

Service Metadata Locator (SML)

Status: Published - Version: 1.3.0

Last updated: 2025-02-06

**OpenPeppol AISBL**

Rond-point Schuman 6, box 5

1040 Brussels Belgium

info@peppol.eu

www.peppol.org

Table of Contents

[Revision History 5](#_Toc183626386)

[Contributors 5](#_Toc183626387)

[Organisations 5](#_Toc183626388)

[Persons 6](#_Toc183626389)

[1 Introduction 7](#_Toc183626390)

[1.1 Objective 7](#_Toc183626391)

[1.2 Scope 7](#_Toc183626392)

[1.3 Goals and non-goals 8](#_Toc183626393)

[1.4 Terminology 8](#_Toc183626394)

[1.4.1 Notational conventions 8](#_Toc183626395)

[1.4.2 Normative references 8](#_Toc183626396)

[1.4.3 Non-normative references 9](#_Toc183626397)

[1.5 Namespaces 9](#_Toc183626399)

[2 The Service Discovery Process 10](#_Toc183626400)

[2.1 Discovery flow 10](#_Toc183626401)

[2.1.1 U-NAPTR Resource Records 12](#_Toc183626402)

[2.2 Flows Relating to Service Metadata Publishers 12](#_Toc183626403)

[3 Interfaces and Data Model 16](#_Toc183626404)

[3.1 Service Metadata Locator Service, logical interface 16](#_Toc183626405)

[3.1.1 Format of Participant Identifiers 17](#_Toc183626406)

[3.1.2 ManageBusinessIdentifier interface 17](#_Toc183626407)

[3.1.2.1 Create() 18](#_Toc183626408)

[3.1.2.2 CreateList() 18](#_Toc183626409)

[3.1.2.3 Delete() 19](#_Toc183626410)

[3.1.2.4 DeleteList() 19](#_Toc183626411)

[3.1.2.5 PrepareToMigrate() 20](#_Toc183626412)

[3.1.2.6 Migrate() 21](#_Toc183626413)

[3.1.2.7 List() 21](#_Toc183626414)

[3.1.3 ManageServiceMetadata interface 22](#_Toc183626415)

[3.1.3.1 Create() 22](#_Toc183626416)

[3.1.3.2 Read() 23](#_Toc183626417)

[3.1.3.3 Update() 23](#_Toc183626418)

[3.1.3.4 Delete() 24](#_Toc183626419)

[3.1.4 Fault Descriptions 24](#_Toc183626420)

[3.1.4.1 SMP Not Found Fault 24](#_Toc183626421)

[3.1.4.2 Unauthorized Fault 25](#_Toc183626422)

[3.1.4.3 Bad Request Fault 25](#_Toc183626423)

[3.1.4.4 Internal Error Fault 25](#_Toc183626424)

[3.2 Service Metadata Locator - data model 25](#_Toc183626425)

[3.2.1 ServiceMetadataPublisherService datatype 26](#_Toc183626426)

[3.2.2 ServiceMetadataPublisherServiceForParticipant datatype 26](#_Toc183626427)

[3.2.3 ParticipantIdentifier datatype 27](#_Toc183626428)

[3.2.4 ParticipantIdentifier format 27](#_Toc183626429)

[3.2.5 ParticipantIdentifierPage datatype 27](#_Toc183626430)

[3.2.6 MigrationRecord 28](#_Toc183626431)

[4 Service Bindings 28](#_Toc183626432)

[4.1 Services Provided as Web services - characteristics 28](#_Toc183626433)

[4.2 ManageBusinessIdentifier service - binding 28](#_Toc183626434)

[4.2.1 Transport binding 29](#_Toc183626435)

[4.2.2 Security 29](#_Toc183626436)

[4.3 ManageServiceMetadata service - binding 29](#_Toc183626437)

[4.3.1 Transport binding 29](#_Toc183626438)

[4.3.2 Security 29](#_Toc183626439)

[5 DNS Spoof Mitigation 29](#_Toc183626440)

**Statement of originality**

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

**Statement of copyright**



This deliverable is released under the terms of the Creative Commons Licence accessed through the following link: <https://creativecommons.org/licenses/by-nc-nd/4.0/>

