Specification

Acronym:	PEPPOL
Grant Agreement number:	224974
Title:	Pan-European Public Procurement Online

PEPPOL Transport Infrastructure Service Metadata Locator (SML)

Version: 1.01

Authors:

Gert Sylvest (NITA/Avanade)
Jens Jakob Andersen (NITA)
Klaus Vilstrup Pedersen (DIFI)
Mikkel Hippe Brun (NITA)
Mike Edwards (NITA/IBM)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Version	Date	Author	Organisation	Description
1.0	20100215	Mike Edwards	NITA/IBM	First version (pending EC approval)
1.01	20101001	Klaus Vilstrup Pedersen	DIFI	EC Approved

Contributors

Organisations

DIFI (Direktoratet for forvaltning og IKT)[1], Norway, www.difi.no
NITA (IT- og Telestyrelsen)[2], Denmark, www.itst.dk
BRZ (Bundesrechenzentrum), Austria
Consip, Italy

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, BRZ
Thomas Gundel, NITA/IT Crew

[1] English: Agency for Public Management and eGovernment

[2] English: National IT- and Telecom Agency

Table of Content

1 Introduction

1.1 Objective

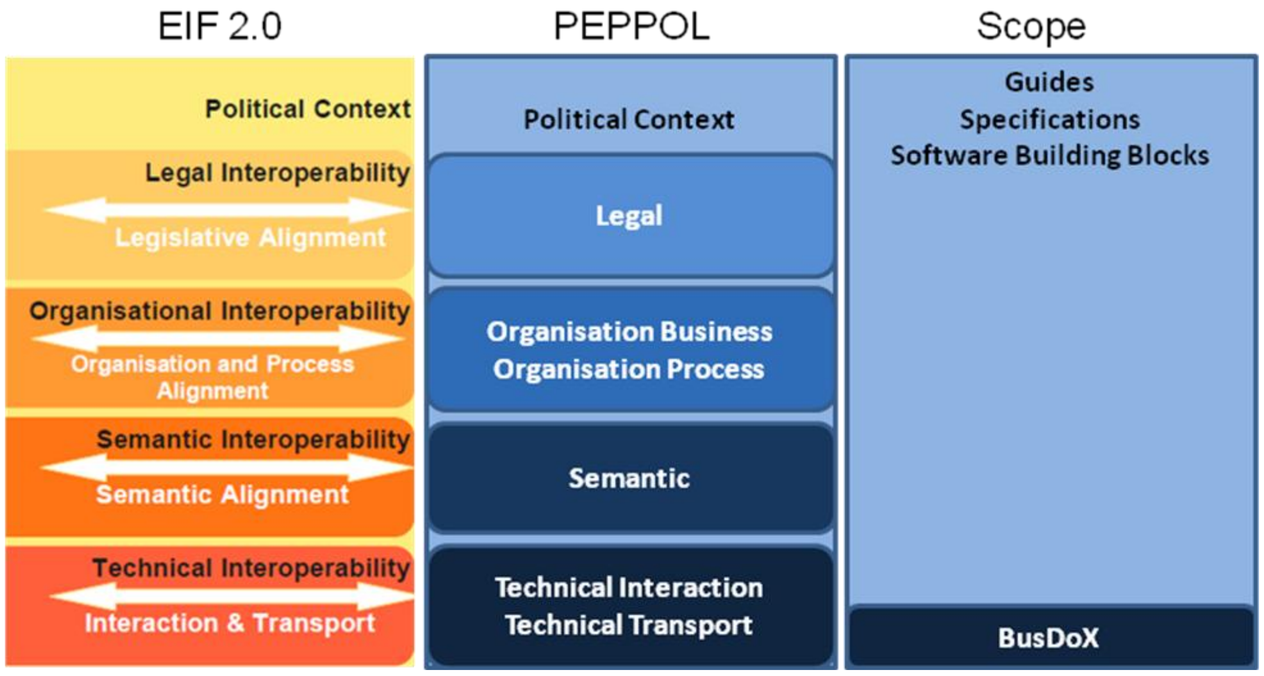
This document defines the profiles for the discovery and management interfaces for the Business Document Exchange Network (BUSDOX) Service Metadata Locator service.
The Service Metadata Locator service exposes three interfaces:

- *Service Metadata discovery 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 for managing the metadata relating to specific participant identifiers that they make available.
- *Manage service metadata interface*. This is the interface for Service Metadata publishers 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 03.1.

