
Reference</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="RequireBusinessLevelSignature" type="xs:boolean"> <xs:annotation>
<xs:documentation>Set to &quot;true&quot; if the recipient requires business-level()
Signatures for the message, meaning a() Signature applied to the business message
before the message is put on the transport. This is independent of the transport-
level() Signatures that a specific transport profile, such as the START profile,
might mandate. This flag does not indicate which type of business-level() Signature
might be required. Setting or consuming business-level() Signatures would typically
be the responsibility of the final senders and receivers of messages, rather than a
set of APs.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="MinimumAuthenticationLevel" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>Indicates the minimum authentication level that recipient

```

```

requires. The specific semantics of this field is defined in a specific instance of
the BUSDOX infrastructure. It could for example reflect the value of the
&quot;urn:eu:busdox:attribute:assurance-level&quot; SAML attribute defined in the
START specification.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="ServiceActivationDate" type="xs:dateTime" minOccurs="0">
<xs:annotation>
<xs:documentation>Activation date of the service. Senders should ignore services
that are not yet activated.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="ServiceExpirationDate" type="xs:dateTime" minOccurs="0">
<xs:annotation>
<xs:documentation>Expiration date of the service. Senders should ignore services
that are expired.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="Certificate" type="xs:string"> <xs:annotation>
<xs:documentation>Holds the complete signing certificate of the recipient AP, as a
PEM base 64 encoded X509 DER formatted value.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="ServiceDescription" type="xs:string"> <xs:annotation>
<xs:documentation>A human readable description of the service</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="TechnicalContactUrl" type="xs:anyURI">
<xs:annotation>
<xs:documentation>Represents a link to human readable contact information. This
might also be an email
address.</xs:documentation> </xs:annotation>
</xs:element>
<xs:element name="TechnicalInformationUrl" type="xs:anyURI" minOccurs="0">
<xs:annotation>
<xs:documentation>A URL to human readable documentation of the service format. This
could for example be a web
site containing links to XML Schemas, WSDLs, Schematrons and other relevant
resources.</xs:documentation> </xs:annotation>
</xs:element>
<xs:element name="Extension" type="ExtensionType" minOccurs="0">
<xs:annotation>
<xs:documentation>The extension element may contain any XML element. Clients MAY
ignore this
element.</xs:documentation> </xs:annotation>
</xs:element> </xs:sequence>
<xs:attribute name="transportProfile" type="xs:string"> <xs:annotation>
<xs:documentation>Indicates the type of BUSDOX transport that is being used between
access points, e.g. the BUSDOX START profile. This specification defines the
following identifier URI which denotes the BUSDOX START transport: &quot;busdox-
transport- start&quot;</xs:documentation>
</xs:annotation>
</xs:attribute>
</xs:complexType>
<xs:complexType name="ServiceGroupType">

```

```

<xs:annotation>
<xs:documentation>The ServiceGroup structure represents a set of services
associated with a specific participant identifier that is handled by a specific
Service Metadata Publisher. The ServiceGroup structure holds a list of references to
SignedServiceMetadata resources in the ServiceList structure.</xs:documentation>
</xs:annotation> <xs:sequence>
<xs:element ref="ids:ParticipantIdentifier"> <xs:annotation>
<xs:documentation>Represents the business level endpoint key and key type, e.g. a
DUNS or GLN number that is associated with a group of services. </xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="ServiceMetadataReferenceCollection"
type="ServiceMetadataReferenceCollectionType"> <xs:annotation>
<xs:documentation>The ServiceMetadataReferenceCollection structure holds a list of
references to SignedServiceMetadata structures. From this list, a sender can follow
the references to get each SignedServiceMetadata structure.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="Extension" type="ExtensionType" minOccurs="0"> <xs:annotation>
<xs:documentation>The extension element may contain any XML element. Clients MAY
ignore this element.</xs:documentation>
</xs:annotation> </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceMetadataReferenceCollectionType">
<xs:annotation>
<xs:documentation>Contains the URL to a specific SignedServiceMetadata instance.
Note
that references MUST refer to SignedServiceMetadata records that are signed by the
certificate of the SMP. It must not point to SignedServiceMetadata resources
published by external SMPs.</xs:documentation>
</xs:annotation> <xs:sequence>
<xs:element name="ServiceMetadataReference" type="ServiceMetadataReferenceType"
minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceMetadataReferenceType">
<xs:attribute name="href" type="xs:anyURI"> <xs:annotation>
<xs:documentation>Contains the URL to a specific SignedServiceMetadata
instance.</xs:documentation> </xs:annotation>
</xs:attribute>
</xs:complexType>
<xs:complexType name="RedirectType">
<xs:sequence>
<xs:element name="CertificateUID" type="xs:string">
<xs:annotation>
<xs:documentation>Holds the Subject Unique Identifier of the certificate of the
destination SMP. A client SHOULD
validate that the Subject Unique Identifier of the certificate used to sign the
resource at the destination SMP matches the Subject Unique Identifier published in
the redirecting SMP.</xs:documentation>
</xs:annotation> </xs:element>
<xs:element name="Extension" type="ExtensionType" minOccurs="0"> <xs:annotation>

```

