EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

SML

Interface Control Document

DomiSML

Version [1.40]

Status [Final]

Date: 07/09/2023

Document Approver(s):

Approver Name	Role
Bogdan DUMITRIU	eDelivery Project officer

Document Reviewers:

Reviewer Name	Role
Joze Rihtarsic	Product owner and developer

Summary of Changes:

Version	Date	Created by	Short Description of Changes
0.01	20/07/2016	Yves ADAM	Initial version
0.02	25/07/2016	Yves ADAM	First delivery for review
0.03	25/08/2016	Yves ADAM	Integrate comments of Flavio SANTOS
0.04	07/09/2016	Yves ADAM	Integrate comments of Pedro TAVARES
0.05	13/09/2016	Yves ADAM	Integrate comments of Adrien FERIAL
1.00	15/09/2016	Yves ADAM	Integrate comments of Flavio SANTOS
1.01	06/03/2017	Flavio SANTOS	Updating *MigrationKey* constraint
1.02	07/03/2017	Tiago MIGUEL	Changes related to Is Alive
1.03	25/04/2017	Flavio SANTOS	Update license to EUPL 1.1
1.04	19/07/2017	Flavio SANTOS	Update participant id error message
1.05	16/01/2018	Flavio SANTOS	Update license to EUPL 1.2
1.06	27/03/2018	CEF Support	Reuse policy notice added. E-SENS BDXL profile replaced by eDelivery BDXL profile.
1.07	25/09/2019	Caroline AEBY	No more standby services
1.08	06/05/2019	Joze RIHTARSIC	Update use cases – BDMSLService, add new usecases BDMSLAdminService, remove the 'Directory user'.
1.09	04/01/2021	Joze RIHTARSIC	Update : - use cases for BDMSLAdminService Truststore management: UC29 – UC32 - BDMSLAdminService wsdl and xsd - update subdomain certificate operation description
1.10	07/04/2022	Caroline AEBY	No more CEF
1.20	15/09/2022	Joze RIHTARSIC	SML 4.2 version updates Add new input parameters to the services: UC21 CreateSubDomain UC22 UpdateSubDomain Add new web-service for validating existence of the participant identifier
1.30	14/10/2022	Caroline AEBY	CEF => Digital Europe Programme heading
1.40	07/09/2023	Joze RIHTARSIC Caroline Aeby	DomiSML 4.3 updates: BDMSL => DomiSML, UC34

			ManageServiceMetadataPublisher added, ICD list removed and replaced URLs to sources. Removed the WSDL and xsd schemas and keep the URL to GIT repo.

Table of Contents

1. Introduction

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.

1.1. Purpose of the Interface Control Document

This document will univocally define the participant's interface to the BDMSL. This document describes the WSDL and the observable behaviour of the interface.

There are 4 interfaces described in this document:

Interface	Description	Version
ManageServiceMetadataService-1.0 .wsdl	Definition of the service to Manage Service Metadata.	1.0
ManageBusinessIdentifierService-1.0 .wsdl	Definition of the service to Manage Participant Identifier's.	1.0
BDMSLService-1.0.wsdl	Definition of the services to Manage Service Metadata and to monitor SML specific to this implementation.	1.0
BDMSLAdminService-1.0.wsdl	Definition of the services for administration of the SML application. Services are specific to this implementation.	1.0

Table 1 - Documented services

1.2. 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

1.3. Audience

This document is intended to:

- 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** will find it 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** will find it 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** will find it 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** can use this document in order to test the interface by following the use cases described; in particular the communications of the BDMSL with the SMP.

1.4. References

The table below provides the reader with the list of reference documents.

#	Document	Contents outline
[REF1]	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.

#	Document	Contents outline
[REF2]	PEPPOL	The OpenPEPPOL Association is responsible for the governance and maintenance of the PEPPOL specifications that enable European businesses to easily deal electronically with any European public sector buyer in their procurement processes.
[REF3]	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.
[REF4]	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.
[REF5]	eDelivery BDXL profile	Specifications of eDelivery BDXLprofile.

#	Document	Contents outline
[REF6]	Policy for use of Identifiers (PEPPOL Transport Infrastructure)	This document describes a PEPPOL policy and guidelines for use of identifiers within the PEPPOL network.

2. Interface Functional Specification

2.1. 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.

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.

2.2. Use cases overview

2.2.1. 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.
Monitor user	The Monitor user who is allowed to call the isAlive service for monitoring health of SML application.

Table 2 - Actors

2.2.2. Use cases diagram

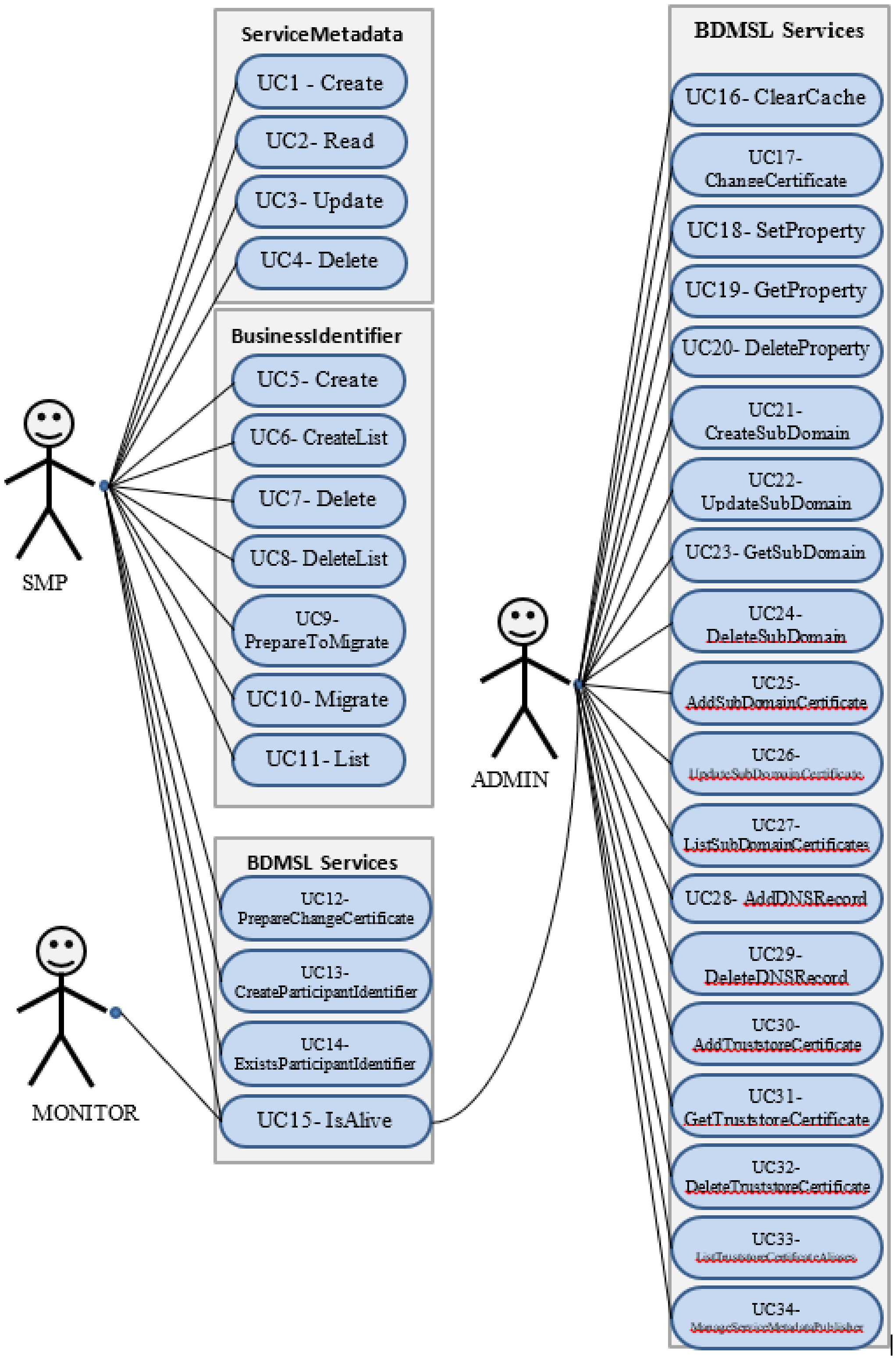


Figure 1 - Use case diagrams

ID	Operation	Actor	Short description
Interface : ManageServiceMetadataService-1.0 .wsdl			
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.
Interface : ManageBusinessIdentifierService-1.0.wsdl			
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

ID	Operation	Actor	Short description
			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.
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

ID	Operation	Actor	Short description
			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 *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.
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.
Interface: BDMSLService-1.0.wsdl			
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.

ID	Operation	Actor	Short description
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)
Interface : BDMSLAdminService-1.0.wsdl			
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 crontask 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.
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

ID	Operation	Actor	Short description
			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

ID	Operation	Actor	Short description
			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 Enable, Disable, Delete or Update ServiceMetadataPublisher instances.

Table 3 – Use cases

2.3. Sample Requests and Responses

In addition to the detailed use cases provided here below, sample requests and responses can be found in the SoapUI project referenced here:

https://ec.europa.eu/digital-building-blocks/code/projects/EDELIVERY/repos/bdmsl/browse/bdmsl-soapui-tests/src/test/resources/SML-soapui-project-TEST.xml

These examples come in addition to the structure definition of the interface (cf. §3.2 - WSDL – Data model) based on their WSDL files and related XSDs.

2.4. Detailed use cases - ManageServiceMetadataService

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

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.

2.4.1. UC1 Create

Description UC01 Create

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

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**	
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
	5	Use case ends

Alternative flows

None.

Exception flows

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)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if the supplied CreateServiceMetadataPublisherService does not contain consistent data (E2) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E3)

2.4.2. UC2 Read

Description UC02 Read		
Retrieves the Service Metadata Publisher record for the service metadata publisher.		
Actors		
SMP		
Preconditions		
C1		The user has a valid certificate
C2		The role associated to the user is ROLE_SMP
C3		The SMP already exists in the SML
C4		Input ReadServiceMetadataPublisherService: ServiceMetadataPublisherID - the unique ID of the Service Metadata Publisher for which the record is required
Basic Flow		
Actor	**Step**	
SMP	1	Invokes the Read() 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
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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
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)

unauthorizedFault (401) - returned if the caller is not authorized to invoke the Create operation (E1)

badRequestFault (400) - returned if the supplied parameter does not contain consistent data (E2)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E3)

notFoundFault (404) - returned if the identifier of the SMP could not be found (E4)

2.4.3. UC3 Update

Description UC03 Update

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

Actors

SMP

Preconditions

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

Basic Flow

Actor	Step	
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

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
2.2.3		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
Successful conditions
The SMP record has been created into the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)
badRequestFault (400) - returned if the supplied UpdateServiceMetadataPublisheServicer does not contain consistent data (E2)
notFoundFault (404) - returned if the identifier of the SMP could not be found (E3)
internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.4.4. UC4 Delete

Description UC04 Delete		
Deletes the Service Metadata Publisher record for the service metadata publisher.		
Actors		
SMP		
Preconditions		
C1		The user has a valid certificate
C2		The role associated to the user is ROLE_SMP
C3		The SMP already exists in the SML
C4		Input DeleteServiceMetadataPublisherService: ServiceMetadataPublisherID - the unique ID of the Service Metadata Publisher to delete
Basic Flow		
Actor	**Step**	
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		
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 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
Successful conditions
The SMP record has been created into the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if the supplied DeleteServiceMetadataPublisherService does not contain consistent data (E2) notFoundFault (404) - returned if the identifier of the SMP could not be found (E3) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5. Detailed use cases - ManageBusinessIdentifierService

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

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.

2.5.1. UC5 - Create

Description UC05 Create
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		
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**	
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		
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**	
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

Successful conditions

The SMP record has been created into the SML

Failure Conditions (HTTP errors)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if the supplied CreateParticipantIdentifier does not contain consistent data (E2)

notFoundFault (404) - returned if the identifier of the SMP could not be found (E3)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.2. UC6 - CreateList

Description UC06 CreateList

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

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 participants don't already exist
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	
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

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

Successful conditions
The SMP records have been created into the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if: o The supplied CreateList does not contain consistent data o The number of participants in the list is greater than 100 (E2) notFoundFault (404) - returned if one of the identifiers of the SMPs or of the participants could not be found (E3) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.3. UC7- Delete

Description UC07 Delete		
Deletes the information that the SML Service holds for a specific Participant Identifier.		
Actors		
SMP		
Preconditions		
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 already exists
C5		Input DeleteParticipantIdentifier: ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP that publishes its metadata
Basic Flow		
Actor	**Step**	
SMP	1	Invokes the Delete() operation
SML	2	Authenticates the user, validates the request, and removes the SMP 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		
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 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 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
Successful conditions
The SMP record has been deleted from the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if the supplied DeleteParticipantIdentifier does not contain consistent data (E2) notFoundFault (404) - returned if the identifier of the SMP or of the participant could not be found (E3) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.4. UC8-DeleteList

Description UC08 DeleteList

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

Actors

SMP

Preconditions

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 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	
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

E1	**The user is not authorized**	
2.1.1	SML	Returns an HTTP error 401 as response to the requester
2.1.2	SMP	Recevies 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	Recevies the error response
2.2.3		Use case ends
E3	**The 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	Recevies the error response
2.3.3		Use case ends
E4	**Any other error occured 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	Recevies the error response
2.4.3		Use case ends

Post conditions
Successful conditions
The SMP records have been deleted from the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)
badRequestFault (400) - returned if: o The supplied DeleteList does not contain consistent data o The number of participants in the list is greater than 100 (E2)
notFoundFault (404) - returned if one of the identifiers of the SMPs or of the participants could not be found (E3)
internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.5. UC9 PrepareToMigrate

Description UC09 PrepareToMigrate

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

C1	The user has a valid certificate
C2	The role associated to the user is ROLE_SMP
C3	The 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**	
SMP	1	Invokes the PrepareToMigrate() 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 prepared to migrate confirmation
	5	Use case ends

Alternative flows

None.

Exception flows

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 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 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

Successful conditions

The SMP record is ready to be migrated into the SML

Failure Conditions (HTTP errors)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if the supplied PrepateMigrationRecord does not contain consistent data (E2)

notFoundFault (404) - returned if the participant identifier or the identifier of the SMP could not be found (E3)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.6. UC10 Migrate

Description UC10 Migrate
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

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 already exists
C5	The PrepareToMigrate operation MUST have been previously invoked for the supplied Participant Identifier, using the same MigrationCode
C6	Input CompleteMigrationRecord: MigrationRecordType - contains the Migration Key and the Participant Identifier which is to be migrated from one Service Metadata Publisher to another.