1.2 Scope

This specification relates to the Technical Transport Layer i.e. BusDox specifications. The BusDox specifications can be used in many interoperability settings. In the PEPPOL context, it provides transport for procurement documents as specified in the PEPPOL Profiles.



1.3 Goals and non-goals

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

1.4 Terminology

For a definition of terms, see the Common Definitions document [BDEN-CDEF].

Notational conventions
For a description of notational conventions, see the Common Definitions document [BDEN-CDEF].
Normative references
[BDEN-START] Secure Trusted Asynchronous Reliable Transport (START), STARTProfile.pdf
[BDEN-SMP] Service Metadata Publishing, ServiceMetadataPublishing.pdf
[BDEN-CDEF] Business Document Exchange Network - Common Definitions, CommonDefinitions.pdf
[XML-DSIG] XML Signature Syntax and Processing (Second Edition) http://www.w3.org/TR/xmldsig-core/
[RFC-2119] "Key words for use in RFCs to Indicate Requirement Levels", http://www.ietf.org/rfc/rfc2119.txt
[RFC3986] "Uniform Resource Identifier (URI): Generic Syntax", http://tools.ietf.org/html/rfc3986

Non-normative references
[WSDL-2.0] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", http://www.w3.org/TR/wsdl20/
[WS-I BP] "WS-I Basic Profile Version 1.1" http://www.ws-i.org/Profiles/BasicProfile-1.1.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" http://tools.ietf.org/html/rfc1034
[DNS-1035] "Domain Names - Implementation and Specification" http://tools.ietf.org/html/rfc1035
[MD5] The MD5 Message-Digest Algorithm http://tools.ietf.org/html/rfc1321

1.5 Namespaces

For a list of namespaces and prefixes used in this document, see the Common Definitions document [BDEN-CDEF].

2 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. BUSDOX mandates the following interfaces for these services:

- Service Metadata Locator:
  - Discovery interface for senders
  - Management interface for SMPs
- Service Metadata Publishers:
  - Discovery interface for senders

This specification only covers the interfaces for the Service Metadata Locator.
The Service Metadata Locator 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.

2.1 Discovery flow

For a sender, the first step in the Discovery process is to establish the location of the Service Metadata 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 Service Metadata Publisher. The sender constructs the address for the service metadata for a given recipient participant identifier using a standard format, as follows:
http://<hash over recipientID>.<schemeID>.<SML domain>/<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 Service Metadata Publishing 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 1.
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 4.1.1 Format of Participant Identifiers.