```

<xs:documentation>The extension element may contain any XML element. Clients MAY
ignore this element.</xs:documentation>
</xs:annotation> </xs:element>
</xs:sequence>
<xs:attribute name="href" type="xs:anyURI">
<xs:annotation>
<xs:documentation>The destination URL of the redirect.</xs:documentation>
</xs:annotation> </xs:attribute>
</xs:complexType>
<xs:complexType name="ExtensionType">
<xs:annotation>
<xs:documentation>
Child elements of the &lt;smp:Extension&gt; element are known as &quot;custom
extension elements&quot;. Extension points may be used for optional extensions
of service metadata. This implies:
* Extension elements added to a specific Service Metadata resource MUST be ignorable
by any client of the transport infrastructure. The ability to parse and adjust
client
behavior based on an extension element MUST NOT be a prerequisite for a client to
locate a service, or to make a successful request at the referenced service.
* A client MAY ignore any extension element added to specific service metadata
resource instances.
</xs:documentation>
</xs:annotation>
<xs:sequence>
<!-- TODO processContents="skip" will be added after 1.1.0 -->
<xs:any/>
</xs:sequence>
</xs:complexType>
</xs:schema>

```

▼ *BDMSLSERVICE-1.0.xsds*

```

<?xml version="1.0" encoding="utf-8"?>
<!--
(C) Copyright 2018 - European Commission | CEF eDelivery
Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License. You may obtain a
copy of the License at
https://joinup.ec.europa.eu/sites/default/files/custom-
page/attachment/eupl_v1.2_en.pdf
Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the specific
language governing permissions and limitations under the License.
-->
<xs:schema xmlns="ec:services:wsl:BDMSL:data:1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bd="http://busdox.org/serviceMetadata/locator/1.0/"
xmlns:id="http://busdox.org/transport/identifiers/1.0/"

```

```

targetNamespace="ec:services:wsdl:BDMSL:data:1.0" elementFormDefault="qualified"
id="BDMSLTypes">
<xs:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
schemaLocation="ServiceMetadataLocatorTypes-1.0.xsd"/>
<xs:import namespace="http://busdox.org/transport/identifiers/1.0/"
schemaLocation="Identifiers-1.0.xsd"/>
<xs:element name="PrepareChangeCertificate" type="PrepareChangeCertificateType"/>
<xs:element name="SMPAdvancedServiceForParticipantService"
type="SMPAdvancedServiceForParticipantType"/>
<xs:element name="IsAlive" type="IsAliveType"/>
<xs:element name="ExistsParticipant" type="ParticipantsType"/> <xs:element
name="ExistsParticipantResponse"
type="ExistsParticipantResponseType"/>
<xs:complexType name="PrepareChangeCertificateType">
<xs:sequence>
<xs:element name="newCertificatePublicKey" type="xs:string">
<xs:annotation>
<xs:documentation>The new public key contained in the
certificate.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="migrationDate" type="xs:date" minOccurs="0">
<xs:annotation>
<xs:documentation>The migration date for the new
certificate. Can't be in the past.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="SMPAdvancedServiceForParticipantType"> <xs:sequence>
<xs:element name="CreateParticipantIdentifier"
type="bd:ServiceMetadataPublisherServiceForParticipantType"/>
<xs:element name="serviceName" type="xs:string"> <xs:annotation>
<xs:documentation>The name of the service for the NAPTR record.</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ListParticipantsType">
<xs:sequence>
<xs:element name="participant" type="ParticipantsType"
minOccurs="0" maxOccurs="unbounded"/> </xs:sequence>
</xs:complexType>
<xs:complexType name="ParticipantsType">
<xs:sequence>
<xs:element ref="id:ParticipantIdentifier">
<xs:annotation>
<xs:documentation>The participant
identifier</xs:documentation>
</xs:annotation>

```