Basic Flow

Actor	Step	
SMP	1	Invokes the Migrate() 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

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
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

Successful conditions

The participant identifier has been migrated into the SML

Failure Conditions (HTTP errors)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if the supplied CompleteMigrationRecord does not contain consistent data (E2)

notFoundFault (404) -returned if the migration key or the identifier of the SMP could not be found (E3)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.5.7. UC11 List

Description UC11 List		
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		
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
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**	
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		
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
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

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

Failure Conditions (HTTP errors)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if the supplied NextPage does not contain consistent data (E2)

notFoundFault (404) - returned if the next page or the identifier of the SMP could not be found (E3)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.6. Detailed use cases – BDMSLService

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 BDMSLService service.

2.6.1. UC12- PrepareChangeCertificate

Description UC12 PrepareChangeCertificate

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

C1	The 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	
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 a positive response to the requester
SMP	4	Receives the creation confirmation
	5	Use case ends

Alternative flows

None.

Exception flows

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)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if:
o The supplied request does not contain consistent data
o The new certificate is not valid at the date provided in the migrationDate element
o The migrationDate is not in the future.
o The migrationDate is not provided and the "Not Before" date of the new certificate is not in the future
o The migrationDate is not provided and the "Valid From" is in the past
(E2)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E3)

2.6.2. UC13- CreateParticipantIdentifier

Description UC13 CreateParticipantIdentifier

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

Actors

SMP

Preconditions

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 Additional parameter: serviceName

Basic Flow

Actor	Step	
SMP	1	Invokes the CreateParticipantIdentifier() 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

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**	
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
Successful conditions
The SMP record has been created into the SML
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if the supplied CreateParticipantIdentifier does not contain consistent data (E2) notFoundFault (404) - returned if the identifier of the SMP could not be found (E3) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E4)

2.6.3. UC14 - ExistsParticipantIdentifier

Description UC14 ExistsParticipantIdentifier

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

Actors

SMP

Preconditions

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
C5	Input: ServiceMetadataPublisherServiceForParticipantType - contains the Participant Identifier for a given participant and the identifier of the SMP which holds its data

Basic Flow

Actor	Step	
SMP	1	Invokes the ExistsParticipantIdentifier() operation
SML	2	Authenticates the user, validates the request, and adds the SMP record into its configuration database.
SML	3	Returns a response with query participant data and exists parameter set to true if the entry already exists or false if it does not exist
SMP	4	Receives the response
	5	Use case ends

Alternative flows

None.

Exception flows

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 SML successfully read the database status for the participant.
Failure Conditions (HTTP errors)
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) badRequestFault (400) - returned if the supplied CreateParticipantIdentifier does not contain consistent data (E2) internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E3)

2.6.4. UC15 IsAlive

Description UC15 IsAlive		
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		
C1		The user has a valid certificate
C2		The user has the role ROLE_SMP or ROLE_ADMIN, ROLE_MONITOR
C4		Input: None
Basic Flow		
Actor	**Step**	
SMP	1	Invokes the IsAlive() 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		
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 error occurred that prevented the SML to process the request**	
2.2.1	SML	Returns no response to the requester
2.2.2		Use case ends
Post conditions		
Successful conditions		
HTTP 200 OK response sent to the requester.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)		
internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E2)		

2.7. Detailed use cases – BDMSLService

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.

2.7.1. UC16 ClearCache

Description UC16 ClearCache		
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		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: none.
Basic Flow		
Actor	**Step**	
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 reset confirmation
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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 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	SMP	Receives the error response
2.2.3		Use case ends
Post conditions		
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)		

2.7.2. UC17 ChangeCertificate

Description UC16 ChangeCertificate		
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		
C1		The user has a valid certificate
C2		The role associated to the user is 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**	
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 creation confirmation
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
New Certificate is stored Output : none. HTTP 200 OK expected		

Failure Conditions (HTTP errors)

unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)

badRequestFault (400) - returned if:
o The supplied request does not contain consistent data
o The new certificate is not valid at the date provided in the migrationDate element
o The migrationDate is not in the future.
o The migrationDate is not provided and the "Not Before" date of the new certificate is not in the future
o The migrationDate is not provided and the "Valid From" is in the past
(E2)

internalErrorFault (500) - returned if the SML service is unable to process the request for any reason (E3)

2.7.3. UC18 SetProperty

Description UC17 SetProperty		
This operation allows the admin team to change DomiSML property in database: as passwords, DNS url, …. New property is taken into account when crontask refresh the properties at the same time on all nodes in cluster. Crontab properties are refreshed only with restart of DomiSML server.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Property name and property value
Actor	**Step**	
ADMIN	1	Invokes the SetProperty () operation
SML	2	Authenticates the user, validates the request, if property is password – the password is encrypted, stores the property.
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		
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		
Successful conditions		
Output: PropertyType : the property stored in DomiSML		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.4. UC18 SetProperty

Description UC18 SetProperty		
This operation allows the admin team to retrieve DomiSML property from database, DNS url, smtp configuration, etc.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Property name
Actor	**Step**	
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		
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		
Successful conditions		
Output: PropertyType : the property from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.5. UC20 DeleteProperty

Description UC19 DeleteProperty		
This operation allows the admin team to delete DomiSML non mandatory properties from database.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Property name
Actor	**Step**	
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
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: PropertyType : the deleted property from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.6. UC21 CreateSubDomain

Description UC21 CreateSubDomain		
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		
C1		The 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.
Actor	**Step**	
ADMIN	1	Invokes the CreateSubDomain () operation
SML	2	Authenticates the user, validates the request
SML	3	Stores the domain to database.
ADMIN	4	Receives the stored and normalized values for the SubDomain
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: SubDomainType: the stored SubDomainData from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.7. UC22 UpdateSubDomain

Description UC22 UpdateSubDomain		
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		
C1		The 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.
Actor	**Step**	
ADMIN	1	Invokes the UpdateSubDomain () operation
SML	2	Authenticates the user, validates the request
SML	3	Update the subdomain data to database.
ADMIN	4	Receives the stored and normalized values for the SubDomain
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: SubDomainType : the stored SubDomainData from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)		

internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)

2.7.8. UC23 GetSubDomain

Description UC22 GetSubDomain		
This operation allows the admin team to read DomiSML SubDomain properties.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Domain name.
Actor	**Step**	
ADMIN	1	Invokes the GetSubDomain () operation
SML	2	Authenticates the user, validates the request
SML	3	Get the subdomain data from database.
ADMIN	4	Receives the stored values for the SubDomain
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: SubDomainType : the stored SubDomainData from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.9. UC24 DeleteSubDomain

Description UC23 DeleteSubDomain		
This operation allows the admin team to delete empty DomiSML SubDomain.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Domain name.
Actor	**Step**	
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		
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		
Successful conditions		
Output: SubDomainType : the deleted SubDomainData from DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.10. UC25 AddSubDomainCertificate

Description UC24 AddSubDomainCertificate		
This operation allows the admin team to add new Domain certificate to DomiSML SubDomain. Certificate can be flaged as RootPKI certificate and as Admin certificate. Admin certificate can be only the certificate which is not flaged 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: AddTrustsstoreCertificate.		
Actors		
ADMIN		
Preconditions		
C1		The 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.
Actor	**Step**	
ADMIN	1	Invokes the AddSubDomainCertificate () operation
SML	2	Authenticates the user, validates the request
SML	3	Add new SubDomain certificate to the subdomain.
ADMIN	4	Receives added Certificate SubDomain data
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: SubDomainCertificateType: the new SubDomain Certificate Data stored to DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)		

internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)

2.7.11. UC26 UpdateSubDomainCertificate

Description UC25 UpdateSubDomainCertificate		
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		
C1		The 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.
Actor	**Step**	
ADMIN	1	Invokes the UpdateSubDomainCertificate () operation
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		
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		
Successful conditions		
Output: SubDomainCertificateType : the new SubDomain Certificate Data stored to DomiSML database		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.12. UC27 ListSubDomainCertificates

Description UC26 ListSubDomainCertificates		
This operation allows the admin team to search for domain certificate by partial certificate DN value and/or by the Subdomain.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: Optional: Partial Certificate ID, SubDomain name
Actor	**Step**	
ADMIN	1	Invokes the ListSubDomainCertificates () operation
SML	2	Authenticates the user, validates the request
SML	3	Returns list of SubDomain certificate data accordint to search paremeters
ADMIN	4	Receives list of SubDomain certificate data
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: List of SubDomainCertificateTypes which match search criteria.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.13. UC28 AddDNSRecord

Description UC27 AddDNSRecord		
This operation allows the admin team to add new record to DNS for DNS RecordType: A, CNAME and NAPTR.		
Actors		
ADMIN		
Preconditions		
C1		The 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.
Actor	**Step**	
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		
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		
Successful conditions		
Output: DNSRecord data which are stored to database and optionaly to DNS.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - Returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.14. UC29 DeleteDNSRecord

Description UC28 DeleteDNSRecord		
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		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: DNS Record Name, DNS Zone
Actor	**Step**	
ADMIN	1	Invokes the DeleteDNSRecordType () operation
SML	2	Authenticates the user, validates the request
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		
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		
Successful conditions		
Output: List of DNSRecord data, which are deleted from database.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2).		

2.7.15. UC30 AddTruststoreCertificate

Description UC29 AddTruststoreCertificate		
This operation allows the admin team to add X509 certificate to the truststore.		

Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: X509Certificate
Actor	**Step**	
ADMIN	1	Invokes the AddTruststoreCertificate operation
SML	2	Authenticates the user, validates the request
SML	3	Add the certificate to Truststore and generates new Alias if it is not given in the request
ADMIN	4	Receives inserted certificate with alias
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: Certificate with alias.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.16. UC31 GetTruststoreCertificate

Description UC30 GetTruststoreCertificate		
This operation allows the admin team to retrieve X509 certificate from the truststore by the certificate alias.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: certificate alias
Actor	**Step**	
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		
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		
Successful conditions		
Output: certificate with alias		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.17. UC32 DeleteTruststoreCertificate

Description UC31 DeleteTruststoreCertificate		
This operation allows the admin team to delete X509 certificate from the truststore.		
Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: certificate alias
Actor	**Step**	
ADMIN	1	Invokes the DeleteTruststoreCertificate operation
SML	2	Authenticates the user, validates the request
SML	3	Deletes the certificate from the truststiore
ADMIN	4	Receives deleted certificate with alias
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: Deleted X509 Certificate and alias		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.18. UC33 ListTruststoreCertificateAliases

Description UC32 ListTruststoreCertificateAliases
This operation allows the admin team to list all X509 certificates in the truststore.

Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
C2		The user has the role ROLE_ADMIN
C3		Input: search parameter: null or partial certificate alias
Actor	**Step**	
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
ADMIN	4	Receives truststore aliases which matched search parameter
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: List of truststore aliases.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1) internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)		

2.7.19. UC34 ManageServiceMetadataPublisher

Description: UC34 ManageServiceMetadataPublisher

This operation allows the admin team to:

- DISABLE: Admin team can Disable unused SMPs. When SMP is disabled, all records are removed from the DNS, but they are kept in database.
- ENABLE: Admin team can Enable previously disabled SMP. The action generates all DNS records for the SMP. The action is used to restore disabled SMPs.
- DELETE: Admin team can DELETE previously disabled SMP. The action removes all records for the SMP from Database. The action is not reversable.
- UPDATE: Admin team can update logical and physical address of the SMP.

All actions are done asycnhonously, and the email message is sent to admin, when the action is completed. The email address is provided in request.

Actors		
ADMIN		
Preconditions		
C1		The user has a valid certificate
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.
Actor	**Step**	
ADMIN	1	Invokes the ManageServiceMetadataPublisher operation
SML	2	Authenticates the user, validates the request
SML	3	Creates task in new Thread and executes action
ADMIN	4	Receives email when the task is completed
	5	Use case ends
Alternative flows		
None.		
Exception flows		
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		
Successful conditions		
Output: The SMP is enabled/disabled/deleted/updated.		
Failure Conditions (HTTP errors)		
unauthorizedFault (401) - returned if the caller is not authorized to invoke this operation (E1)		

internalErrorFault (500) - returned if the DomiSML service is unable to process the request for any reason (E2)

3. Interface Behavioural Specification

3.1. Sequence diagrams

3.1.1. *ManageServiceMetadataService*

(cf. [REF3])

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

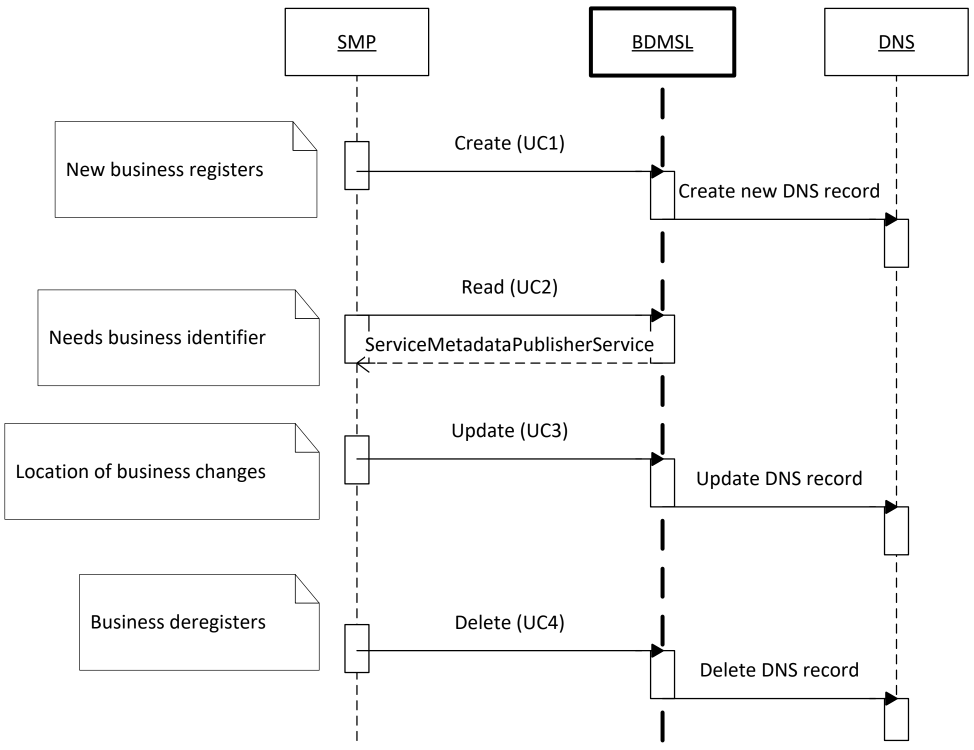


Figure 2 – Sequence diagram: *ManageServiceMetadataService*

3.1.2. ManageBusinessIdentifierService

The ManageParticipantIdentifier interface has the following operations:

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

These services are listed in the sequence diagram below:

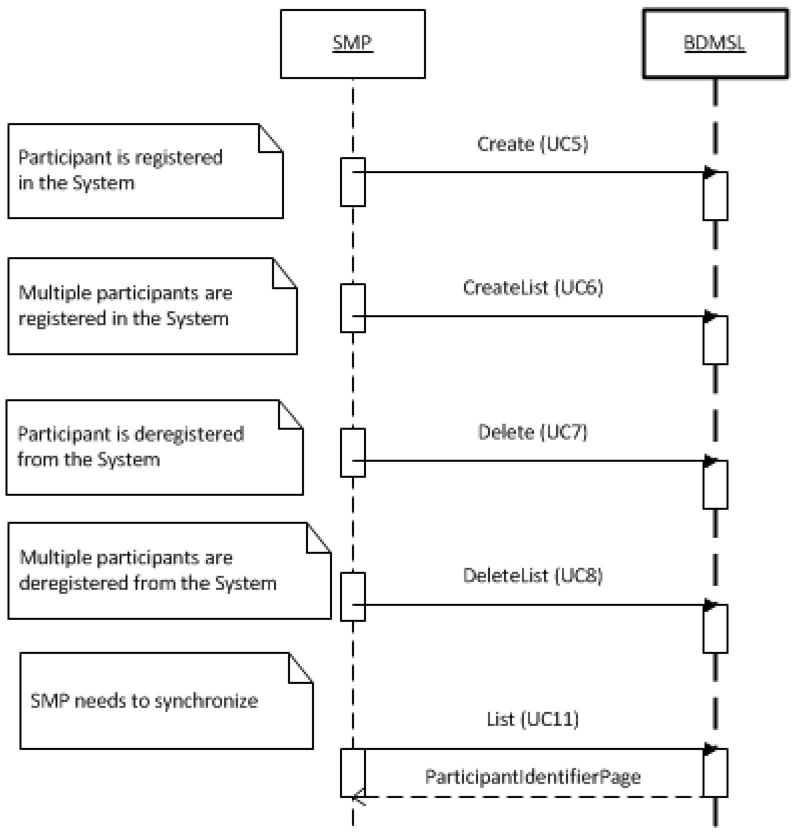

SMP
BDMSL
Participant is registered in the System
Create (UC5)
Multiple participants are registered in the System
CreateList (UC6)
Participant is deregistered from the System
Delete (UC7)
Multiple participants are deregistered from the System
DeleteList (UC8)
SMP needs to synchronize
List (UC11)
ParticipantIdentifierPage

Figure 3 – Sequence diagram: *ManageBusinessIdentifierService*

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:

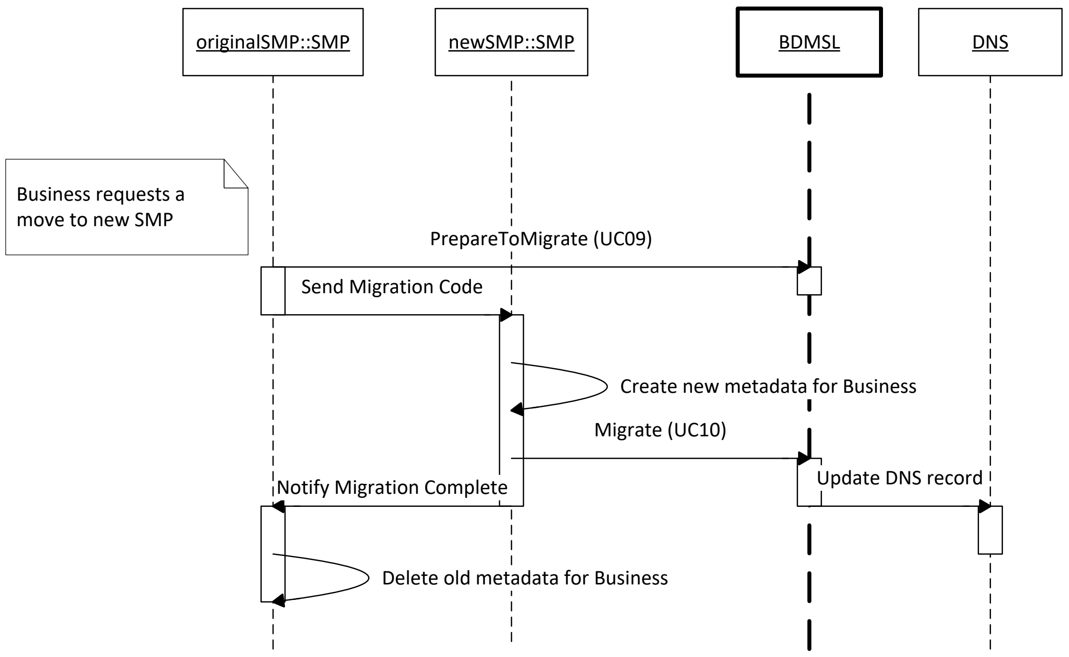

originalSMP::SMP
newSMP::SMP
BDMSL
DNS
Business requests a move to new SMP
PrepareToMigrate (UC09)
Send Migration Code
Create new metadata for Business
Migrate (UC10)
Update DNS record
Notify Migration Complete
Delete old metadata for Business

Figure 4 – Sequence diagram : SMP Migraion

3.1.3. BDMSLService

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

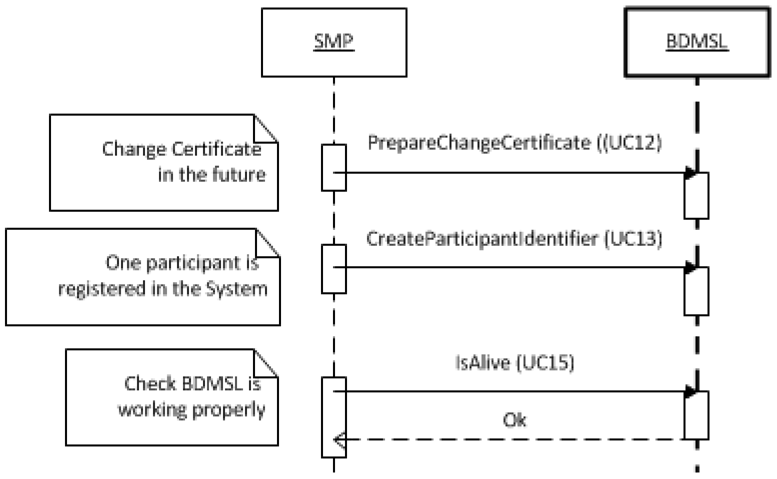

SMP
BDMSL
Change Certificate in the future
PrepareChangeCertificate ((UC12)
One participant is registered in the System
CreateParticipantIdentifier (UC13)
Check BDMSL is working properly
IsAlive (UC15)
Ok

Figure 5 – Sequence diagram: BDMSLService

3.1.4. BDMSLAdminService

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

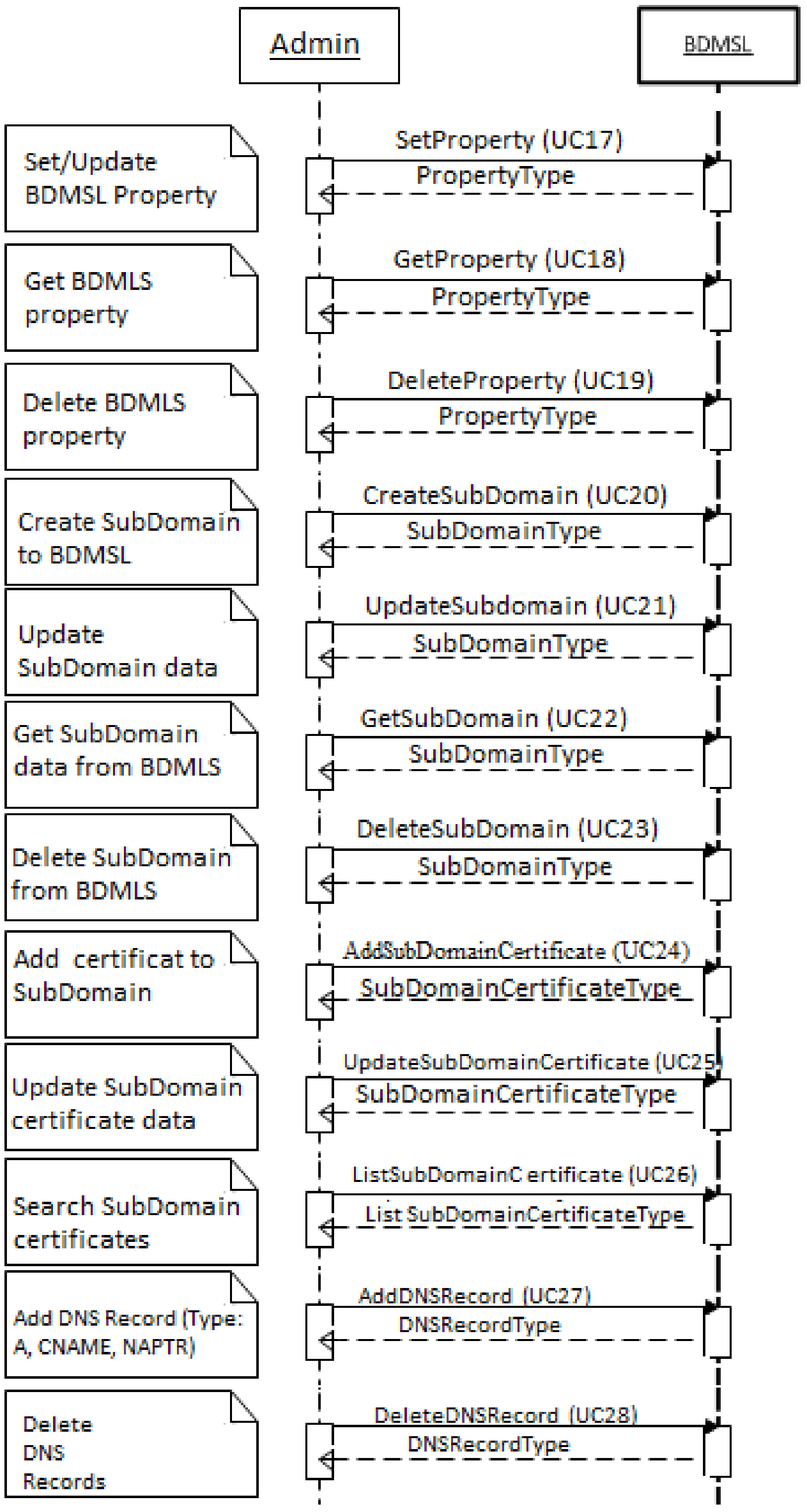
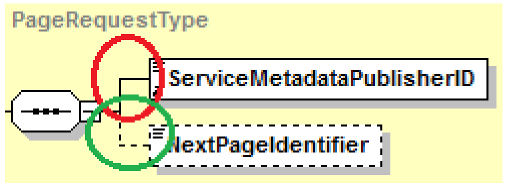


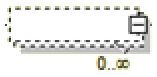
Figure 6 – Sequence diagram: BDMSLService

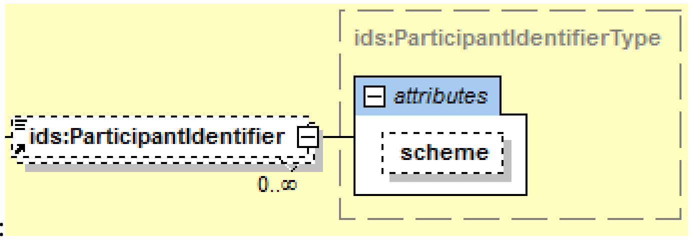
3.2. WSDL – Data model

The interface data model of the DomiSML is described as follows:

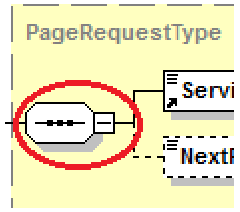
- One paragraph for each of the 3 web services will introduce all their operations
- One paragraph for each operation will specify their Input and Output structures and the fault that these operations may return. In some cases, there is no argument, in which case there is no related structure (the text below will mention "none" in those cases).
- For each input or output structure, the related XSD structure is detailed in a graphical way specifying:
  - The arborescence structure;
  - The mandatory (solid lines) and optional (dashed line) fields.

Example: 

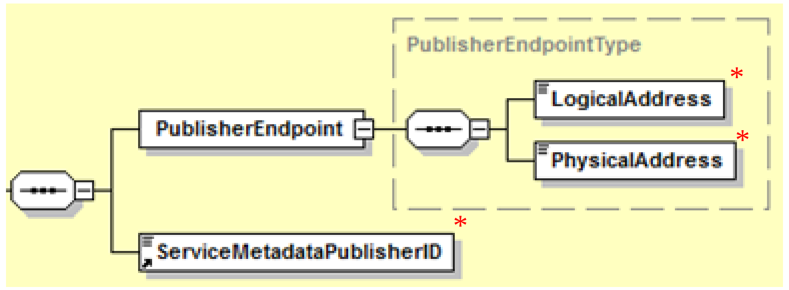
  - The repeated fields and their cardinality (icon  with min/max indication on the bottom right).

Example: 

- The sequences (icon).

Example: 

- The leave attributes as defined in the first paragraph.