You are free to:

**Share** — copy and redistribute the material in any medium or format.

The licensor cannot revoke these freedoms as long as you follow the license terms*.*

Revision History

| Version | Date | Description of changes | Author |
| --- | --- | --- | --- |
| 1.0.0 | 2010-02-15 | First version (pending EC approval) | Mike Edwards, NITA/IBM |
| 1.0.1 | 2010-10-01 | EC approved | Klaus Vilstrup Pedersen, DIFI |
| 1.2.0 | 2021-05-13 | Updated the references  Improved layout  Linking external XSD and WSDLs in the Appendix  Updated rules for migration key  Changed the service name from “ManageParticipant\*” to “ManageBusiness\*” to reflect the current situation | Philip Helger, OpenPeppol OO |
| 1.3.0 | 2025-02-06 | Removed Appendix A: XML Schema (non-normative)  Removed Appendix B: WSDLs (non-normative)  Updated Peppol references  Updated reference URLs  Added reference to OASIS BDX Location 1.0  Switching from CNAME to U-NAPTR DNS records  Removed the CNAME Wildcard option as this is unused and not relevant with current use of iso6523-actorid-upis | Philip Helger, OpenPeppol OO |

Contributors

Organisations

DIFI (Direktoratet for forvaltning og IKT), Norway, www.difi.no

NITA (IT- og Telestyrelsen), Denmark, www.itst.dk

BRZ (Bundesrechenzentrum), Austria, www.brz.gv.at

Consip, Italy

OpenPeppol

Persons

Bergthór Skúlason, NITA

Carl-Markus Piswanger, BRZ

Christian Uldall Pedersen, NITA/Accenture

Dennis Jensen Søgaard, NITA/Accenture

Gert Sylvest, NITA/Avanade

Hans Guldager Knudsen, NITA/Lenio

Jens Jakob Andersen, NITA

Joakim Recht, NITA/Trifork

Kenneth Bengtsson, NITA/Alfa1lab

Klaus Vilstrup Pedersen, DIFI

Mike Edwards, NITA/IBM (editor)

Mikkel Hippe Brun, NITA

Paul Fremantle, NITA/WSO2

Philip Helger, OpenPeppol Operating Office

Thomas Gundel, NITA/IT Crew

1. Introduction
   1. Objective

This document defines the profiles for the discovery and management interfaces for the Peppol Network Service Metadata Locator (SML) service.

The SML service exposes three interfaces:

* Service Metadata discovery interface  
  This is the lookup interface which enables senders to discover service metadata about specific target participants
* Manage participant identifiers interface  
  This is the interface for Service Metadata Publishers (SMP) for managing the metadata relating to specific participant identifiers that they make available.
* Manage service metadata interface  
  This is the interface for SMP for managing the metadata about their services, e.g. binding, interface profile and key information.

This document describes the physical bindings of the logical interfaces in section 3.1.

* 1. Scope

This specification relates to the Technical Transport Layer of the Peppol Network. It provides transport for electronic documents as specified in the Peppol BIS.



Figure : Peppol Interoperability

* 1. Goals and non-goals

The goal of this document is to describe the interface and transport bindings of the Service Metadata Locator (SML) service. It does not consider its implementation or internal data formats, user management and other procedures related to the operation of this service.

* 1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

* + 1. Notational conventions

Pseudo-schemas are provided for each component, before the description of the component. They use BNF-style conventions for attributes and elements: ? denotes optionality (i.e. zero or one occurrences), \* denotes zero or more occurrences, + one or more occurrences, [ and ] are used to form groups, and | represents choice. Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema. Elements with simple content are conventionally assigned a value which corresponds to the type of their content, as defined in the normative schema. Pseudo schemas do not include extension points for brevity.

<!-- sample pseudo-schema -->

<defined\_element

required\_attribute\_of\_type\_string="xs:string"

optional\_attribute\_of\_type\_int="xs:int"? >

<required\_element />

<optional\_element />?