```

</xs:element>
<xs:element ref="bd:ServiceMetadataPublisherID">
<xs:annotation>
<xs:documentation>The SMP identifier</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="IsAliveType"/>
<xs:complexType name="ExistsParticipantResponseType"> <xs:sequence>
<xs:element ref="id:ParticipantIdentifier"> <xs:annotation>
<xs:documentation>The participant identifier</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element ref="bd:ServiceMetadataPublisherID"> <xs:annotation>
<xs:documentation>The SMP identifier</xs:documentation> </xs:annotation>
</xs:element>
<xs:element name="Exist" type="xs:boolean">
<xs:annotation>
<xs:documentation>True if the participant is already
registered on the SMP.</xs:documentation> </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>

```

▼ BDMSLAdminService-1.0.xsd

```

<?xml version="1.0" encoding="utf-8"?>
<!--
(C) Copyright 2019 - European Commission | CEF eDelivery
Licensed under the EUPL, Version 1.2 (the "License");
You may not use this file except in compliance with the License. You may obtain a
copy of the License at
https://joinup.ec.europa.eu/sites/default/files/custom-
page/attachment/eupl_v1.2_en.pdf
Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the specific
language governing permissions and limitations under the License.
-->
<xs:schema xmlns="ec:services:wsdl:BDMSL:admin:data:1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bd="http://busdox.org/serviceMetadata/locator/1.0/"
targetNamespace="ec:services:wsdl:BDMSL:admin:data:1.0"
elementFormDefault="qualified" id="BDMSLAdminTypes">
<xs:element name="GenerateReport" type="GenerateReportType"/>
<xs:element name="GenerateReportResponse" type="GenerateReportResponseType"/>
<xs:element name="GenerateInconsistencyReport"
type="GenerateInconsistencyReportType"/>

```



```

<xs:element name="GenerateInconsistencyResponse"
type="GenerateInconsistencyResponseType"/>
<xs:element name="CreateSubDomainRequest" type="SubDomainType"/> <xs:element
name="CreateSubDomainResponse" type="SubDomainType"/> <xs:element
name="CreateDomainCertificateRequest"
type="SubDomainType"/>
<xs:element name="UpdateSubDomainRequest" type="UpdateSubDomainType"/> <xs:element
name="UpdateSubDomainResponse" type="SubDomainType"/> <xs:element
name="DeleteSubDomainRequest" type="DeleteSubDomainType"/> <xs:element
name="DeleteSubDomainResponse" type="SubDomainType"/> <xs:element
name="GetSubDomainRequest" type="GetSubDomainType"/> <xs:element
name="GetSubDomainResponse" type="SubDomainType"/> <xs:element
name="AddSubDomainCertificateRequest"
type="AddDomainCertificateType"/>
<xs:element name="AddSubDomainCertificateResponse"
type="DomainCertificateType"/>
<xs:element name="UpdateSubDomainCertificateRequest"
type="UpdateDomainCertificateType"/>
<xs:element name="UpdateSubDomainCertificateResponse"
type="DomainCertificateType"/>
<xs:element name="ListSubDomainCertificateRequest"
type="ListSubDomainCertificateRequestType"/> <xs:element
name="ListSubDomainCertificateResponse"
type="ListSubDomainCertificateResponseType"/> <xs:element
name="AddTruststoreCertificateRequest"
type="TruststoreCertificateType"/>
  <xs:element name="AddTruststoreCertificateResponse"
type="TruststoreCertificateType"/>
<xs:element name="DeleteTruststoreCertificateRequest"
type="DeleteTruststoreCertificateType"/>
<xs:element name="DeleteTruststoreCertificateResponse"
type="TruststoreCertificateType"/>
<xs:element name="GetTruststoreCertificateRequest"
type="GetTruststoreCertificateType"/>
<xs:element name="GetTruststoreCertificateResponse" type=
"TruststoreCertificateType"/>
<xs:element name="ListTruststoreCertificateAliasesRequest"
type="ListTruststoreCertificateAliasesRequestType"/>
<xs:element name="ListTruststoreCertificateAliasesResponse"
type="ListTruststoreCertificateAliasesResponseType"/>
<xs:element name="AddDNSRecordRequest" type="DNSRecordType"/> <xs:element
name="AddDNSRecordResponse" type="DNSRecordType"/> <xs:element
name="DeleteDNSRecordRequest" type="DeleteDNSRecordType"/> <xs:element
name="DeleteDNSRecordResponse" type="DNSRecordListType"/> <xs:element
name="SetPropertyRequest" type="PropertyType"/> <xs:element
name="SetPropertyResponse" type="PropertyType"/> <xs:element
name="GetPropertyRequest" type="PropertyKeyType"/> <xs:element
name="GetPropertyResponse" type="PropertyType"/> <xs:element
name="DeletePropertyRequest" type="PropertyKeyType"/> <xs:element
name="DeletePropertyResponse" type="PropertyType"/> <xs:element
name="ChangeCertificate" type="ChangeCertificateType"/> <xs:element

```