Examples (*): 

3.2.1. WSDL model for ManageServiceMetadataService

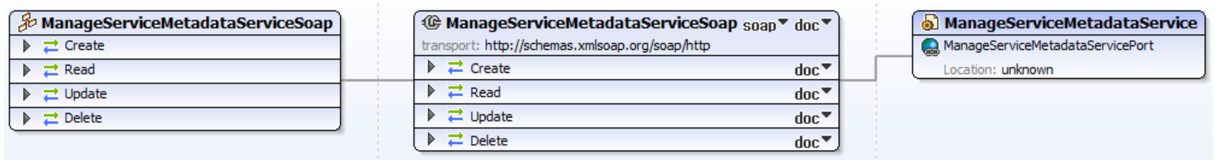
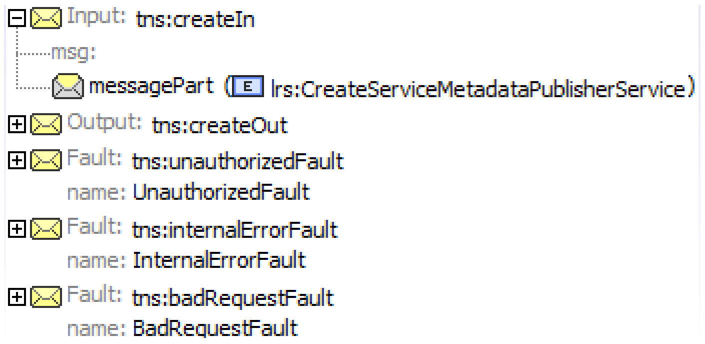

ManageServiceMetadataServiceSoap
Create
Read
Update
Delete
ManageServiceMetadataServiceSoap soap doc
transport: http://schemas.xmlsoap.org/soap/http
Create doc
Read doc
Update doc
Delete doc
ManageServiceMetadataService
ManageServiceMetadataServicePort
Location: unknown

Figure 7 – WSDL ManageServiceMetadataService

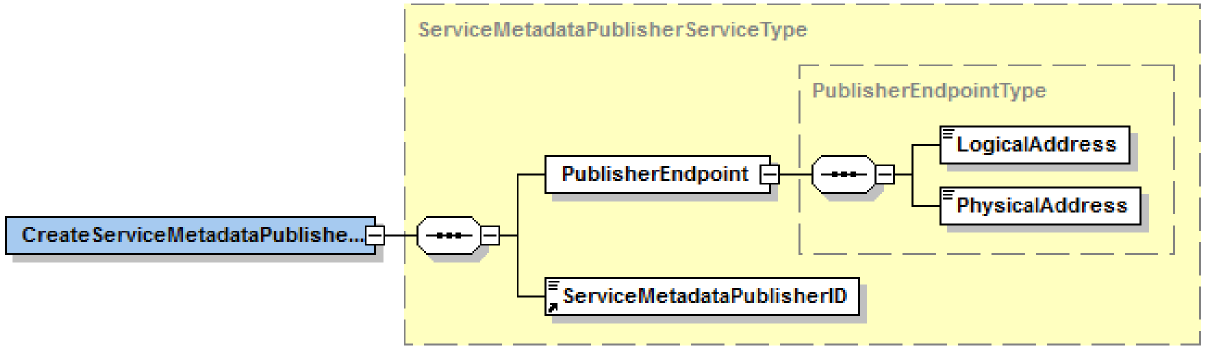
3.2.1.1. Operations Signatures

3.2.1.1.1. Create() Signature



Input: tns:createIn
msg:
messagePart (E lrs:CreateServiceMetadataPublisherService)
Output: tns:createOut
Fault: tns:unauthorizedFault
name: UnauthorizedFault
Fault: tns:internalErrorFault
name: InternalErrorFault
Fault: tns:badRequestFault
name: BadRequestFault

3.2.1.1.2. Create() Input


ServiceMetadataPublisherServiceType
PublisherEndpointType
LogicalAddress
PublisherEndpoint
PhysicalAddress
CreateServiceMetadataPublishe...
ServiceMetadataPublisherID

Argument	Description	Format/XSD/Xpath	Constraint
PublisherEndpoint.LogicalAddress	The logical address of the endpoint (Domain name)	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.
PublisherEndpoint.PhysicalAddress	IP Address of the endpoint	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

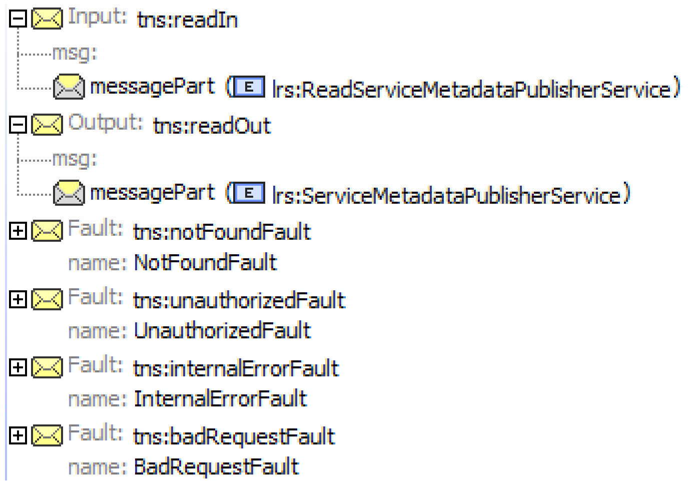
Argument	Description	Format/XSD/Xpath	Constraint
			on a different type of DNS resource records called "URI-enabled Naming Authority Pointer records" (U-NAPTR), which are defined to support Dynamic Delegation Discovery Service (DDDS). The result of a query is a full URI, which can use HTTPS and supports server and optionally client authentication. Must not be null or empty. Must be valid according to IPv4 and well formatted.
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.

Table 4 – Create() input

3.2.1.1.3. Create() Output

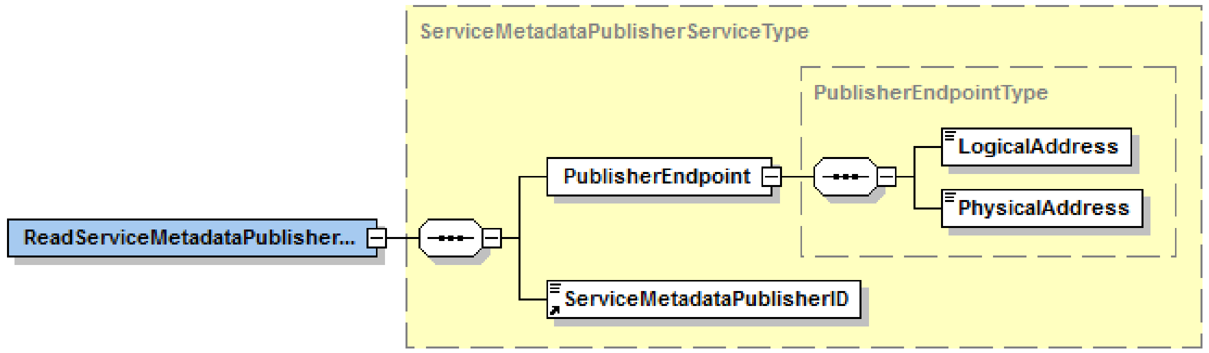
None

3.2.1.1.4. Read() Signature



Input: tns:readIn
msg:
messagePart (E lrs:ReadServiceMetadataPublisherService)
Output: tns:readOut
msg:
messagePart (E lrs:ServiceMetadataPublisherService)
Fault: tns:notFoundFault
name: NotFoundFault
Fault: tns:unauthorizedFault
name: UnauthorizedFault
Fault: tns:internalErrorFault
name: InternalErrorFault
Fault: tns:badRequestFault
name: BadRequestFault

3.2.1.1.5. Read() Input

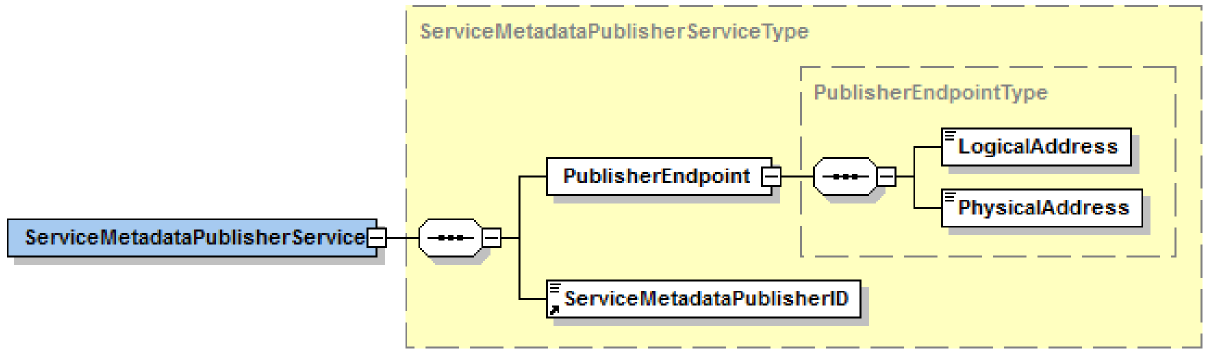

ServiceMetadataPublisherServiceType
PublisherEndpointType
ReadServiceMetadataPublisher...
PublisherEndpoint
LogicalAddress
PhysicalAddress
ServiceMetadataPublisherID

Argument	Description	Format/XSD/Xpath	Constraint
PublisherEndpoint.LogicalAddress	The logical address of the endpoint (Domain name)	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.
PublisherEndpoint.PhysicalAddress	IP Address of the endpoint	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

Argument	Description	Format/XSD/Xpath	Constraint
			authentication. Must not be null or empty. Must be valid according to IPv4 and well formatted.
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.

Table 5 – Read input()

3.2.1.1.6. Read() Output

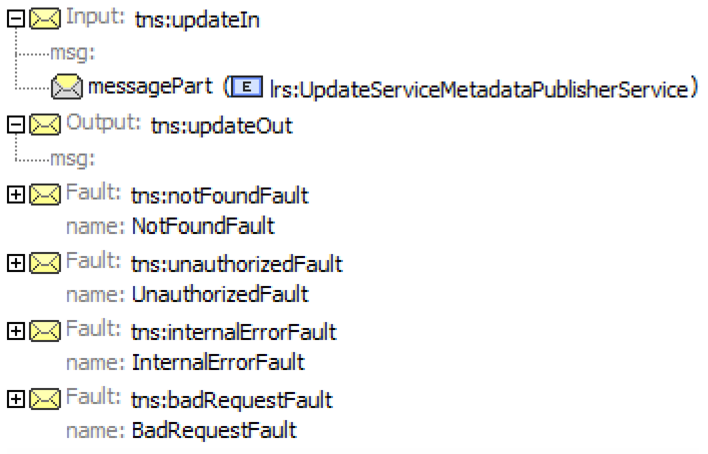

ServiceMetadataPublisherServiceType
PublisherEndpointType
LogicalAddress
PhysicalAddress
PublisherEndpoint
ServiceMetadataPublisherService
ServiceMetadataPublisherID

Argument	Description	Format/XSD/Xpath	Constraint
PublisherEndpoint.LogicalAddress	The logical address of the endpoint (Domain name)	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.
PublisherEndpoint.PhysicalAddress	IP Address of the endpoint	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

Argument	Description	Format/XSD/Xpath	Constraint
			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.
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.

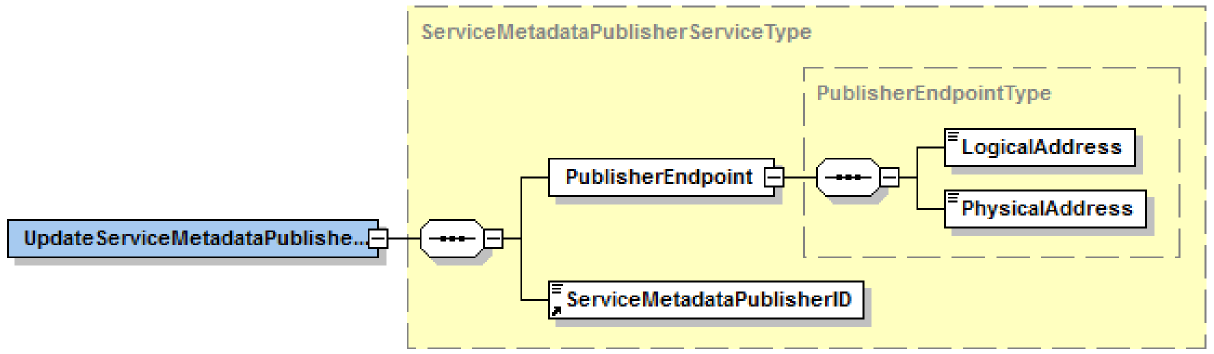
Table 6 – Read() ouput

3.2.1.1.7. Update() Signature



```
Input: tns:updateIn
  msg:
  messagePart (E lrs:UpdateServiceMetadataPublisherService)
Output: tns:updateOut
  msg:
Fault: tns:notFoundFault
  name: NotFoundFault
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.1.1.8. Update() Input


ServiceMetadataPublisherServiceType
PublisherEndpointType
LogicalAddress
PhysicalAddress
PublisherEndpoint
UpdateServiceMetadataPublishe...
ServiceMetadataPublisherID

Argument	Description	Format/XSD/Xpath	Constraint
PublisherEndpoint.LogicalAddress	The logical address of the endpoint (Domain name)	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.
PublisherEndpoint.PhysicalAddress	IP Address of the endpoint	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.

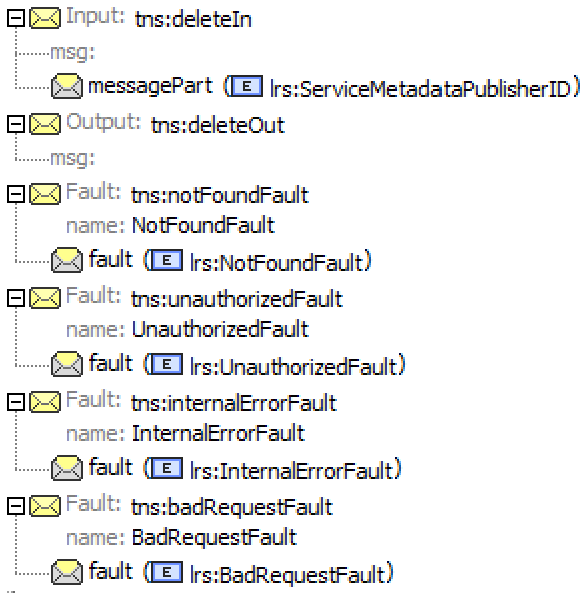
Argument	Description	Format/XSD/Xpath	Constraint
			Must not be null or empty. Must be valid according to IPv4 and well formatted.
ServiceMetadataPublisherID	Unique identifier of the SMP	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']//*[local-name()='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.

Table 7 – Update() input

3.2.1.1.9. Update() Output

None

3.2.1.1.10. Delete() Signature