<one\_or\_more\_of\_these\_elements />+

[ <choice\_1 /> | <choice\_2 /> ]\*

</defined\_element>

* + 1. Normative references

[BDEN-SMP] “Peppol Service Metadata Publishing (SMP) 1.3.0”,  
<https://docs.peppol.eu/edelivery/>

[XML-DSIG] “XML Signature Syntax and Processing (Second Edition)”,  
<https://www.w3.org/TR/xmldsig-core/>

[RFC-2119] “Key words for use in RFCs to Indicate Requirement Levels”,  
<https://datatracker.ietf.org/doc/html/rfc2119>

[RFC3986] “Uniform Resource Identifier (URI): Generic Syntax”,  
<https://datatracker.ietf.org/doc/html/rfc3986>

[RFC4848] “Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)”, <https://datatracker.ietf.org/doc/html/rfc4848>

[PFUOI4] “Peppol Policy for use of Identifiers 4.4.0”,  
<https://docs.peppol.eu/edelivery/>

[BDXL1] “Business Document Metadata Service Location Version 1.0”, <https://docs.oasis-open.org/bdxr/BDX-Location/v1.0/BDX-Location-v1.0.html>

* + 1. Non-normative references

[WSDL-2.0] “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”,  
<https://www.w3.org/TR/wsdl20/>

[WS-I BP] “WS-I Basic Profile Version 1.1”,  
<http://www.ws-i.org/deliverables/basic1.1.html>

[WS-I BSP] “WS-I Basic Security Profile Version 1.0”,  
<http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

[DNS-1034] “Domain Names - Concepts and Facilities”,  
<https://datatracker.ietf.org/doc/html/rfc1034>

[DNS-1035] “Domain Names - Implementation and Specification”,  
<https://datatracker.ietf.org/doc/html/rfc1035>

[SHA256] “US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)”, <https://datatracker.ietf.org/doc/html/rfc6234>

* 1. Namespaces

The following table lists XML namespaces that are used in this document. The choice of any namespace prefix is arbitrary and not semantically significant.

| Prefix | Namespace URI |
| --- | --- |
| ids | http://busdox.org/transport/identifiers/1.0/ |
| lrs | http://busdox.org/serviceMetadata/locator/1.0/ |
| soap | http://schemas.xmlsoap.org/wsdl/soap/ |
| wsdl | http://schemas.xmlsoap.org/wsdl/ |
| xs | http://www.w3.org/2001/XMLSchema |

1. The Service Discovery Process

The interfaces of the Service Metadata Locator (SML) service and the Service Metadata Publisher (SMP) service cover both sender-side lookup and metadata management performed by SMPs. The following interfaces are mandated for these services:

* SML:
  + Discovery interface for senders
  + Management interface for SMPs
* SMP:
  + Discovery interface for senders

This specification only covers the interfaces for the SML.

The SML service specification is based on the use of DNS (Domain Name System) lookups to find the address of the Service Metadata for a given participant ID [DNS-1034] [DNS-1035]. This approach has the advantage that it does not need a single central server to run the Discovery interface, with its associated single point of failure. Instead, the already distributed and highly redundant infrastructure which supports DNS is used. The SML service itself thus plays the role of providing controlled access to the creation and update of entries in the DNS.

* 1. Discovery flow

For a sender, the first step in the Discovery process is to establish the location of the SMP relating to the particular Participant Identifier to which the sender wants to transmit a message. Each participant identifier is registered with one and only one SMP.

1. The sender constructs the domain name for the SMP for a given recipient participant identifier using a standard format, as follows:

<hash over recipientID>.<schemeID>.<SML domain>

1. The sender performs a DNS U-NAPTR record lookup with the domain name created in the previous step and extracts the base URL for the effective SMP query (incl. the URL scheme).
2. The sender constructs the address for the SMP for a given recipient participant identifier using a standard format, as follows:

<smpUrlFromNaptrLookup>/<recipientID>/services/<documentType>

The sender uses this URL in an HTTP GET operation which returns the metadata relating to that recipient and the specific document type (for details, see the SMP specification [BDEN-SMP]). The sender can obtain the information necessary to transmit a message containing that document type to that recipient from the returned metadata. This sequence is shown in Figure 2.