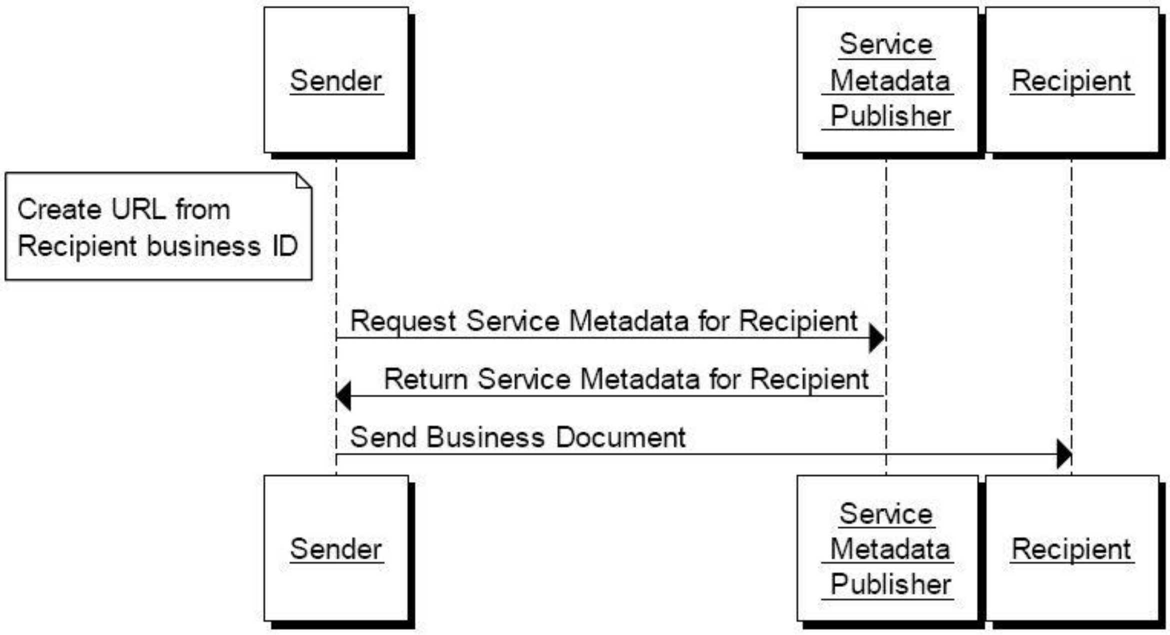

Sender
Service Metadata Publisher
Recipient
Create URL from Recipient business ID
Request Service Metadata for Recipient
Return Service Metadata for Recipient
Send Business Document
Sender
Service Metadata Publisher
Recipient

Figure 1: 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) CNAME records which correspond to the Domain Name in the format given above, namely that there is a CNAME record for the domain name "<hash over recipientID>.<schemeID>.<SML domain>". Furthermore, that CNAME record points at the Service Metadata Publisher which holds the metadata about that recipient. This means that an address lookup for the domain name by the sender naturally resolves to the Service Metadata Publisher holding the metadata. The resolution of Web URLs in this way is a fundamental part of the World Wide Web and so it is based on standard technology that it available to all users.

2.2 Flows Relating to Service Metadata Publishers

The management of the DNS CNAME records for a given participant identifier is performed through the Management interface of the Service Metadata Locator. The management interface is primarily for use by the Service Metadata Publisher which controls the service metadata for a given participant identifier. Note that the DNS CNAME records are ***not*** manipulated directly by the Service Metadata Publisher, but are manipulated by the Service Metadata Locator 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 2.