```
Input: tns:deleteIn
  msg:
    messagePart (E lrs:ServiceMetadataPublisherID)
Output: tns:deleteOut
  msg:
Fault: tns:notFoundFault
  name: NotFoundFault
    fault (E lrs:NotFoundFault)
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
    fault (E lrs:UnauthorizedFault)
Fault: tns:internalErrorFault
  name: InternalErrorFault
    fault (E lrs:InternalErrorFault)
Fault: tns:badRequestFault
  name: BadRequestFault
    fault (E lrs:BadRequestFault)
```

3.2.1.1.11. Delete() Input

Argument	Description	Format/XSD/Xpath	Constraint
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.

Table 8 – Delete() input

3.2.1.1.12. Delete() Output

None

3.2.2. WSDL model for ManageBusinessIdentifierService

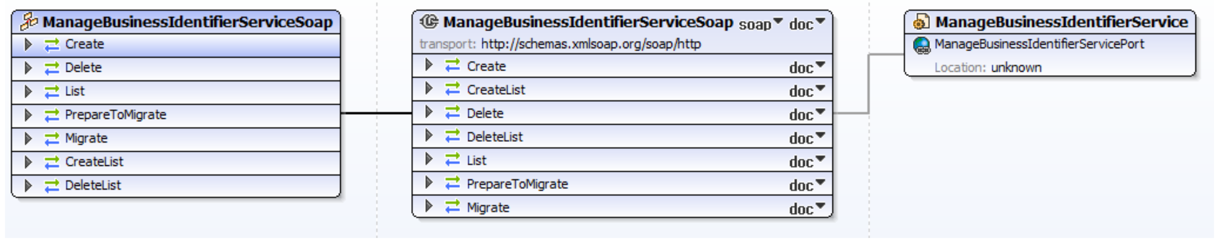
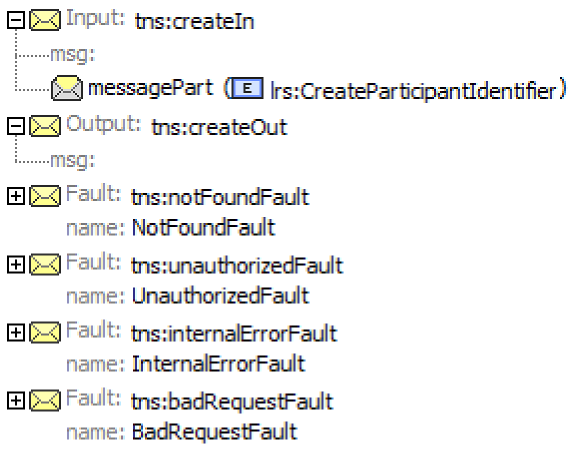

ManageBusinessIdentifierServiceSoap
Create
Delete
List
PrepareToMigrate
Migrate
CreateList
DeleteList
ManageBusinessIdentifierServiceSoap soap doc
transport: http://schemas.xmlsoap.org/soap/http
Create doc
CreateList doc
Delete doc
DeleteList doc
List doc
PrepareToMigrate doc
Migrate doc
ManageBusinessIdentifierService
ManageBusinessIdentifierServicePort
Location: unknown

Figure 8 – WSDL ManageBusinessIdentifierService

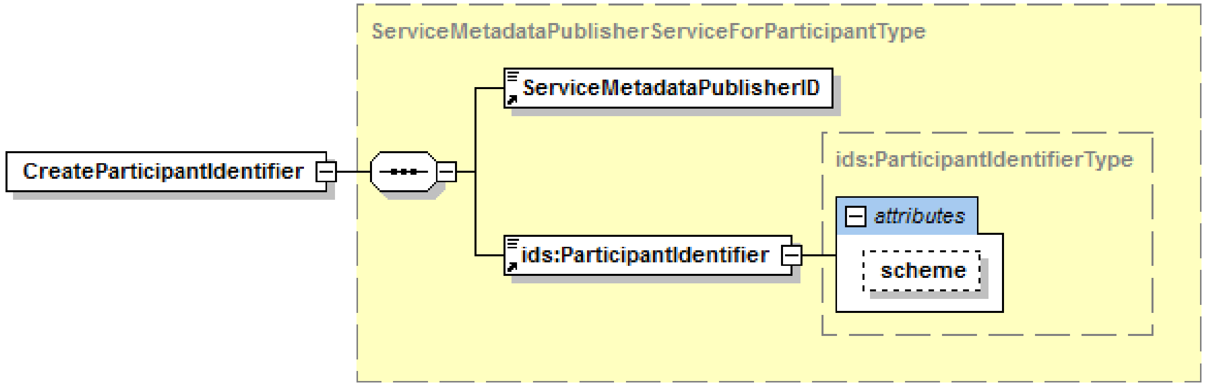
3.2.2.1. Operations Signatures

3.2.2.1.1. Create() Signature



```
Input: tns:createIn
  msg:
    messagePart (E lrs:CreateParticipantIdentifier)
Output: tns:createOut
  msg:
Fault: tns:notFoundFault
  name: NotFoundFault
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.2.1.2. Create() Input


ServiceMetadataPublisherServiceForParticipantType
ServiceMetadataPublisherID
CreateParticipantIdentifier
ids:ParticipantIdentifierType
attributes
ids:ParticipantIdentifier
scheme

Argument	Description	Format/XSD/Xpath	Constraint
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.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier	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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

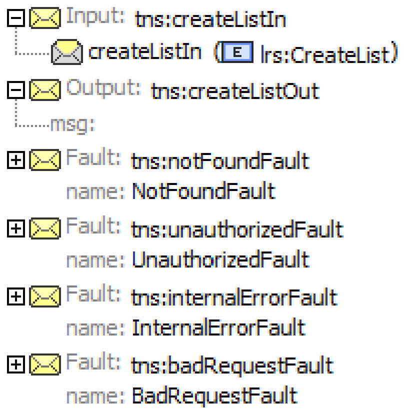
Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure 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]+-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>

Table 9 – Create() input

3.2.2.1.3. Create() Output

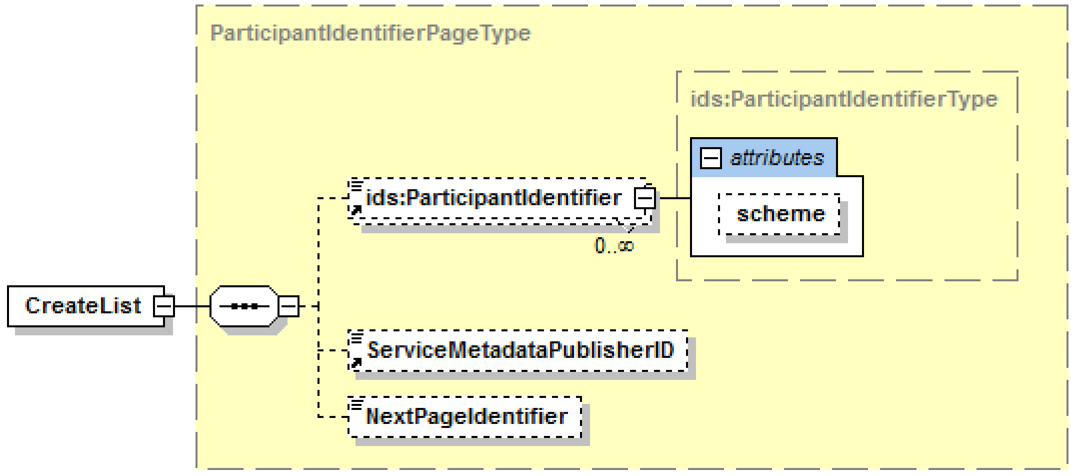
None

3.2.2.1.4. CreateList() Signature



```
Input: tns:createListIn
    createListIn (E lrs:CreateList)
Output: tns:createListOut
    msg:
Fault: tns:notFoundFault
    name: NotFoundFault
Fault: tns:unauthorizedFault
    name: UnauthorizedFault
Fault: tns:internalErrorFault
    name: InternalErrorFault
Fault: tns:badRequestFault
    name: BadRequestFault
```

3.2.2.1.5. CreateList() Input


ParticipantIdentifierPageType
ids:ParticipantIdentifierType
attributes
scheme
ids:ParticipantIdentifier
0..∞
CreateList
ServiceMetadataPublisherID
NextPageIdentifier

Note: the NextPageIndentifier 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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

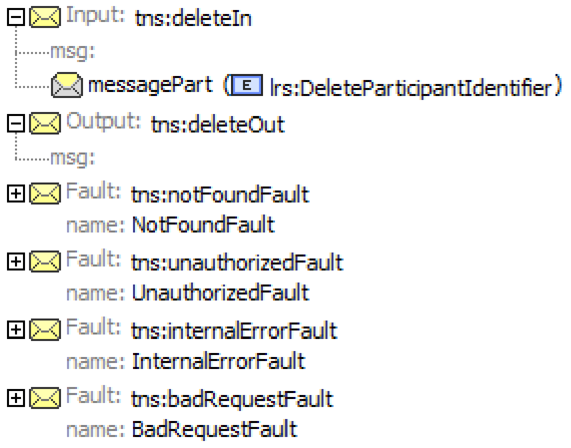
Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme (0..n)	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered 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]+-[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

Table 10 – CreateList() input

3.2.2.1.6. CreateList() Output

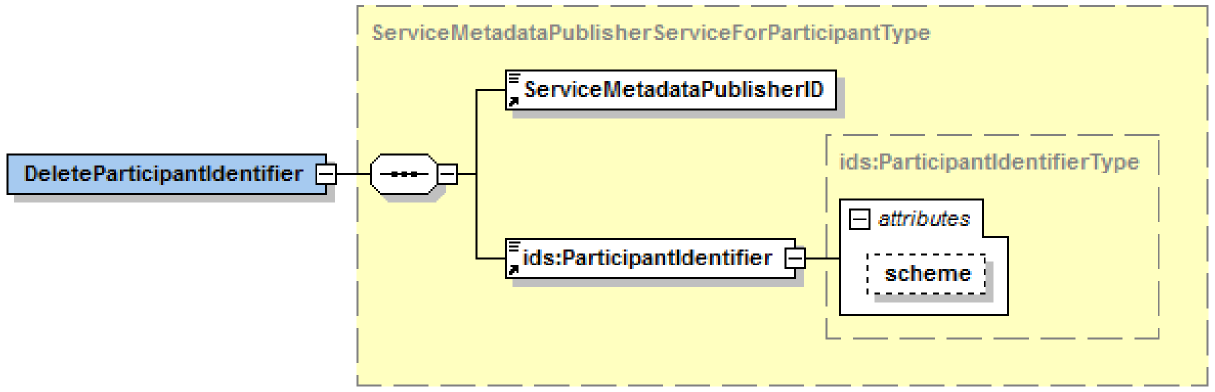
None.

3.2.2.1.7. Delete() Signature



```
Input: tns:deleteIn
  msg:
  messagePart (E lrs:DeleteParticipantIdentifier)
Output: tns:deleteOut
  msg:
Fault: tns:notFoundFault
  name: NotFoundFault
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.2.1.8. Delete() Input


ServiceMetadataPublisherServiceForParticipantType
DeleteParticipantIdentifier
ServiceMetadataPublisherID
ids:ParticipantIdentifierType
attributes
scheme
ids:ParticipantIdentifier

Argument	Description	Format/XSD/Xpath	Constraint
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.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier	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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

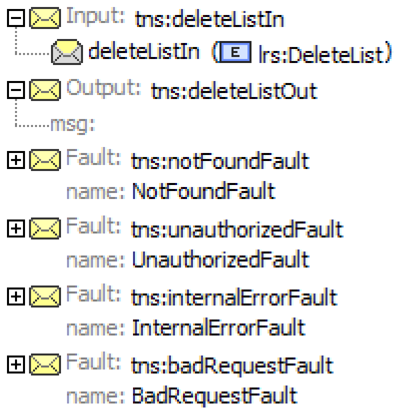
Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure 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]+-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>

Table 11 – Delete() input

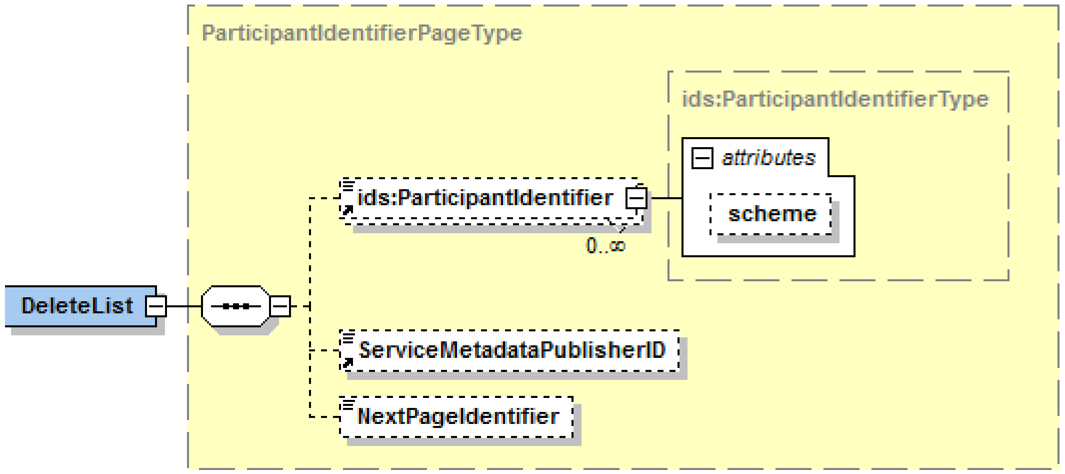
3.2.2.1.9. Delete() Output

None

3.2.2.1.10. DeleteList() Signature