```

name="ClearCache" type="ClearCacheType"/>
<xs:complexType name="ClearCacheType"/> <xs:complexType name=
"ChangeCertificateType">
<xs:sequence>
<xs:element ref="bd:ServiceMetadataPublisherID">
<xs:annotation>
<xs:documentation>The SMP identifier</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="newCertificatePublicKey" type="xs:base64Binary">
<xs:annotation>
<xs:documentation>The new public key contained in the
certificate.</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="GenerateInconsistencyReportType">
<xs:sequence>
<xs:element name="ReceiverEmailAddress" type="xs:string">
<xs:annotation>
<xs:documentation>Receiver email
address!</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="GenerateInconsistencyResponseType">
<xs:annotation>
<xs:documentation>Status description</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string">
<xs:minLength value="0"/>
<xs:maxLength value="253"/>
</xs:restriction>
</xs:simpleType>
<xs:complexType name="GenerateReportType"> <xs:sequence>
<xs:element name="ReportCode" type="xs:string"> <xs:annotation>
<xs:documentation>Report code. Check documentation for supported
codes!</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="ReceiverEmailAddress" type="xs:string"> <xs:annotation>
<xs:documentation>Receiver email
address!</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="GenerateReportResponseType">
<xs:annotation>

```



```

<xs:documentation>Status description</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="253"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="SubDomainType"> <xs:sequence>
  <xs:element name="SubDomainName" type="SubDomainNameType"> <xs:annotation>
    <xs:documentation>SubDomain name. Name must be unique on SML
    server!</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="SubDomainDescription" minOccurs="0"> <xs:annotation>
    <xs:documentation>Short SubDomain description</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string"> <xs:maxLength value="1024"/>
    </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="DNSZone">
    <xs:annotation>
      <xs:documentation>Domain (dns zone) for
      SubDomain.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="512"/>
        <xs:minLength value="1"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="ParticipantRegularExpression"
    type="ParticipantRegularExpressionType">
    <xs:annotation>
      <xs:documentation>Regex allows specific and described ids only or * instead for
      having wildcards.
    </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="DNSRecordType" type="DNSRecordTypeType"> <xs:annotation>
    <xs:documentation>Type of DNS Record when registering/updating participant, all
    means that both DNS record types are accepted as possible values: [cname, naptr,
    all].
  </xs:documentation>
  </xs:annotation>
  </xs:element>
  <xs:element name="SmpUrlScheme" type="SmpUrlSchemeType">
    <xs:annotation>
      <xs:documentation>

```

Protocol that MUST be used for LogicalAddress when registering new SMP, all means both protocols are accepted possible values: [http, https, all].