Note that the sender is required to know 2 pieces of information about the recipient - the recipient's participant ID and the ID of the Scheme of the participant ID (i.e. the format or type of the participant ID). This provides for flexibility in the types of participant identifier that can be used in the system. Since in general a participant ID may not have a format that is acceptable in an HTTP URL, the ID is hashed into a string as described in section 3.1.1 Format of Participant Identifiers.

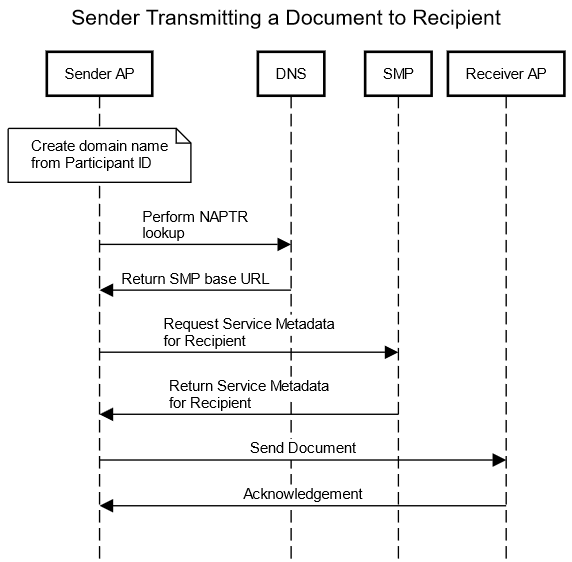


Figure : Sequence Diagram for Sender transmitting Document to Recipient

The underlying design of the Discovery process is based on the use of Domain Name System (DNS) U-NAPTR records (see [BDXL1]) which correspond to the Domain Name in the format given above, namely that there is a U-NAPTR record for the domain name <hash over recipientID>.<schemeID>.<SML domain>. Furthermore, that U-NAPTR record points at the SMP which holds the metadata about that recipient.

* + 1. U-NAPTR Resource Records

The NAPTR service name MUST be Meta:SMP. Other service names MUST NOT be used in relation to this specification. Note that the service field is case-insensitive, according to [RFC4848].

URI values stored in BDXL U-NAPTR records MUST

* use only the “https” URL scheme.
* NOT use username and/or password in the domain authority section
* NOT include query or fragment parts, in addition to the domain authority and path parts

Note that URI scheme and host name are case insensitive. All other URI components MUST be treated as case sensitive (see [RFC3986]).

Note when querying NAPTR records for a Participant, more than one record with different service names may be returned for other purposes than locating the SMP (see [BDXL1]).

* 1. Flows Relating to Service Metadata Publishers

The management of the DNS U-NAPTR records for a given participant identifier is performed through the Management interface of the SML. The management interface is primarily for use by the SMP which controls the service metadata for a given participant identifier. Note that the DNS U-NAPTR records are **not** manipulated directly by the SMP but are manipulated by the SML service following requests made to its Management interface. The basic process steps for the SMP to manipulate the metadata relating to a given participant are shown in Figure 3.