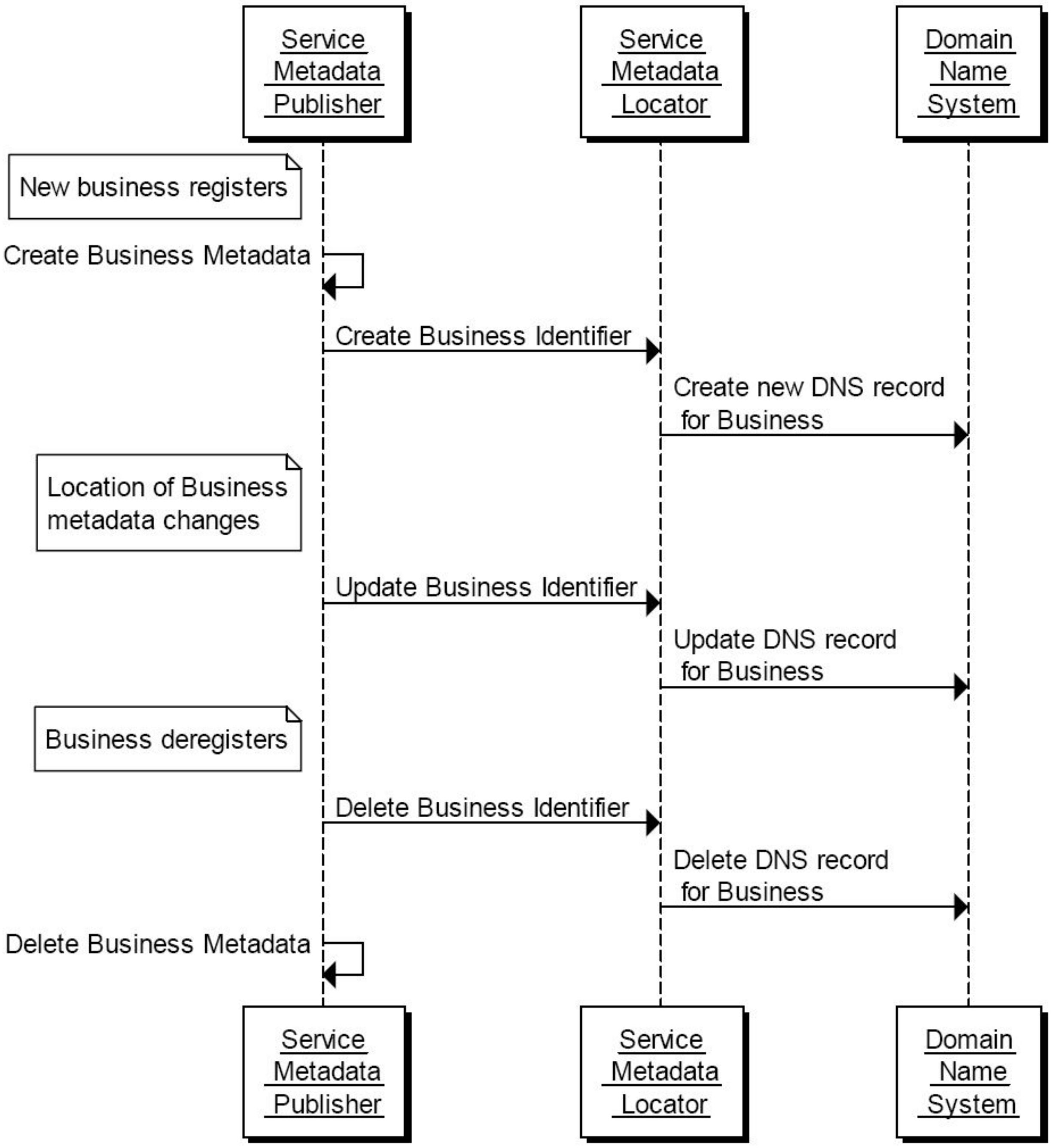

Service Metadata Publisher
Service Metadata Locator
Domain Name System
New business registers
Create Business Metadata
Create Business Identifier
Create new DNS record for Business
Location of Business metadata changes
Update Business Identifier
Update DNS record for Business
Business deregisters
Delete Business Identifier
Delete DNS record for Business
Delete Business Metadata
Service Metadata Publisher
Service Metadata Locator
Domain Name System

Figure 2: Sequence Diagram for Service Metadata Publisher Adding, Updating and Removing Metadata for a Participant

Each Service Metadata Publisher is required to register the address of its server with the Service Metadata Locator. 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 3.