```
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="CertSubjectRegularExpression"
type="CertSubjectRegularExpressionType" minOccurs="0">
<xs:annotation>
<xs:documentation>Regex validation of Certificate subject for Issuer based
authorization certificates.
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="CertPolicyOIDs" type="CertPolicyOIDsType" minOccurs="0">
<xs:annotation>
<xs:documentation>User with issuer-authorized SMP
certificate is granted SMP_ROLE only if one of the certificate policy extension
matches the list. Value is a list of certificate policy OIDs separated by ','.
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="MaxParticipantCountForDomain" type="xs:integer" minOccurs="0">
<xs:annotation>
<xs:documentation>Maximum number of participant allowed
to be registered on the domain.
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="MaxParticipantCountForSMP" type="xs:integer" minOccurs="0">
<xs:annotation>
<xs:documentation>Maximum number of participant allowed
to be registered on the SMP.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="PropertyType"> <xs:sequence>
<xs:element name="Key">
<xs:annotation>
<xs:documentation>Property key.</xs:documentation>
</xs:annotation>
<xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="512"/> <xs:minLength
value="1"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Value">
<xs:annotation>
<xs:documentation>Property Value.</xs:documentation>
```

```

        </xs:annotation>
        <xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="4000"/> <xs:minLength
value="1"/>
        </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Description" minOccurs="0">
    <xs:annotation>
        <xs:documentation>Property
description.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
<xs:restriction base="xs:string">
        <xs:maxLength value="4000"/>
        <xs:minLength value="1"/>
    </xs:restriction>
    </xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="PropertyKeyType">
    <xs:sequence>
        <xs:element name="Key">
<xs:annotation>
<xs:documentation>Property key.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
<xs:restriction base="xs:string"> <xs:maxLength value="512"/> <xs:minLength
value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="UpdateSubDomainType">
<xs:sequence>
<xs:element name="SubDomainName" type="SubDomainNameType">
<xs:annotation>
<xs:documentation>SubDomain name. Name must be unique on SML server!
</xs:documentation>
        </xs:annotation>
    </xs:element>
<xs:element name="ParticipantRegularExpression"
type="ParticipantRegularExpressionType" minOccurs="0">
<xs:annotation>
<xs:documentation>Regex allows specific and described
ids only or * instead for having wildcards.
        </xs:documentation>
    </xs:annotation>
</xs:element>

```

```

<xs:element name="DNSRecordType" type="DNSRecordTypeType" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Type of DNS Record when
    registering/updating participant, all means that both DNS
    record types are accepted as possible values:
    [cname, naptr, all].
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="SmpUrlScheme" type="SmpUrlSchemeType" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Protocol that MUST be used for
      LogicalAddress when registering new SMP, all means
      both protocols are accepted possible values: [
      http, https, all].
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="CertSubjectRegularExpression"
      type="CertSubjectRegularExpressionType" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Regex validation of Certificate
        subject for Issuer based authorization
        certificates.
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="CertPolicyOIDs" type="CertPolicyOIDsType" minOccurs="0">
        <xs:annotation>
          <xs:documentation>User with issuer-authorized smp
          certificate is granted SMP_ROLE only if one of the certificate policy extension
          match the list. Value is in list of certificate policy OIDs separated by ', '.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="MaxParticipantCountForDomain" type="xs:integer" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Maximum number of participant allowed
            to be registered on the domain.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="MaxParticipantCountForSMP" type="xs:integer" minOccurs="0">
            <xs:annotation>
              <xs:documentation>Maximum number of participant allowed
              to be registered on the SMP.
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>

```