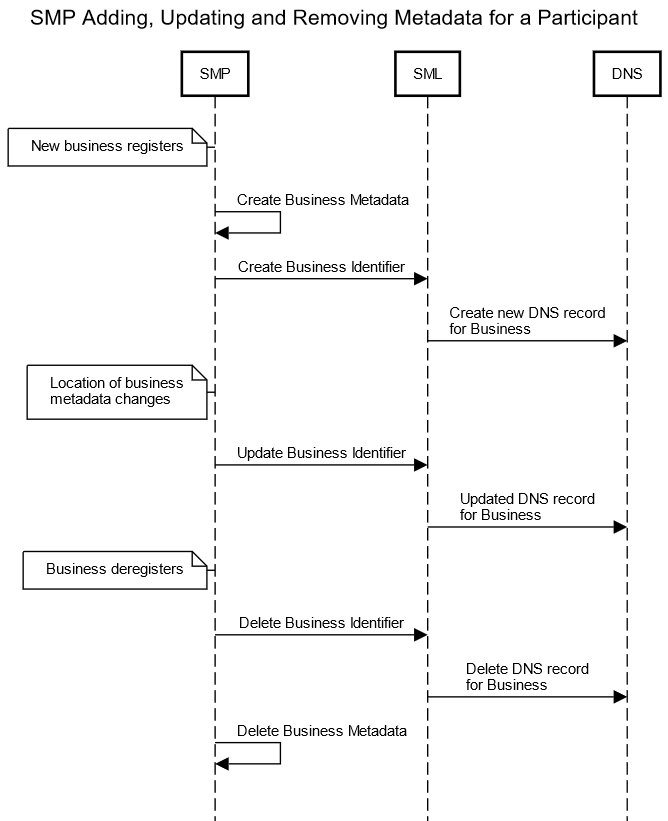


Figure : Sequence Diagram for SMP Adding, Updating and Removing Metadata for a Participant

Each SMP is required to register the address of its server with the SML. Only once this has been done can information relating to specific Participant Identifiers be presented to the SML. The address for the metadata for a given participant is tied to the address of the SMP with which the participant is registered. For this purpose, the SMP uses the ManageServiceMetadata interface with flows as shown in Figure 4.

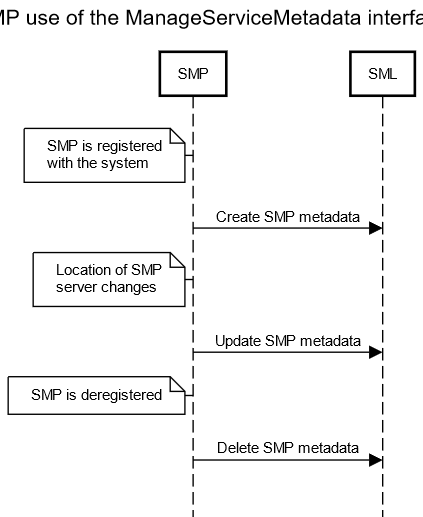


Figure : SMP use of the ManageServiceMetadata

Another set of steps relating to SMPs and the SML relates to the migration of the metadata about a participant from one SMP to another SMP (for example, the participant decides to change suppliers for this function). There are interfaces to the SML to support migrations of this kind, which imply following a sequence of steps along the lines shown in Figure 5.

In this sequence, the original SMP receives a request from a participant to migrate its metadata to a new SMP (a step that is done out-of-band: there are no interfaces defined in these specifications for this). The original SMP generates a “Migration Key” and invokes the PrepareToMigrate operation of the SML and then passes the Migration Key to the new SMP (the key passing is an out-of-band step not defined in these specifications). When the new SMP has created the relevant metadata for the participant, it signals that it is taking over by invoking the Migrate operation of the SML, which then causes the DNS record(s) for that participant ID to be updated to point at the new SMP. Once this switch is complete, the original SMP can remove the metadata which it holds for the participant.

The following rules apply to the Migration Key

* MUST have at least 8 characters and not more than 24 characters
* MUST contain at least 2 lower case characters (a-z)
* MUST contain at least 2 upper case characters (A-Z)
* MUST contain at least 2 digits (0-9)
* MUST contain at least 2 characters from this set: “@” (ASCII code 64), “#” (35), “$” (36), “%” (37), “(“ (40), “)” (41), “[“ (91), “]” (93), “{“ (123), “}” (125), “\*” (42), “^” (94), “-“ (45), “!” (33), “~” (126), “|” (124), “+” (43) and “=” (61)
* MUST NOT contain whitespace characters



Figure : Steps in Migrating Metadata for a Participant from one SMP to a new SMP

1. Interfaces and Data Model

This section outlines the service interfaces and the related data model.

* 1. Service Metadata Locator Service, logical interface

The SML Service interface is divided into 2 logical parts:

* Manage participant identifiers interface  
  This is the interface for SMPs for managing the registered participant identifiers they expose.
* Manage service metadata interface  
  This is the interface for SMPs for managing the metadata about their metadata publishing service, e.g. binding, interface profile and key information.
  + 1. Format of Participant Identifiers

The Peppol Network functions by means of logical addresses for the metadata of services offered by a participant, of the forms

<hash over recipientID>.<schemeID>.<SML domain>

and after DNS resolution in the form

https://<smpBaseUrlFromNaptrLookup>/<recipientID>/services/<documentType>

Peppol is flexible regarding the use of any one of a wide range of schemes for the format of participant identifiers, represented by the schemeID. However, when using this form of HTTP Web address, which is resolved through the DNS system, the format of the recipientID and the schemeID is constrained by the requirements of the DNS system. This means that both the recipientID and the schemeID must be strings which use the ASCII alphanumeric characters only and which have to start with an alphanumeric character.