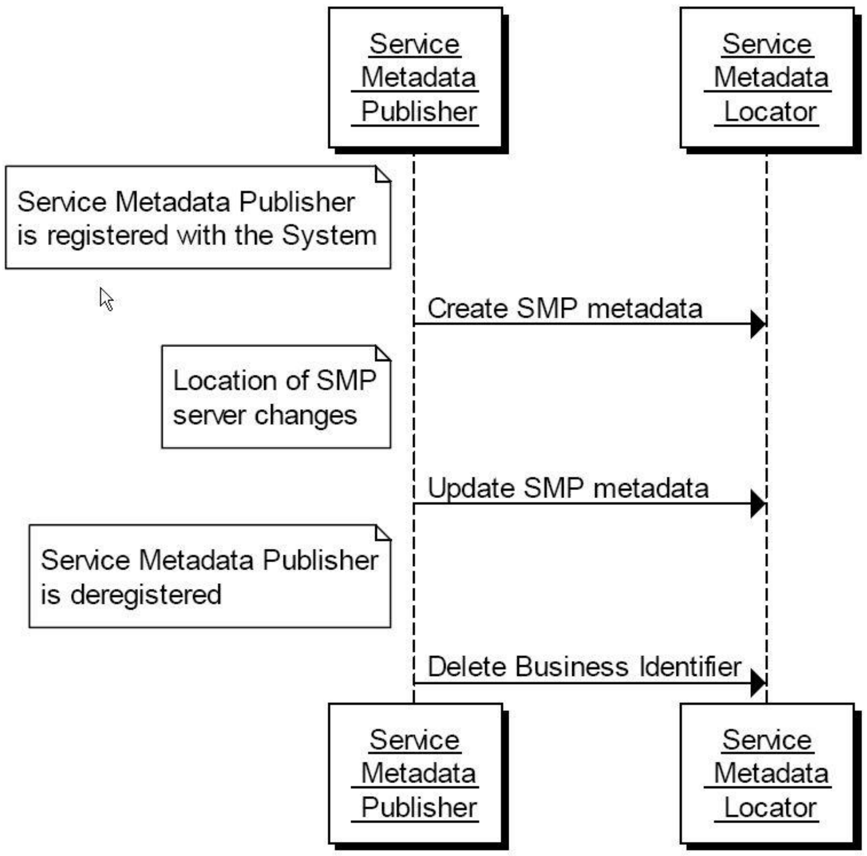

Service Metadata Publisher
Service Metadata Locator
Service Metadata Publisher is registered with the System
Create SMP metadata
Location of SMP server changes
Update SMP metadata
Service Metadata Publisher is deregistered
Delete Business Identifier
Service Metadata Publisher
Service Metadata Locator

Figure 3: Service Metadata Publisher 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 4.

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 SMP generates a Migration Key which is a unique string containing characters and numbers only, with a maximum length of 24 characters. The original SMP 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.

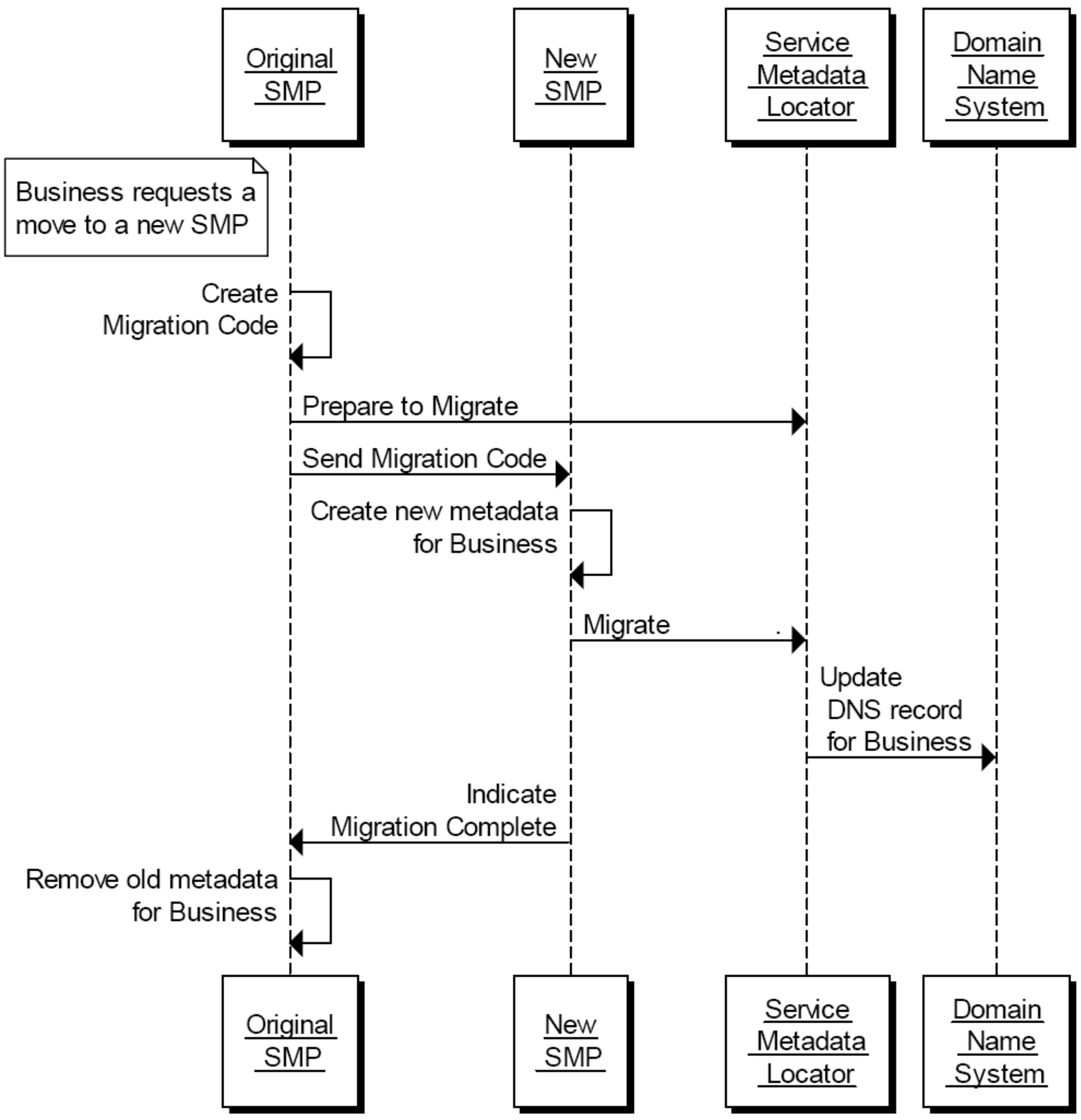


Figure 4: Steps in Migrating Metadata for a Participant from one SMP to a new SMP