```

<xs:complexType name="DeleteSubDomainType">
<xs:sequence>
<xs:element name="SubDomainName" type="SubDomainNameType">
  <xs:annotation> <xs:documentation>SubDomain name to be
deleted</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="GetSubDomainType">
<xs:sequence>
<xs:element name="SubDomainName" type="SubDomainNameType">
<xs:annotation> <xs:documentation>SubDomain name to be
retrieved</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="ParticipantRegularExpressionType">
<xs:annotation>
<xs:documentation>Regex allows specific and described ids only
or * instead for having wildcards.
  </xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string">
  <xs:maxLength value="1024"/>
  <xs:minLength value="1"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="CertSubjectRegularExpressionType">
<xs:annotation>
<xs:documentation>User with issuer-authorized smp certificate
is granted SMP_ROLE only if Subject dn match the regular expression.
</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string">
  <xs:maxLength value="1024"/>
  <xs:minLength value="1"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="CertPolicyOIDsType">
<xs:annotation>
<xs:documentation>Value is list of certificate policy OIDs
separated by ','.</xs:documentation> </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="1024"/>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DNSRecordTypeType"> <xs:annotation>
<xs:documentation>Type of DNS Record when registering/updating participant, all

```

means that both DNS record types are accepted as possible values: [cname, naptr, all]. `</xs:documentation>`

```

    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:minLength value="1"/>
      <xs:maxLength value="128"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="SmpUrlSchemeType">
    <xs:annotation>
      <xs:documentation>Protocol that MUST be used for LogicalAddress
when registering new SMP, all means both
protocols are accepted possible values: [http, https, all].
    </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:minLength value="1"/>
      <xs:maxLength value="128"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="SubDomainNameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="255"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CertAliasType"> <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CertificateDomainType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DNSNameType"> <xs:annotation>
  <xs:documentation>Record name. Must end with valid zone name.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="253"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DNSZoneType">
  <xs:annotation>
    <xs:documentation>DNS zone name.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>

```

```

        <xs:maxLength value="253"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CRLDistributionPointsUrlType">
    <xs:restriction base="xs:anyURI">
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="DomainCertificateType"> <xs:sequence>
<xs:element name="SubDomainName" type="SubDomainNameType"> <xs:annotation>
<xs:documentation>SubDomain name</xs:documentation> </xs:annotation>
</xs:element>
<xs:element name="IsRootCertificate" type="xs:boolean">
    <xs:annotation>
    <xs:documentation>Flag if certificate is root certificate</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="IsAdminCertificate" type="xs:boolean"> <xs:annotation>
<xs:documentation>Flag if certificate has admin rights. Only non root certificate
could have admin
rights.
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="CertificateDomainId" type="CertificateDomainType">
    <xs:annotation>
    <xs:documentation>Certificate
identifier</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Alias" type="CertAliasType" minOccurs="0">
<xs:annotation> <xs:documentation>Certificate alias in
truststore</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="CRLDistributionPointsUrl" type="CRLDistributionPointsUrlType"
minOccurs="0">
    <xs:annotation>
    <xs:documentation>Certificate
identifier</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="AddDomainCertificateType">
<xs:sequence>
<xs:element name="SubDomainName" type="SubDomainNameType">
<xs:annotation>
<xs:documentation>SubDomain name</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="IsRootCertificate" type="xs:boolean"> <xs:annotation>

```

```

<xs:documentation>Flag if certificate is root certificate</xs:documentation>
    </xs:annotation>
  </xs:element>
<xs:element name="IsAdminCertificate" type="xs:boolean"> <xs:annotation>
<xs:documentation>Flag if certificate has admin rights. Only non root certificate
can have admin
    rights
  </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="CertificatePublicKey" type="xs:base64Binary"> <xs:annotation>
  <xs:documentation>Domain
certificate.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="Alias" type="CertAliasType" minOccurs="0">
<xs:annotation>
<xs:documentation>If truststore is enabled this is
Certificate alias for the truststore. If alias is
not given value is generated!
  </xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="TruststoreCertificateType">
<xs:sequence>
<xs:element name="Alias" type="CertAliasType" minOccurs="0">
<xs:annotation>
<xs:documentation>Certificate alias.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="CertificatePublicKey" type="xs:base64Binary"> <xs:annotation>
  <xs:documentation>Domain
certificate.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="GetTruststoreCertificateType">
<xs:sequence>
<xs:element name="Alias" type="CertAliasType">
<xs:annotation>
<xs:documentation>Certificate alias.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="DeleteTruststoreCertificateType">
<xs:sequence>
<xs:element name="Alias" type="CertAliasType">

```