Peppol allocates schemeIDs to conform to this requirement. However, there is no guarantee that the participant IDs will conform to this requirement for any given scheme (remembering that in many cases the participant ID scheme will be a pre-existing scheme with its own format rules that might violate the requirements of a DNS name). Therefore, a hash of the lowercased participant ID is always used, using the SHA-256 hash algorithm (see [SHA256]). The obtained digest is Base32 encoded and any eventually trailing = characters MUST be removed. See POLICY 7 of the [PFUOI4] for details.

An example participant ID is 0010:5798000000001, for which the SHA-256 hash is XUKHFQABQZIKI3YKVR2FHR4SNFA3PF5VPQ6K4TONV3LMVSY5ARVQ.

* + 1. ManageBusinessIdentifier interface

The ManageBusinessIdentifier interface allows SMPs to manage the information in the SML Service relating to individual participant identifiers for which they hold metadata.

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

The ManageBusinessIdentifier interface has the following operations:

* Create
* CreateList
* Delete
* DeleteList
* PrepareToMigrate
* Migrate
* List

Create()

Creates an entry in the SML 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 SML Service by the SMP which exposes the services for that participant.

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

CreateList()

Creates a set of entries in the SML 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 SML Service by the SMP which exposes the services for that participant.

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

Delete()

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

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

DeleteList()

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

* Input DeleteList: ParticipantIdentifier  
  contains the list of Participant Identifiers for the participants which are removed from the SML Service. The NextPageIdentifier element is absent.
* Fault: notFoundFault  
  returned if one or more participant identifiers or the identifier of the SMP could not be found
* Fault: unauthorizedFault  
  returned if the caller is not authorized to invoke the DeleteList operation
* Fault: badRequestFault  
  returned if the supplied DeleteList does not contain consistent data
* Fault: internalErrorFault  
  returned if the SML service is unable to process the request for any reason

PrepareToMigrate()

Prepares a Participant Identifier for migration to another SMP. This operation is called by the SMP which currently publishes the metadata for the Participant Identifier. The SMP supplies a Migration Code which is used to control the migration process. The Migration Code must be passed (out of band) to the SMP 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 SMP which currently publishes the metadata for the specified Participant Identifier.

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

Migrate()

Migrates a Participant Identifier already held by the SML Service to target a new SMP. This operation is called by the SMP which is taking over the publishing for the Participant Identifier. The operation requires the new SMP to provide a migration code which was originally obtained from the old SMP.

The PrepareToMigrate() operation MUST have been previously invoked for the supplied Participant Identifier, using the same MigrationCode, otherwise the Migrate() operation fails.

Following the successful invocation of this operation, the lookup of the metadata for the service endpoints relating to a particular Participant Identifier will resolve (via DNS) to the new SMP.

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

List()

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

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

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

* + 1. ManageServiceMetadata interface

The ManageServiceMetadata interface allows SMPs to manage the metadata held in the SML Service about their SMP 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 SMP associated with the service metadata which is managed via this interface.

The ManageServiceMetadata interface has the following operations:

* Create
* Read
* Update
* Delete

Create()

Establishes a SMP metadata record, containing the metadata about the SMP, as outlined in the ServiceMetadataPublisherService data type.

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

Read()

Retrieves the SMP record for the SMP.