```
Input: tns:deleteListIn
    deleteListIn (E lrs:DeleteList)
Output: tns:deleteListOut
    msg:
Fault: tns:notFoundFault
    name: NotFoundFault
Fault: tns:unauthorizedFault
    name: UnauthorizedFault
Fault: tns:internalErrorFault
    name: InternalErrorFault
Fault: tns:badRequestFault
    name: BadRequestFault
```

3.2.2.1.11. DeleteList() Input


ParticipantIdentifierPageType
ids:ParticipantIdentifierType
attributes
scheme
ids:ParticipantIdentifier
0..∞
DeleteList
ServiceMetadataPublisherID
NextPageIdentifier

<u>Note</u>: the NextPageIndentifier 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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

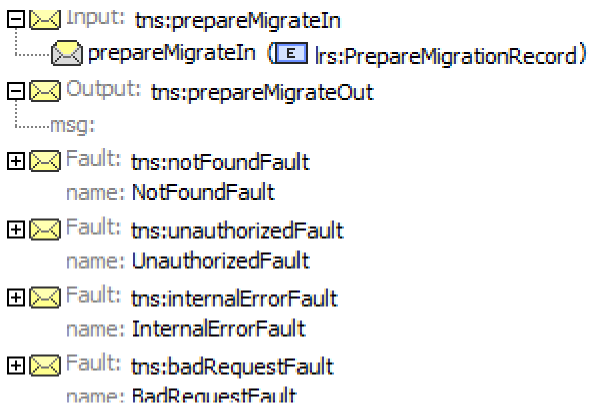
Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme (0..n)	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered 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]+-[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

Table 12 – DeleteList() input

3.2.2.1.12. DeleteList() Output

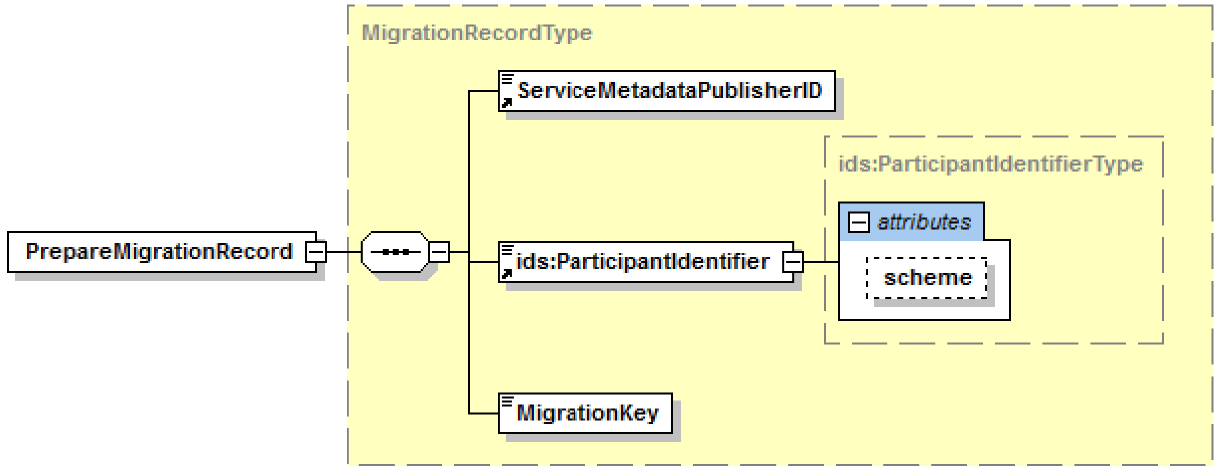
None

3.2.2.1.13. PrepareToMigrate() Signature



Input: tns:prepareMigrateIn
prepareMigrateIn (E lrs:PrepareMigrationRecord)
Output: tns:prepareMigrateOut
msg:
Fault: tns:notFoundFault
name: NotFoundFault
Fault: tns:unauthorizedFault
name: UnauthorizedFault
Fault: tns:internalErrorFault
name: InternalErrorFault
Fault: tns:badRequestFault
name: BadRequestFault

3.2.2.1.14. PrepareToMigrate() Input


MigrationRecordType
ServiceMetadataPublisherID
ids:ParticipantIdentifierType
attributes
PrepareMigrationRecord
ids:ParticipantIdentifier
scheme
MigrationKey

Argument	Description	Format/XSD/Xpath	Constraint
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.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier	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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme	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]+-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>
MigrationKey	String which is a unique key controlling the migration of the metadata for a given ParticipantIdentifier from one Service Metadata Publisher to another.	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='MigrationRecordType']/*[local-name()='sequence']/*[local-name()='element' and @name='MigrationKey']	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; This code must not be null nor empty; This code must contain: • 8 characters minimum • 24 characters maximum • 2 Special Characters @#$%()[]{}*^-!~|+= • 2 Upper Case letters minimum

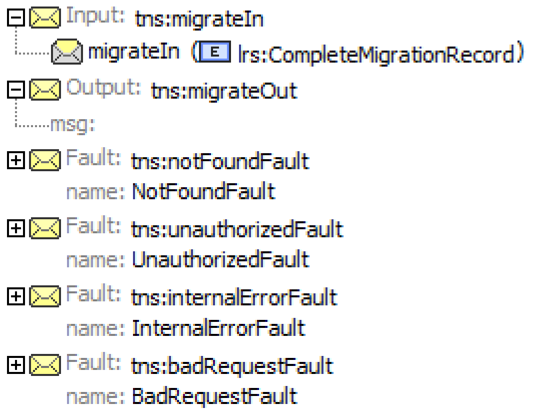
Argument	Description	Format/XSD/Xpath	Constraint
			• 2 Lower Case letters minimum • 2 Numbers minimum • No white spaces

Table 13 – PrepareToMigrate() input

3.2.2.1.15. PrepareToMigrate() Output

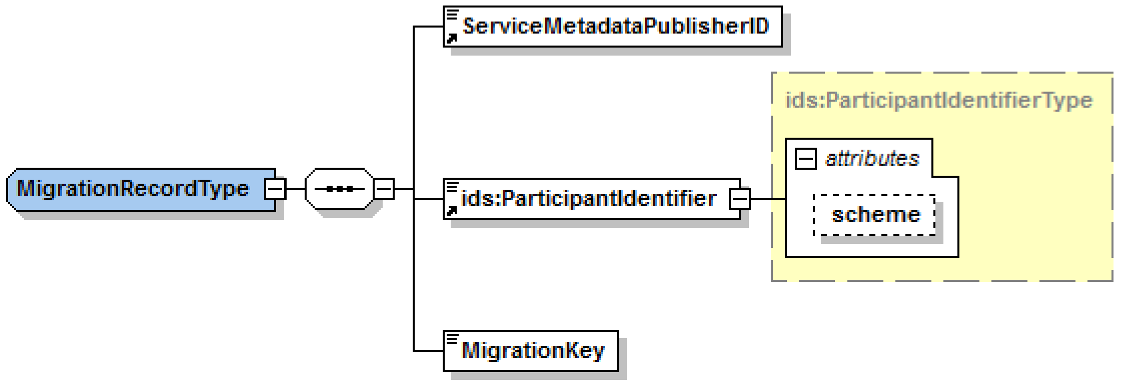
None

3.2.2.1.16. Migrate() Signature



```
Input: tns:migrateIn
    migrateIn (E lrs:CompleteMigrationRecord)
Output: tns:migrateOut
    msg:
Fault: tns:notFoundFault
    name: NotFoundFault
Fault: tns:unauthorizedFault
    name: UnauthorizedFault
Fault: tns:internalErrorFault
    name: InternalErrorFault
Fault: tns:badRequestFault
    name: BadRequestFault
```

3.2.2.1.17. Migrate() Input


ServiceMetadataPublisherID
ids:ParticipantIdentifierType
attributes
MigrationRecordType
ids:ParticipantIdentifier
scheme
MigrationKey

Argument	Description	Format/XSD/Xpath	Constraint
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.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier	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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

Argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme	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]+-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>
MigrationKey	String which is a unique key controlling the migration of the metadata for a given ParticipantIdentifier from one Service Metadata Publisher to another.	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='MigrationRecordType']/*[local-name()='sequence']/*[local-name()='element' and @name='MigrationKey']	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; This code must not be null nor empty; This code must contain: • 8 characters minimum • 24 characters maximum • 2 Special Characters @#$%()[]{}*^-!~|+= • 2 Upper Case letters minimum

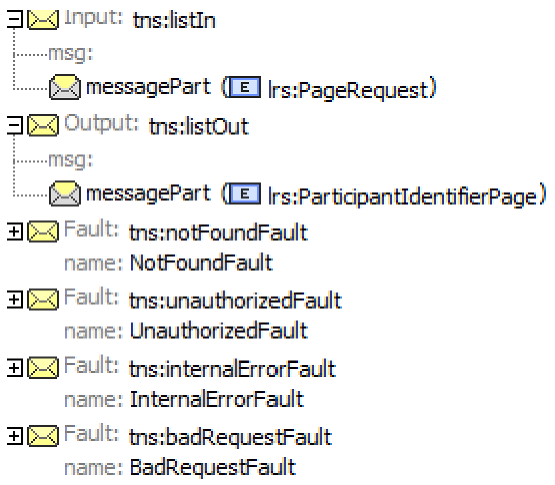
Argument	Description	Format/XSD/Xpath	Constraint
			• 2 Lower Case letters minimum • 2 Numbers minimum • No white spaces

Table 14 – Migrate() input

3.2.2.1.18. Migrate() Output

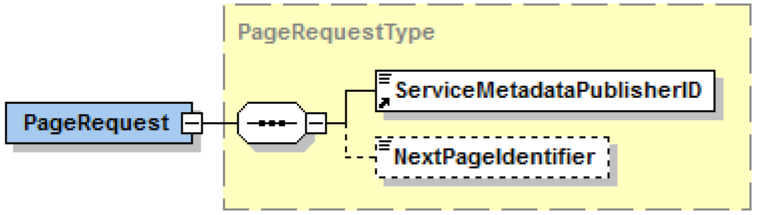
None

3.2.2.1.19. List() Signature



```
Input: tns:listIn
  msg:
    messagePart (E lrs:PageRequest)
Output: tns:listOut
  msg:
    messagePart (E lrs:ParticipantIdentifierPage)
Fault: tns:notFoundFault
  name: NotFoundFault
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.2.1.20. List() Input

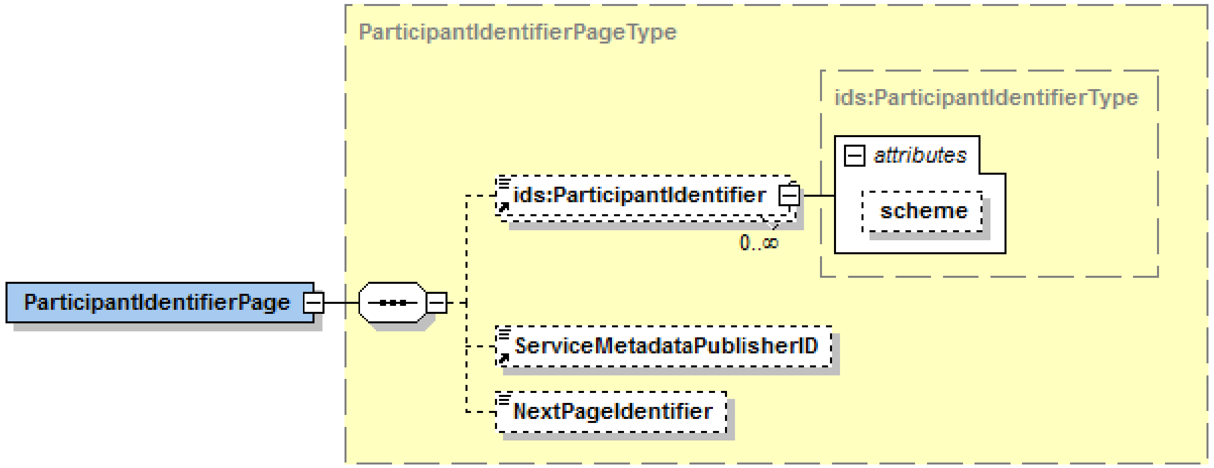


argument	Description	Format/XSD/Xpath	Constraint
ServiceMetadataPublisherID	Unique identifier of the SMP	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ServiceMetadataPublisherServiceForParticipantType']//*[local-name()='sequence']/*[local-name()='element' 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	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	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='PageRequestType']/*[local-name()='sequence']/*[local-name()='element' and @name='NextPageIdentifier']	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).

argument	Description	Format/XSD/Xpath	Constraint
	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.		

Table 15 – List() input

3.2.2.1.21. List() Output


ParticipantIdentifierPageType
ids:ParticipantIdentifierType
attributes
scheme
ids:ParticipantIdentifier
0..∞
ParticipantIdentifierPage
ServiceMetadataPublisherID
NextPageIdentifier

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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: 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]+-[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.

argument	Description	Format/XSD/Xpath	Constraint
NextPageIdentifier	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.	xs:string ServiceMetadataLocatorTypes-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='PageRequestType']/*[local-name()='sequence']/*[local-name()='element' and @name='NextPageIdentifier']	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).

Table 16 – List() output

3.2.3. WSDL model for BDMSLService

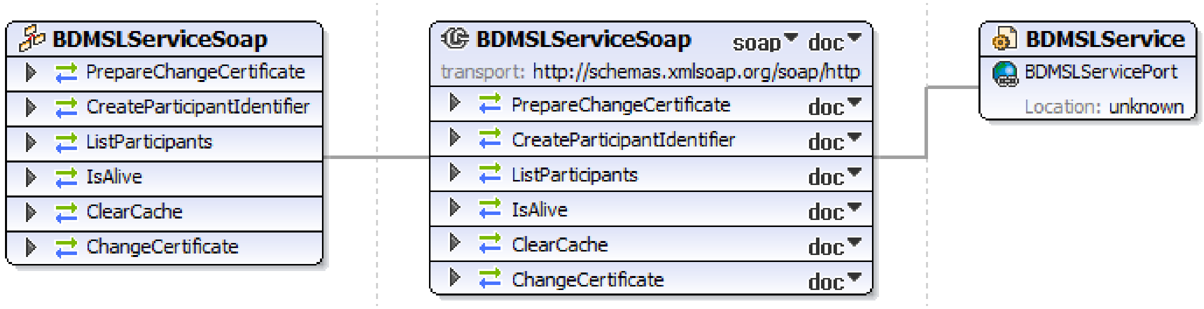
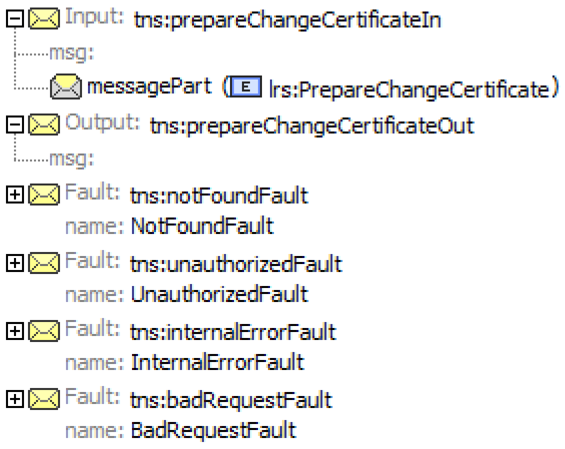


Figure 9 – WSDL BDMSLService

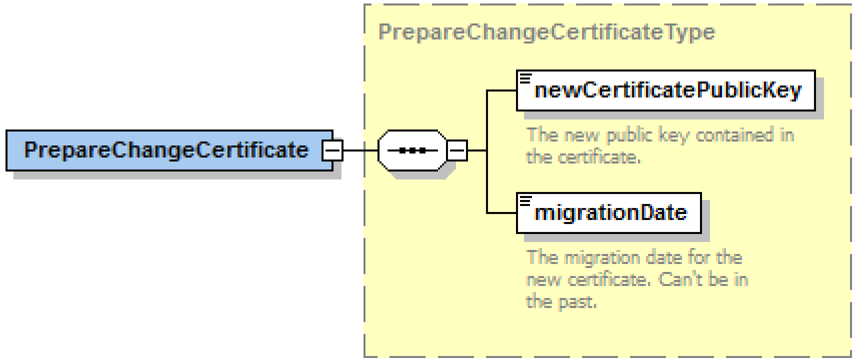
3.2.3.1. Operations Signatures

3.2.3.1.1. PrepareChangeCertificate() Signature



Input: tns:prepareChangeCertificateIn
msg:
messagePart (E lrs:PrepareChangeCertificate)
Output: tns:prepareChangeCertificateOut
msg:
Fault: tns:notFoundFault
name: NotFoundFault
Fault: tns:unauthorizedFault
name: UnauthorizedFault
Fault: tns:internalErrorFault
name: InternalErrorFault
Fault: tns:badRequestFault
name: BadRequestFault

3.2.3.1.2. PrepareChangeCertificate() Input


PrepareChangeCertificateType
PrepareChangeCertificate
newCertificatePublicKey
The new public key contained in the certificate.
migrationDate
The migration date for the new certificate. Can't be in the past.

argument	Description	Format/XSD/Xpath	Constraint
newCertificatePublicKey	The new public key contained in the certificate	base64Binary BDMSLService-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='PrepareChangeCertificateType']/*[local-name()='sequence']/*[local-name()='element' and @name='newCertificatePublicKey']	Must be valid and belong to the list of authorized root certificate aliases.