3 Interfaces and Data Model

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

3.1 Service Metadata Locator Service, logical interface

The Service Metadata Locator Service interface is divided into 2 logical parts:

- *Manage participant identifiers interface*. This is the interface for Service Metadata Publishers for managing the registered participant identifiers they expose.
- *Manage service metadata interface*. This is the interface for Service Metadata Publishers for managing the metadata about their metadata publishing service, e.g. binding, interface profile and key information.

Format of Participant Identifiers

BUSDOX functions by means of logical addresses for the metadata of services offered by a participant, of the form:
http://<hash over recipientID>.<schemeID>.<SML domain>/<recipientID>/services/<documentType>
BUSDOX is flexible with regard to 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 alphabetic character.
BUSDOX 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 participant ID is always used, using the MD5 hash algorithm [MD5], and prefixed by "B-".
An example participant ID is "0010:5798000000001" (see the Common Definitions document for more details [BDEN-CDEF]), for which the MD5 hash is "e49b223851f6e97cbfce4f72c3402aac".

ManageParticipantIdentifier interface

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 are managed via this interface.
It is possible for a given Service Metadata Publisher to provide the metadata for all participant identifiers belonging to a particular participant identifier scheme. If this is the case, then it corresponds to the concept of a "wildcard" CNAME record in the DNS, along the lines:

*.<schemeID>.<SML domain> CNAME <SMP domain>

<SMP domain> may either be the domain name associated with the SMP, or an alias for it.
This implies that all participant identifiers for that schemeID will have addresses that resolve to the single address of that one SMP - and that as result only one SMP can handle the metadata for all participant identifiers of that scheme. Wildcard records are indicated through the use of "*" as the participant identifier in the operations of the ManageParticipantIdentifier interface.
The ManageParticipantIdentifier interface has the following operations:

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

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.

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

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

- ***Fault: notFoundFault*** **-** returned if one or more participant identifiers or the identifier of the SMP could not be found
- ***Fault: unauthorizedFault*** **-** returned if the caller is not authorized to invoke the DeleteList operation
- ***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 a new Service Metadata Publisher. This operation is called by the Service Metadata Publisher which currently publishes the metadata for the Participant Identifier. The Service Metadata Publisher supplies a Migration Code which is used to control the migration process. The Migration Code must be passed (out of band) to the Service Metadata Publisher which is taking over the publishing of the metadata for the Participant Identifier and which MUST be used on the invocation of the Migrate() operation.
This operation can only be invoked by the Service Metadata Publisher which currently publishes the metadata for the specified Participant Identifier.

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

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.

- ***Input CompleteMigrationRecord: MigrationRecordType*** - contains the Migration Key and the Participant Identifier which is to be migrated from one Service Metadata Publisher to another.

- ***Fault: notFoundFault*** - returned if the migration key or the identifier of the SMP could not be found

- ***Fault: unauthorizedFault*** - returned if the caller is not authorized to invoke the Migrate operation

- ***Fault: badRequestFault*** - returned if the supplied CompleteMigrationRecord does not contain consistent data

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

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.

- ***Input Page: PageRequest*** - contains a PageRequest containing the ServiceMetadataPublisherID of the SMP and (if required) an identifier representing the next page of data to retrieve. If the NextPageIdentifier is absent, the first page is returned.

- ***Output: ParticipantIdentifierPage*** - a page of Participant Identifier entries associated with the Service Metadata Publisher, also containing a <Page/> element containing the identifier that represents the next page, if any.