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

Update()

Updates the SMP record for the SMP

* Input UpdateServiceMetadataPublisheServicer: ServiceMetadataPublisherService  
  contains the service metadata for the SMP, which includes the logical and physical addresses for the SMP (Domain name and IP address)
* Fault: notFoundFault  
  returned if the identifier of the SMP could not be found
* Fault: unauthorizedFault  
  returned if the caller is not authorized to invoke the Update operation
* Fault: badRequestFault  
  returned if the supplied UpdateServiceMetadataPublisheServicer does not contain consistent data
* Fault: internalErrorFault  
  returned if the SML service is unable to process the request for any reason

Delete()

Deletes the SMP record for the SMP

* Input DeleteServiceMetadataPublisherService: ServiceMetadataPublisherID  
  the unique ID of the SMP to delete
* Fault: notFoundFault  
  returned if the identifier of the SMP could not be found
* Fault: unauthorizedFault  
  returned if the caller is not authorized to invoke the Delete operation
* Fault: badRequestFault  
  returned if the supplied DeleteServiceMetadataPublisherService does not contain consistent data
* Fault: internalErrorFault  
  returned if the SML service is unable to process the request for any reason
  + 1. Fault Descriptions

****SMP Not Found Fault****

|  |  |
| --- | --- |
| [action] | http://busdox.org/2010/02/locator/fault |
| Code | Sender |
| Subcode | notFoundFault |
| Reason | The identifier of the SMP supplied could not be found by the SML |
| Detail | As detailed by the SML |

Unauthorized Fault

|  |  |
| --- | --- |
| [action] | http://busdox.org/2010/02/locator/fault |
| Code | Sender |
| Subcode | unauthorizedFault |
| Reason | The caller is not authorized to perform the operation requested |
| Detail | As detailed by the SML |

Bad Request Fault

|  |  |
| --- | --- |
| [action] | http://busdox.org/2010/02/locator/fault |
| Code | Sender |
| Subcode | badRequestFault |
| Reason | The operation request was incorrect in some way |
| Detail | As detailed by the SML |

Internal Error Fault

|  |  |
| --- | --- |
| [action] | http://busdox.org/2010/02/locator/fault |
| Code | Sender |
| Subcode | internalErrorFault |
| Reason | The SML encountered an error while processing the request |
| Detail | As detailed by the SML |

* 1. Service Metadata Locator - data model

The data model for the SML involves the following data types:

* ServiceMetadataPublisher
* RecipientParticipantIdentifier
* ParticipantIdentifierPage
* MigrationRecord

Each of these data types is described in detail in the following subsections.

* + 1. ServiceMetadataPublisherService datatype

Represents an SMP Service.

<ServiceMetadataPublisherService>

<PublisherEndpoint>

<EndpointAddress/>

</PublisherEndpoint>

<ServiceMetadataPublisherID/>

</ServiceMetadataPublisherService>

ServiceMetadataPublisherService has the following sub-elements:

* PublisherEndpoint (1..1) : PublisherEndpointType  
  the technical endpoint address of the SMP, which can be used to query information about particular participant identifiers. ServiceEndpointList is a type defined in the ServiceMetadataPublishingTypes Schema. The PublisherEndpoint element may be a domain name or an IP address of the SMP.
* ServiceMetadataPublisherID (1..1) : xs:string  
  holds the Unique Identifier of the SMP. When creating a ServiceMetadataPublisherService record, it is assumed that the publisher ID has been obtained out of band.
  + 1. ServiceMetadataPublisherServiceForParticipant datatype

Represents an SMP Service containing information about a particular Participant Identifier.

<ServiceMetadataPublisherServiceForParticipant>

<ServiceMetadataPublisherID/>

<ids:ParticipantIdentifier/>

</ServiceMetadataPublisherServiceForParticipant>

ServiceMetadataPublisherService has the following subelements:

* ServiceMetadataPublisherID (1..1) : xs:string  
  holds the Unique Identifier of the SMP.
* ParticipantIdentifier (1..1) : ids:ParticipantIdentifierType  
  the Participant Identifier which has its services registered in the SMP. See the “ParticipantIdentifier” section on the format.
  + 1. ParticipantIdentifier datatype

Represents a Participant Identifier which has its service metadata held by a specific SMP.

<ids:ParticipantIdentifier scheme=”xs:string”>

xs:string

</ids:ParticipantIdentifier>

ParticipantIdentifier has the following sub elements:

* ParticipantIdentifier (1..1): xs:string  
  the participant identifier
* @scheme (1..1): xs:string  
  the format scheme of the participant identifier
  + 1. ParticipantIdentifier format

For a description of the ParticipantIdentifier format, see the “Peppol Policy for use of Identifier” document [PFUOI4].