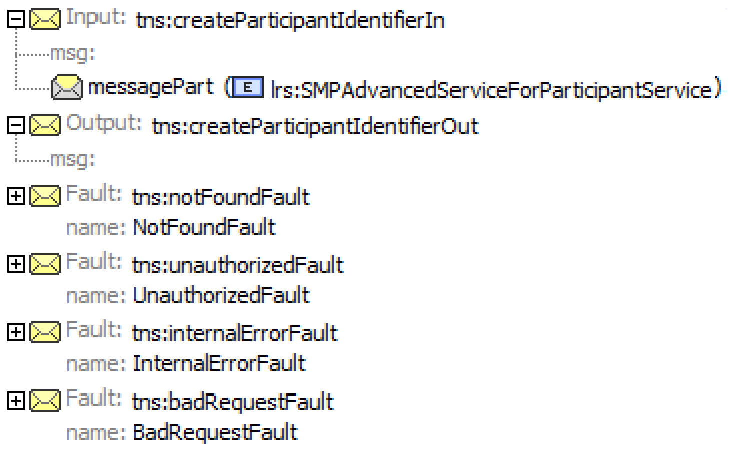
argument	Description	Format/XSD/Xpath	Constraint
migrationDate	The migration date for the new certificate	xs:date BDMSLService-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='PrepareChangeCertificateType']/*[local-name()='sequence']/*[local-name()='element' and @name='migrationDate']	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.

Table 17 – PrepareChangeCertificate() input

3.2.3.1.3. PrepareChangeCertificate() Output

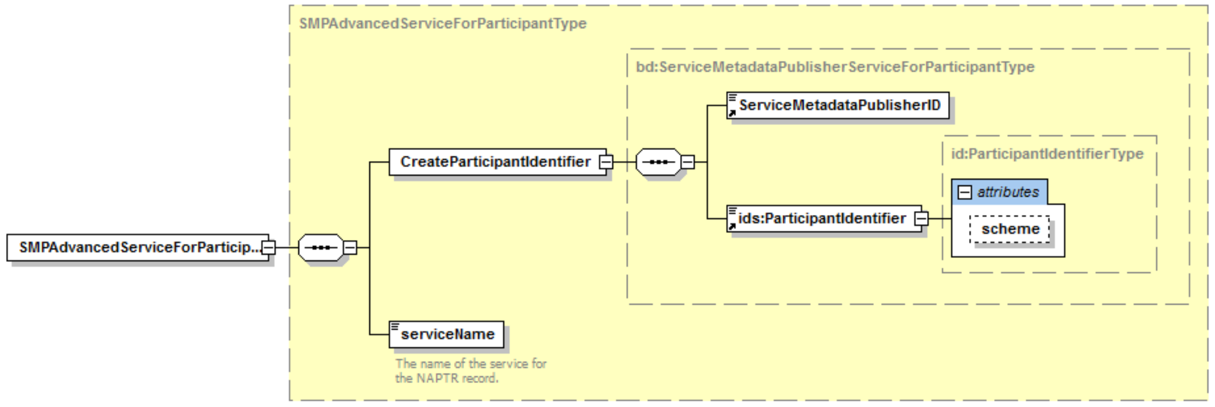
None

3.2.3.1.4. CreateParticipantIdentifier() Signature



```
Input: tns:createParticipantIdentifierIn
  msg:
    messagePart (E lrs:SMPAdvancedServiceForParticipantService)
Output: tns:createParticipantIdentifierOut
  msg:
Fault: tns:notFoundFault
  name: NotFoundFault
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.3.1.5. CreateParticipantIdentifier() Input


SMPAdvancedServiceForParticipantType
bd:ServiceMetadataPublisherServiceForParticipantType
ServiceMetadataPublisherID
id:ParticipantIdentifierType
CreateParticipantIdentifier
attributes
ids:ParticipantIdentifier
scheme
SMPAdvancedServiceForParticip...
serviceName
The name of the service for the NAPTR record.

argument	Description	Format/XSD/Xpath	Constraint
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.

argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier	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 [REF6]. Example: 0088:4035811991014 Identifiers-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='ParticipantIdentifierType']	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 (cf. §0 - "List of valid issuing agencies for PEPPOL") Remark: the participant identifier is case insensitive.

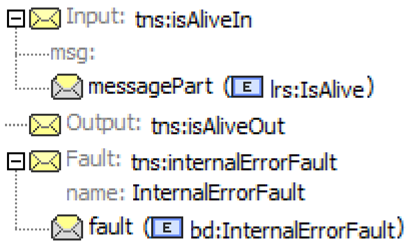
argument	Description	Format/XSD/Xpath	Constraint
ParticipantIdentifier.scheme	The scheme of the participant identifier	Identifier schemes for all schemed identifier types (participants, documents, profiles, transports) may be defined outside of this specification. Any instance of a 4-cornered infrastructure 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]+-[a-zA-Z0-9] Which defines the following parts: <domain>-<identifier Area>-<identifier type>
serviceName	The name of the service for the NAPTR record.	xs:string BDMSLService-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='SMPAdvancedServiceForParticipantType']/*[local-name()='sequence']/*[local-name()='element' and @name='serviceName']	No constraint is enforced by the DomiSML

Table 18 – CreateParticipantIdentifier() input

3.2.3.1.6. CreateParticipantIdentifier() Output

None

3.2.3.1.7. IsAlive() Signature



```
Input: tns:isAliveIn
  msg:
    messagePart (E lrs:IsAlive)
Output: tns:isAliveOut
Fault: tns:internalErrorFault
  name: InternalErrorFault
    fault (E bd:InternalErrorFault)
```

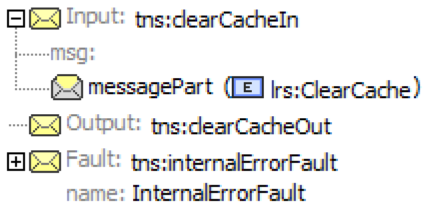
3.2.3.1.8. IsAlive() Input

(empty)

3.2.3.1.9. IsAlive() Output

None

3.2.3.1.10. ClearCache() Signature



```
Input: tns:clearCacheIn
  msg:
    messagePart (E lrs:ClearCache)
Output: tns:clearCacheOut
Fault: tns:internalErrorFault
  name: InternalErrorFault
```

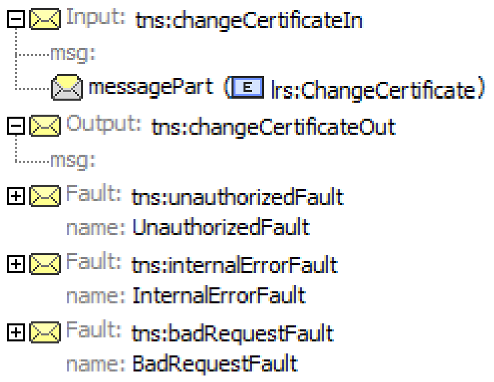
3.2.3.1.11. ClearCache() Input

none

3.2.3.1.12. ClearCache() Output

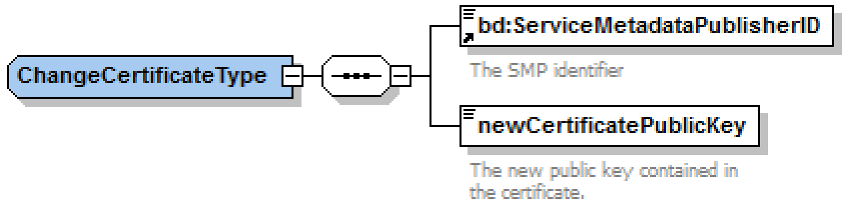
none

3.2.3.1.13. ChangeCertificate() Signature



```
Input: tns:changeCertificateIn
  msg:
    messagePart (E lrs:ChangeCertificate)
Output: tns:changeCertificateOut
  msg:
Fault: tns:unauthorizedFault
  name: UnauthorizedFault
Fault: tns:internalErrorFault
  name: InternalErrorFault
Fault: tns:badRequestFault
  name: BadRequestFault
```

3.2.3.1.14. ChangeCertificate() Input


ChangeCertificateType
bd:ServiceMetadataPublisherID
The SMP identifier
newCertificatePublicKey
The new public key contained in the certificate.

argument	Description	Format/XSD/Xpath	Constraint
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.
newCertificatePublicKey	The new public key contained in the certificate	base64Binary BDMSLService-1.0.xsd /*[local-name()='schema']/*[local-name()='complexType' and @name='PrepareChangeCertificateType']/*[local-name()='sequence']/*[local-name()='element' and @name='newCertificatePublicKey']	Must be valid and belong to the list of authorized root certificate aliases.

Table 19 – ChangeCertificate() input

3.2.3.1.15. ChangeCertificate() Output

None

3.2.4. Faults

3.2.4.1. 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
- internaltFault: any other error occurred during the processing of the request

3.2.4.2. Error Codes

Whenever a fault occurs, more details on the source of the error will 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

Table 20 – Error Codes

3.2.4.3. Faults specific usages

The table below shows the applicability of these errors for all operations specified in this document:

	ManageServiceMetadata				ManageBusinessIdentifier							BDMSLService		
	Create	Read	Update	Delete	Create	CreateList	Delete	DeleteList	PrepareToMigrate	Migrate	List	PrepareChangeCertificate	CreateParticipantIdentifier	IsAlive
notFoundFault:		X	X	X	X	X	X	X	X	X	X	X	X	
unauthorizedFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X
internalErrorFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X
badRequestFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 21 – Fault specific usage

The table below shows the applicability of these errors for all operations specified in this document:

	BDMSLAdminService																		
	SetProperty	GetProperty	DeleteProperty	CreateSuDomain	UpdateSubDomain	GetSubDomain	DeleteSubDomain	AddSubDomainCertificate	UpdateSubDomainCertificate	ListSubDomainCertificates	AddDNSRecord	DeleteDNSRecord	ClearCache	ChangeCertificate	AddTruststoreCertificate	GetTruststoreCertificate	DeleteTruststoreCertificate	ListTruststoreCertificateAliases	ManageServiceMetadaPublisher
notFoundFault:	X	X	X	X	X	X	X	X	X		X	X		X		X	X		X
unauthorizedFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
internalErrorFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
badRequestFault:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 22 – Fault specific usage

3.2.4.4. 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 'testSMPPrepareToMigrate520' doesn't exist.
[14ojYP8Op4vWs78XmcVtLBVFsbPat1mOE9h8HChQW0O48XbOOZXu!-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 'testSMPPrepareToMigrate520' 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>
```

4. Annexes

4.1. Interface Message standards

The WSDL and XSDdocuments 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

4.1.1. WSDL's

4.1.1.1. 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="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="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="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="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="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="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>
```

4.1.1.2. ManageServiceMetadataService

```
<?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.
-->
<wsdl:definitions xmlns:tns="http://busdox.org/serviceMetadata/ManageServiceMetadataService/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="ManageServiceMetadataService"
targetNamespace="http://busdox.org/serviceMetadata/ManageServiceMetadataService/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/ManageServiceMetadataService/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:CreateServiceMetadataPublisherService"/>
    </wsdl:message>
    <wsdl:message name="createOut">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    </wsdl:message>
    <wsdl:message name="readIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart" element="lrs:ReadServiceMetadataPublisherService"/>
    </wsdl:message>
    <wsdl:message name="readOut">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart" element="lrs:ServiceMetadataPublisherService"/>
    </wsdl:message>
    <wsdl:message name="updateIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart" element="lrs:UpdateServiceMetadataPublisherService"/>
    </wsdl:message>
    <wsdl:message name="updateOut">
        <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:ServiceMetadataPublisherID"/>
    </wsdl:message>
    <wsdl:message name="deleteOut">
        <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="ManageServiceMetadataServiceSoap">
        <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:unauthorizedFault" name="UnauthorizedFault"/>
            <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
            <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
        </wsdl:operation>
        <wsdl:operation name="Read">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
            <wsdl:input message="tns:readIn"/>
            <wsdl:output message="tns:readOut"/>
            <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="Update">
            <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
            <wsdl:input message="tns:updateIn"/>
            <wsdl:output message="tns:updateOut"/>
            <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:portType>
        <wsdl:binding name="ManageServiceMetadataServiceSoap" type="tns:ManageServiceMetadataServiceSoap">
            <soap11:binding transport="http://schemas.xmlsoap.org/soap/http"/>
            <wsdl:operation name="Create">
                <soap11:operation soapAction="http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/:createIn"
style="document"/>
                <wsdl:input>
                    <soap11:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap11:body use="literal"/>
                </wsdl:output>
                <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="Read">
                <soap11:operation soapAction="http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/:readIn"
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="Update">
                <soap11:operation soapAction="http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/:updateIn"
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/ManageServiceMetadataService/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="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="ManageServiceMetadataService">
            <wsdl:port name="ManageServiceMetadataServicePort" binding="tns:ManageServiceMetadataServiceSoap">
                <soap:address location="unknown"/>
            </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

4.1.1.3. BDMSLService

```
<!--
(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="ec:services:wsdl:BDMSL:1.0"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:lrs="ec:services:wsdl:BDMSL:data:1.0"
                  xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
name="BDMSLService"
                  targetNamespace="ec:services:wsdl:BDMSL:1.0"

xmlns:bd="http://busdox.org/serviceMetadata/locator/1.0/">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:types>
        <s:schema>
```

```
            <s:import namespace="ec:services:wsdl:BDMSL:data:1.0"
schemaLocation="BDMSLService-1.0.xsd"/>
            <s:import
namespace="http://busdox.org/serviceMetadata/locator/1.0/"
schemaLocation="ServiceMetadataLocatorTypes-1.0.xsd"/>
        </s:schema>
    </wsdl:types>
    <wsdl:message name="prepareChangeCertificateIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart"
element="lrs:PrepareChangeCertificate"/>
    </wsdl:message>
    <wsdl:message name="prepareChangeCertificateOut">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    </wsdl:message>
    <wsdl:message name="createParticipantIdentifierIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart"
element="lrs:SMPAdvancedServiceForParticipantService"/>
    </wsdl:message>
    <wsdl:message name="createParticipantIdentifierOut">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    </wsdl:message>
    <wsdl:message name="isAliveIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart" element="lrs:IsAlive" />
    </wsdl:message>
    <wsdl:message name="isAliveOut"/>

    <wsdl:message name="existsParticipantIdentifierIn">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart" element="lrs:ExistsParticipant" />
    </wsdl:message>
    <wsdl:message name="existsParticipantIdentifierOut">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:part name="messagePart"
element="lrs:ExistsParticipantResponse" />
    </wsdl:message>

    <wsdl:message name="badRequestFault">
        <wsdl:part name="fault" element="bd:BadRequestFault"/>
    </wsdl:message>
    <wsdl:message name="internalErrorFault">
        <wsdl:part name="fault" element="bd:InternalErrorFault"/>
    </wsdl:message>
    <wsdl:message name="notFoundFault">
        <wsdl:part name="fault" element="bd:NotFoundFault"/>
    </wsdl:message>
    <wsdl:message name="unauthorizedFault">
        <wsdl:part name="fault" element="bd:UnauthorizedFault"/>
    </wsdl:message>

   <wsdl:portType name="BDMSLServiceSoap">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
        <wsdl:operation name="PrepareChangeCertificate">
            <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
            <wsdl:input message="tns:prepareChangeCertificateIn"/>
            <wsdl:output message="tns:prepareChangeCertificateOut"/>
            <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="CreateParticipantIdentifier">
            <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
            <wsdl:input message="tns:createParticipantIdentifierIn"/>
            <wsdl:output message="tns:createParticipantIdentifierOut"/>
            <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="IsAlive">
            <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
            <wsdl:input message="tns:isAliveIn"/>
            <wsdl:output message="tns:isAliveOut"/>
            <wsdl:fault message="tns:internalErrorFault"
name="InternalErrorFault"/>
        </wsdl:operation>
       <wsdl:operation name="ExistsParticipantIdentifier">
           <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
           <wsdl:input message="tns:existsParticipantIdentifierIn"/>
           <wsdl:output message="tns:existsParticipantIdentifierOut"/>
           <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="BDMSLServiceSoap" type="tns:BDMSLServiceSoap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="PrepareChangeCertificate">
            <soap:operation
soapAction="ec:services:wsdl:BDMSL:1.0:prepareChangeCertificateIn"
style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap: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="CreateParticipantIdentifier">
            <soap:operation
soapAction="ec:services:wsdl:BDMSL:1.0:createParticipantIdentifierIn"
style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap: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="IsAlive">
            <soap:operation
soapAction="ec:services:wsdl:BDMSL:1.0:isAliveIn" style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ExistsParticipantIdentifier">
            <soap:operation
soapAction="ec:services:wsdl:BDMSL:1.0:existsParticipantIdentifierIn"
style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap: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="BDMSLService">
        <wsdl:port name="BDMSLServicePort" binding="tns:BDMSLServiceSoap">
```

```
            <soap:address location="unknown"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

4.1.1.4. BDMSLAdminService