- ***Fault: notFoundFault*** - returned if the next page or the identifier of the SMP could not be found

- ***Fault: unauthorizedFault*** - returned if the caller is not authorized to invoke the List operation

- ***Fault: badRequestFault*** - returned if the supplied NextPage does not contain consistent data

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

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

ManageServiceMetadata interface

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**

Create()

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

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

Read()

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

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

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

Update()
Updates the Service Metadata Publisher record for the service metadata publisher

- ***Input UpdateServiceMetadataPublisheServicer: ServiceMetadataPublisherService*** **-** contains the service metadata for the service metadata publisher, which includes the logical and physical addresses for the SMP (Domain name and IP address)

- ***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 Service Metadata Publisher record for the service metadata publisher

- ***Input DeleteServiceMetadataPublisherService: ServiceMetadataPublisherID*** **-** the unique ID of the Service Metadata Publisher to delete

- ***Fault: notFoundFault*** **-** returned if the identifier of the SMP could not be found

- ***Fault: unauthorizedFault*** **-** returned if the caller is not authorized to invoke the Delete operation

- ***Fault: badRequestFault*** **-** returned if the supplied DeleteServiceMetadataPublisherService does not contain consistent data

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

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

3.2 Service Metadata Locator - data model

The data model for the Service Metadata Locator involves the following data types:

- ServiceMetadataPublisher
- RecipientParticipantIdentifier
- ParticipantIdentifierPage
- MigrationRecord

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

ServiceMetadataPublisherService datatype

Represents a Metadata Publisher Service.

```
<ServiceMetadataPublisherService>
    <PublisherEndpoint>
        <EndpointAddress/>
    </PublisherEndpoint>
    <ServiceMetadataPublisherID/>
</ServiceMetadataPublisherService>
```

ServiceMetadataPublisherService has the following subelements:

- ***PublisherEndpoint (1..1) : PublisherEndpointType*** - the technical endpoint address of the Service Metadata Publisher, 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, or a wildcard expression based on the domain name.

- ***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.

ServiceMetadataPublisherServiceForParticipant datatype
Represents a Metadata Publisher Service containing information about a particular Participant Identifier.

```
<ServiceMetadataPublisherServiceForParticipant>
    <ServiceMetadataPublisherID/>
    <ids:ParticipantIdentifier/>
</ServiceMetadataPublisherServiceForParticipant>
```

ServiceMetadataPublisherService has the following subelements:

- ***ServiceMetadataPublisherID (1..1) : String*** - Holds the Unique Identifier of the SMP.
- ***ParticipantIdentifier (1..1) : ids:ParticipantIdentifierType*** - the Participant Identifier which has its services registered in the Service Metadata Publisher. See the “ParticipantIdentifier” section on the format.

ParticipantIdentifier datatype
Represents a Participant Identifier which has its service metadata held by a specific Service Metadata Publisher.

```
<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

ParticipantIdentifier format
For a description of the ParticipantIdentifier format, see the BUSDOX Common Definitions document [BDEN-CDEF].

ParticipantIdentifierPage datatype
Represents a page of ParticipantIdentifiers for which data is held by the Service Metadata Locator 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.

MigrationRecord
The MigrationRecord represents the data required to control the process of migrating a ParticipantIdentifier from the control of one Service Metadata Publisher to a different Service Metadata Publisher.

```
<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 Service Metadata Publisher to another. The MigrationKey string is a string of characters and numbers only, with a maximum length of 24 characters.

4 Service Bindings

This section describes the Bindings of the services provided by the Service Metadata Locator to specific transports.

4.1 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.

4.2 ManageParticipantIdentifier service - binding

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

Transport binding
The ‘manage participant identifier’ interface is bound to an HTTP SOAP 1.1 transport.
See a WSDL for this in “Appendix B: WSDLs”.

Security
The service is secured at the transport level with a two-way SSL / TLS connection. The requestor must authenticate using a client certificate issued for use in the infrastructure by a trusted third-party. For example, in the PEPPOL infrastructure (an instance of the BUSDOX infrastructure), a PEPPOL certificate will be issued to the participants when they have signed peering agreements and live up to the stated requirements. The server must reject SSL clients that do not authenticate with a certificate issued under the PEPPOL root.

4.3 ManageServiceMetadata service - binding

Service Metadata Publishers 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.

Transport binding
The ‘ManageServiceMetadata’ interface is bound to an HTTP SOAP 1.1 transport.
See a WSDL for this in “Appendix B: WSDLs”.