* + 1. ParticipantIdentifierPage datatype

Represents a page of ParticipantIdentifiers for which data is held by the SML Service.

<ParticipantIdentifierPage>

<ServiceMetadataPublisherID/>

<ParticipantIdentifier/>\*

<NextPageIdentifier/>?

</ParticipantIdentifierPage>

* ServiceMetadataPublisherID (1..1) : xs:string  
  holds the Unique Identifier of the SMP
* ids:ParticipantIdentifier (1..1): xs:string  
  the participant identifier
* NextPageIdentifier (0..1): xs:string  
  an element containing a string identifying the next page of ParticipantIdentifiers:

<NextPageIdentifier>

[ Identifier for\_Next\_Page ]

</NextPageIdentifier>

If no <NextPageIdentifier/> element is present, it implies that there are no further pages.

* + 1. MigrationRecord

The MigrationRecord represents the data required to control the process of migrating a ParticipantIdentifier from the control of one SMP to another SMP.

<MigrationRecord>

<ServiceMetadataPublisherID/>

<ParticipantIdentifier/>\*

<MigrationKey/>?

</MigrationRecord>

MigrationRecord has the following sub elements:

* ServiceMetadataPublisherID (1..1) : xs:string  
  holds the Unique Identifier of the SMP.
* ParticipantIdentifier (1..1) : ids:ParticipantIdentifierType  
  the participant identifier
* MigrationKey (1..1) : xs:string  
  a string which is a unique key controlling the migration of the metadata for a given ParticipantIdentifier from one SMP to another. The MigrationKey string is a string of characters and numbers only, with a maximum length of 24 characters.

1. Service Bindings

This section describes the Bindings of the services provided by the SML to specific transports.

* 1. Services Provided as Web services - characteristics

Some of the services described by this specification are provided through Web service bindings.

Where services are provided through Web services bindings, those bindings MUST conform to the relevant WS-I Profiles, in particular WS-I Basic Profile 1.1 and WS-I Basic Security Profile 1.0.

* 1. ManageBusinessIdentifier service - binding

The ManageBusinessIdentifier service is provided in the form of a SOAP-based Web service.

* + 1. Transport binding

The ManageBusinessIdentifier interface is bound to an HTTP SOAP 1.1 transport.

The WSDL files are published together with this specification.

* + 1. Security

The service is secured at the transport level with a two-way TLS connection. The requestor must authenticate using a client certificate (mTLS) issued for use in the infrastructure by a trusted third-party. In the Peppol Network, a Peppol SMP certificate will be issued to the participants when they have signed the Service Provider agreements and live up to the stated requirements. The server must reject TLS clients that do not authenticate with a certificate issued under the Peppol root CA.

* 1. ManageServiceMetadata service - binding

SMPs use this interface to create or update metadata such as the endpoint address for retrieval of metadata about specific participant services.

The ManageServiceMetadata service is provided in the form of a SOAP-based Web service.

* + 1. Transport binding

The ManageServiceMetadata interface is bound to an HTTP SOAP 1.1 transport.

The WSDL files are published together with this specification.

* + 1. Security

The service is secured at the transport level with a two-way TLS connection. The requestor must authenticate using a client certificate issued for use in the infrastructure by a trusted third-party.

1. DNS Spoof Mitigation

The regular lookup of the address of the SMP for a given participant ID is performed using a standard DNS lookup. There is a potential vulnerability of this process if there exists at least one "rogue" certificate (e.g. stolen or otherwise illegally obtained).

In this vulnerability, someone possessing such a rogue certificate could perform a DNS poisoning or a man-in-the-middle attack to fool senders of documents into making a lookup for a specific identifier in a malicious SMP (that uses the rogue certificate), effectively routing all messages intended for one or more recipients to a malicious access point. This attack could be used for disrupting message flow for those recipients, or for gaining access to confidential information in these messages (if the messages were not separately encrypted).

One mitigation for this kind of attack on the DNS lookup process is to use DNSSEC rather than plain DNS. DNSSEC allow the authenticity of the DNS resolutions to be checked by means of a trust anchor in the domain chain. Therefore, it is recommended that an SML instance uses the DNSSEC infrastructure.