```
<!--
(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="ec:services:wsdl:BDMSL:admin:1.0" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:lrs="ec:services:wsdl:BDMSL:admin:data:1.0"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="BDMSLAdminService"
    targetNamespace="ec:services:wsdl:BDMSL:admin:1.0"
    xmlns:bd="http://busdox.org/serviceMetadata/locator/1.0/">
 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
 <wsdl:types>
  <s:schema>
   <s:import namespace="ec:services:wsdl:BDMSL:admin:data:1.0" schemaLocation="BDMSLAdminService-1.0.xsd"/>
   <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
     schemaLocation="ServiceMetadataLocatorTypes-1.0.xsd"/>
  </s:schema>
 </wsdl:types>
 <wsdl:message name="generateInconsistencyReportIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GenerateInconsistencyReport"/>
 </wsdl:message>
 <wsdl:message name="generateInconsistencyReportOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GenerateInconsistencyResponse"/>
 </wsdl:message>
 <!-- SubDomain messages-->
 <wsdl:message name="createSubDomainIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:CreateSubDomainRequest"/>
 </wsdl:message>
 <wsdl:message name="createSubDomainOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:CreateSubDomainResponse"/>
 </wsdl:message>
 <wsdl:message name="updateSubDomainIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:UpdateSubDomainRequest"/>
 </wsdl:message>
 <wsdl:message name="updateSubDomainOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:UpdateSubDomainResponse"/>
 </wsdl:message>
 <wsdl:message name="deleteSubDomainIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="DeleteSubDomain" element="lrs:DeleteSubDomainRequest"/>
 </wsdl:message>
 <wsdl:message name="deleteSubDomainOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="DeletedSubDomain" element="lrs:DeleteSubDomainResponse"/>
 </wsdl:message>
 <wsdl:message name="getSubDomainIn">
```

```
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetSubDomainRequest"/>
</wsdl:message>
<wsdl:message name="getSubDomainOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetSubDomainResponse"/>
</wsdl:message>
<!-- subdomain certificate -->
<wsdl:message name="addSubDomainCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddSubDomainCertificateRequest"/>
</wsdl:message>
<wsdl:message name="addSubDomainCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddSubDomainCertificateResponse"/>
</wsdl:message>
<wsdl:message name="updateSubDomainCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:UpdateSubDomainCertificateRequest"/>
</wsdl:message>
<wsdl:message name="updateSubDomainCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:UpdateSubDomainCertificateResponse"/>
</wsdl:message>
<wsdl:message name="listSubDomainCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ListSubDomainCertificateRequest"/>
</wsdl:message>
<wsdl:message name="listSubDomainCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ListSubDomainCertificateResponse"/>
</wsdl:message>
<!-- Manage truststore certificates-->
<wsdl:message name="addTruststoreCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddTruststoreCertificateRequest"/>
</wsdl:message>
<wsdl:message name="addTruststoreCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddTruststoreCertificateResponse"/>
</wsdl:message>
<wsdl:message name="getTruststoreCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetTruststoreCertificateRequest"/>
</wsdl:message>
<wsdl:message name="getTruststoreCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetTruststoreCertificateResponse"/>
</wsdl:message>
<wsdl:message name="deleteTruststoreCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeleteTruststoreCertificateRequest"/>
</wsdl:message>
<wsdl:message name="deleteTruststoreCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeleteTruststoreCertificateResponse"/>
</wsdl:message>
<wsdl:message name="listTruststoreCertificateAliasesIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ListTruststoreCertificateAliasesRequest"/>
</wsdl:message>
<wsdl:message name="listTruststoreCertificateAliasesOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ListTruststoreCertificateAliasesResponse"/>
</wsdl:message>
<!-- DNS records -->
<wsdl:message name="addDNSRecordIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddDNSRecordRequest"/>
 </wsdl:message>
 <wsdl:message name="addDNSRecordOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:AddDNSRecordResponse"/>
 </wsdl:message>
 <wsdl:message name="deleteDNSRecordIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeleteDNSRecordRequest"/>
 </wsdl:message>
 <wsdl:message name="deleteDNSRecordOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeleteDNSRecordResponse"/>
 </wsdl:message>
 <wsdl:message name="setPropertyIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:SetPropertyRequest"/>
 </wsdl:message>
 <wsdl:message name="setPropertyOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:SetPropertyResponse"/>
 </wsdl:message>
 <wsdl:message name="getPropertyIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetPropertyRequest"/>
 </wsdl:message>
 <wsdl:message name="getPropertyOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:GetPropertyResponse"/>
 </wsdl:message>
 <wsdl:message name="deletePropertyIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeletePropertyRequest"/>
 </wsdl:message>
 <wsdl:message name="deletePropertyOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:DeletePropertyResponse"/>
 </wsdl:message>
 <!-- fault messages-->
 <wsdl:message name="badRequestFault">
  <wsdl:part name="fault" element="bd:BadRequestFault"/>
 </wsdl:message>
 <wsdl:message name="internalErrorFault">
  <wsdl:part name="fault" element="bd:InternalErrorFault"/>
 </wsdl:message>
 <wsdl:message name="notFoundFault">
  <wsdl:part name="fault" element="bd:NotFoundFault"/>
 </wsdl:message>
 <wsdl:message name="unauthorizedFault">
  <wsdl:part name="fault" element="bd:UnauthorizedFault"/>
 </wsdl:message>
 <wsdl:message name="changeCertificateIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ChangeCertificate"/>
 </wsdl:message>
 <wsdl:message name="changeCertificateOut">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
 </wsdl:message>
 <wsdl:message name="clearCacheIn">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:part name="messagePart" element="lrs:ClearCache" />
 </wsdl:message>
 <wsdl:message name="clearCacheOut"/>
 <wsdl:portType name="BDMSLAdminServiceSoap">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
  <wsdl:operation name="ChangeCertificate">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
   <wsdl:input message="tns:changeCertificateIn"/>
```

```
    <wsdl:output message="tns:changeCertificateOut"/>
    <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="GenerateInconsistencyReport">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:generateInconsistencyReportIn"/>
    <wsdl:output message="tns:generateInconsistencyReportOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
  </wsdl:operation>
  <wsdl:operation name="CreateSubDomain">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:createSubDomainIn" name="SubDomain"/>
    <wsdl:output message="tns:createSubDomainOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
  </wsdl:operation>
  <wsdl:operation name="UpdateSubDomain">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:updateSubDomainIn"/>
    <wsdl:output message="tns:updateSubDomainOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="DeleteSubDomain">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:deleteSubDomainIn" name="DeleteSubDomain"/>
    <wsdl:output message="tns:deleteSubDomainOut" name="SubDomain"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="GetSubDomain">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:getSubDomainIn" name="GetSubDomain"/>
    <wsdl:output message="tns:getSubDomainOut" name="SubDomain"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="AddSubDomainCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:addSubDomainCertificateIn"/>
    <wsdl:output message="tns:addSubDomainCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="UpdateSubDomainCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:updateSubDomainCertificateIn"/>
    <wsdl:output message="tns:updateSubDomainCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="ListSubDomainCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:listSubDomainCertificateIn"/>
    <wsdl:output message="tns:listSubDomainCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="AddTruststoreCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:addTruststoreCertificateIn"/>
    <wsdl:output message="tns:addTruststoreCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
  </wsdl:operation>
  <wsdl:operation name="GetTruststoreCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:getTruststoreCertificateIn"/>
    <wsdl:output message="tns:getTruststoreCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="DeleteTruststoreCertificate">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:deleteTruststoreCertificateIn"/>
    <wsdl:output message="tns:deleteTruststoreCertificateOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="ListTruststoreCertificateAliases">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:listTruststoreCertificateAliasesIn"/>
    <wsdl:output message="tns:listTruststoreCertificateAliasesOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
  </wsdl:operation>
  <wsdl:operation name="AddDNSRecord">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:addDNSRecordIn"/>
    <wsdl:output message="tns:addDNSRecordOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="DeleteDNSRecord">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:deleteDNSRecordIn"/>
    <wsdl:output message="tns:deleteDNSRecordOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="SetProperty">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
    <wsdl:input message="tns:setPropertyIn"/>
    <wsdl:output message="tns:setPropertyOut"/>
    <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
    <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
    <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
    <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
```

```
  <wsdl:operation name="GetProperty">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
   <wsdl:input message="tns:getPropertyIn"/>
   <wsdl:output message="tns:getPropertyOut"/>
   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
   <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="DeleteProperty">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
   <wsdl:input message="tns:deletePropertyIn"/>
   <wsdl:output message="tns:deletePropertyOut"/>
   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
   <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
  </wsdl:operation>
  <wsdl:operation name="ClearCache">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"/>
   <wsdl:input message="tns:clearCacheIn"/>
   <wsdl:output message="tns:clearCacheOut"/>
   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="BDMSLAdminServiceSoap" type="tns:BDMSLAdminServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="ChangeCertificate">
   <soap:operation soapAction="ec:services:wsdl:BDMSL:1.0:changeCertificateIn" style="document"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
   <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="GenerateInconsistencyReport">
   <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:generateInconsistencyReport" style="document"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
   <wsdl:fault name="InternalErrorFault">
    <soap:fault name="InternalErrorFault" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnauthorizedFault">
    <soap:fault name="UnauthorizedFault" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="BadRequestFault">
    <soap:fault name="BadRequestFault" use="literal"/>
   </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="CreateSubDomain">
   <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:createSubDomain" style="document"/>
   <wsdl:input name="SubDomain">
```

```
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="UpdateSubDomain">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:updateSubDomain" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotFoundFault">
      <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="DeleteSubDomain">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:deleteSubDomain" style="document"/>
    <wsdl:input name="DeleteSubDomain">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="SubDomain">
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotFoundFault">
      <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="GetSubDomain">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:getSubDomain" style="document"/>
    <wsdl:input name="GetSubDomain">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="SubDomain">
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotFoundFault">
      <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="AddSubDomainCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:addSubDomainCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="UpdateSubDomainCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:updateSubDomainCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="ListSubDomainCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:listSubDomainCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="AddTruststoreCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:addTruststoreCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="GetTruststoreCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:getTruststoreCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotFoundFault">
      <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="DeleteTruststoreCertificate">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:deleteTruststoreCertificate" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotFoundFault">
      <soap:fault name="NotFoundFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="ListTruststoreCertificateAliases">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:listTruststoreCertificateAliases" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="AddDNSRecord">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:addDNSRecord" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="DeleteDNSRecord">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:deleteDNSRecord" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="SetProperty">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:setProperty" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <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:fault name="UnauthorizedFault">
      <soap:fault name="UnauthorizedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="GetProperty">
    <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:getProperty" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
```

```
      <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:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="DeleteProperty">
      <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:deleteProperty" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <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:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ClearCache">
      <soap:operation soapAction="ec:services:wsdl:BDMSL:admin:1.0:clearCacheIn" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <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:fault name="UnauthorizedFault">
        <soap:fault name="UnauthorizedFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="BDMSLAdminService">
    <wsdl:port name="BDMSLAdminServicePort" binding="tns:BDMSLAdminServiceSoap">
      <soap:address location="unknown"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

4.1.2. XSD's

4.1.2.1. Identifiers-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/transport/identifiers/1.0/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://busdox.org/transport/identifiers/1.0/" elementFormDefault="qualified" id="Identifiers">
    <xs:annotation>
        <xs:documentation>
  Common identifiers for WSDLs and Schemas
  </xs:documentation>
    </xs:annotation>
    <xs:element name="ParticipantIdentifier" type="ParticipantIdentifierType"/>
    <xs:element name="DocumentIdentifier" type="DocumentIdentifierType"/>
    <xs:element name="ProcessIdentifier" type="ProcessIdentifierType"/>
    <xs:element name="RecipientIdentifier" type="ParticipantIdentifierType"/>
    <xs:element name="SenderIdentifier" type="ParticipantIdentifierType"/>
    <xs:element name="MessageIdentifier" type="MessageIdentifierType"/>
    <xs:element name="ChannelIdentifier" type="ChannelIdentifierType"/>
    <xs:complexType name="ParticipantIdentifierType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="scheme" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="DocumentIdentifierType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="scheme" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="ProcessIdentifierType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="scheme" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="MessageIdentifierType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="ChannelIdentifierType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
</xs:schema>
```

4.1.2.2. ServiceGroupReferenceList.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: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>
```

4.1.2.3. 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="InternalErrorFault" 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>
```

4.1.2.4. 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>
```

4.1.2.5. BDMSLService-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="ec:services:wsdl: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>
```

4.1.2.6. 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>
```

4.2. List of valid issuing agencies for PEPPOL

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/

5. Annexe 1 – Document parts

UC v1.0.xlsx

Drawings - Interface Control Document for

PEPPOL Code Lists 1 2 1-15072016.xls

6. List of Figures

List of Tables

7. Contact Information

eDelivery Support Team

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

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