```

<xs:annotation>
<xs:documentation>Certificate alias.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="UpdateDomainCertificateType">
<xs:sequence>
<xs:element name="CertificateDomainId"
type="CertificateDomainType">
  <xs:annotation>
    <xs:documentation>Certificate
identifier</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="IsAdminCertificate" type="xs:boolean" minOccurs="0">
<xs:annotation>
<xs:documentation>Flag if certificate has admin rights.
Only non root certificate can have admin
rights
  </xs:documentation>
</xs:annotation>
</xs:element>
  <xs:element name="SubDomainName" type="SubDomainNameType" minOccurs="0">
<xs:annotation>
<xs:documentation>SubDomain name</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="CrlDistributionPoint" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:documentation>Certificate revocation list
DistributionPoint URL</xs:documentation> </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ListSubDomainCertificateRequestType">
<xs:sequence>
<xs:element name="CertificateDomainId"
type="CertificateDomainType" minOccurs="0"> <xs:annotation>
<xs:documentation>'Like' parameter for certificate identifier</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="SubDomainName" type="SubDomainNameType" minOccurs="0">
<xs:annotation>
<xs:documentation>SubDomain name</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ListSubDomainCertificateResponseType">
<xs:sequence>

```

```

<xs:element name="DomainCertificateType"
type="DomainCertificateType" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence>
</xs:complexType>
<xs:complexType name="ListTruststoreCertificateAliasesRequestType">
<xs:sequence>
<xs:element name="ContainsStringInAlias" type="CertAliasType"
minOccurs="0">
<xs:annotation> <xs:documentation>Contains string in
alias</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="ListTruststoreCertificateAliasesResponseType">
<xs:sequence>
<xs:element name="Alias" type="CertAliasType" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="DeleteDNSRecordType">
<xs:sequence>
<xs:element name="Name" type="DNSNameType">
<xs:annotation>
<xs:documentation>Dns name. It must be valid
domain.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="DNSZone" type="DNSZoneType">
<xs:annotation>
<xs:documentation>Dns zone on dns
server.</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="DNSRecordListType">
<xs:sequence>
<xs:element name="DNSRecord" type="DNSRecordType" minOccurs="0"
maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>Dns record list.</xs:documentation> </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="DNSRecordType">
<xs:sequence>
<xs:element name="Type" type="xs:token">
<xs:annotation>
<xs:documentation>Supported dns type: A, CName,
Naptr.</xs:documentation>
</xs:annotation>

```

```

</xs:element>
<xs:element name="Name" type="DNSNameType">
  <xs:annotation>
    <xs:documentation>Dns name. It must be valid
    domain.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="DNSZone" type="DNSZoneType">
  <xs:annotation>
    <xs:documentation>Dns zone on dns
    server.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="Value" type="xs:string">
  <xs:annotation>
    <xs:documentation>Dns Value. For A type it must be IP
    address, for CNAME it must valid Domain, for
    NAPTR it must be regular expresion.
  </xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="Service" type="xs:string" minOccurs="0">
  <xs:annotation>
    <xs:documentation>Service - part of naptr record. If
    not given (for naptr) default value is:
    Meta:SMP
  </xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>

```

3.3.2. List of Valid PEPPOL Issuing Agencies

The network specifies a list of organizations that issue identifiers to network participants. The code is called an International Code Identifier (ICD) and becomes part of the customer identifier. An example of the list ISO/IEC 6523.

The ISO/IEC 6523 list can also be extended by then network governance body as examples:

- **EAS code list:** The European standard on eInvoicing defines which code lists may be used for each business term that has the data type "code", such as electronic address, VAT number, currency, etc.
List can be found on pages: <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Registry+of+supporting+artefacts+to+implement+EN16931>
- **Peppol:** Non-profit organization that provides governaces of the network and a set of document specifications that integrate global business processes. The ICD list can be found on page: <https://docs.peppol.eu/poacc/billing/3.0/codelist/ICD/>

Chapter 4. Support

DomiSML Documentation is maintained by the eDelivery Support Team. For any questions, comments or requests for change, please contact:

- **Email:** ec-edelivery-support@ec.europa.eu
- **Hours:** 8AM to 6PM (Normal EC working days)