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

5 DNS Spoof Mitigation

The regular lookup of the address of the SMP for a given participant ID is performed using a standard DNS lookup. There is a potential vulnerability of this process if there exists at least one "rogue" certificate (e.g. stolen or otherwise illegally obtained).
In this vulnerability, someone possessing such a rogue certificate could perform a DNS poisoning or a man-in-the-middle attack to fool senders of documents into making a lookup for a specific identifier in a malicious SMP (that uses the rogue certificate), effectively routing all messages intended for one or more recipients to a malicious access point. This attack could be used for

disrupting message flow for those recipients, or for gaining access to confidential information in these messages (if the messages were not separately encrypted).
One mitigation for this kind of attack on the DNS lookup process is to use DNSSEC rather than plain DNS. DNSSEC allow the authenticity of the DNS resolutions to be checked by means of a trust anchor in the domain chain. Therefore, it is recommended that an SML instance uses the DNSSEC infrastructure.

6 Appendix A: Schema

This section defines the XML Schema types used in the 3 interfaces.

6.1 ServiceMetadataLocatorTypes.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ServiceMetadataPublisherService"
  targetNamespace="http://busdox.org/serviceMetadata/locator/1.0/"
  elementFormDefault="qualified"
  xmlns="http://busdox.org/serviceMetadata/locator/1.0/"
  xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
  xmlns:mstns="http://tempuri.org/ServiceMetadataPublisherService.xsd"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
    namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"/>
  <xs:import schemaLocation="ws-addr.xsd"
    namespace="http://www.w3.org/2005/08/addressing" />
  <xs:import schemaLocation="Identifiers-0.9.xsd"
    namespace="http://busdox.org/transport/identifiers/1.0/"/>

  <xs:element name="ServiceMetadataPublisherID" type="xs:string" />
  <xs:element name="CreateServiceMetadataPublisherService"
    type="ServiceMetadataPublisherServiceType"/>
  <xs:element name="ReadServiceMetadataPublisherService"
    type="ServiceMetadataPublisherIdentifierType"/>
  <xs:element name="UpdateServiceMetadataPublisherService"
    type="ServiceMetadataPublisherServiceType"/>
  <xs:element name="DeleteServiceMetadataPublisherService"
    ref="ServiceMetadataPublisherID"/>
  <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="EndpointAddress" type="xs:anyURI"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ServiceMetadataPublisherServiceForParticipantType">
    <xs:sequence>
      <xs:element ref="ServiceMetadataPublisherID"/>
      <xs:element ref="ids:ParticipantIdentifier" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ServiceMetadataPublisherIdentifierType">
    <xs:sequence>
      <xs:element ref="ServiceMetadataPublisherID"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="CreateParticipantIdentifier"
    type="ServiceMetadataPublisherServiceForParticipantType"/>
  <xs:element name="DeleteParticipantIdentifier"
    type="ServiceMetadataPublisherServiceForParticipantType"/>
  <xs:element name="ServiceMetadataPublisherService"
    type="ServiceMetadataPublisherServiceType">
  </xs:element>
  <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="ServiceMetadataPublisherID"/>
      <xs:element ref="ids:ParticipantIdentifier" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="PageID" 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>
```

7 Appendix B: WSDLs

This section defines the WSDLs for the services offered as Web services.

7.1 ManageParticipantIdentifierService.wsdl

```
<wsdl:definitions
  xmlns:tns="http://busdox.org/serviceMetadata/
    ManageParticipantIdentifierService/1.0/"
  xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  name="ManageParticipantIdentifierService"
  targetNamespace=
    "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/
        1.0/Schema/">
      <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
        schemaLocation="ServiceMetadataLocatorTypes.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="ManageParticipantIdentifierServiceSoap">
  <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="ManageParticipantIdentifierServiceSoap"
    type="tns:ManageParticipantIdentifierServiceSoap">
    <soap11:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="Create">
      <soap11:operation soapAction=

"http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/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="CreateList">
```

```
      <soap11:operation soapAction=

"http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/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/ManageParticipantIdentifierService/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:definitions>
```

7.2 ManageServiceMetadataService.wsdl

```
<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:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  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.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:DeleteServiceMetadataPublisherService" />
  </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:definitions>
```