



# Service Metadata Locator Profile

---

*Interfaces and bindings for the Service Metadata Locator  
Service*



*Version 1.0.0*

WP8 2009-12-21



## Contents

1	Document information .....	4
1.1	Document history .....	4
1.2	Editors .....	5
1.3	Contributors (alphabetically) .....	5
2	Introduction .....	6
2.1	Goals and non-goals.....	6
2.2	Terminology .....	6
2.2.1	Notational conventions.....	6
2.2.2	Normative references .....	6
2.2.3	Non-normative references.....	6
2.3	Namespaces .....	7
3	The Service Discovery Process .....	8
3.1	Discovery flow.....	8
3.2	Flows Relating to Service Metadata Publishers.....	9
4	Interfaces and Data Model .....	13
4.1	Service Metadata Locator Service, logical interface.....	13
4.1.1	Format of Participant Identifiers.....	13
4.1.2	ManageParticipantIdentifier interface .....	14
4.1.3	ManageServiceMetadata interface .....	18
4.1.4	Fault Descriptions .....	20
4.2	Service Metadata Locator - data model .....	21
4.2.1	ServiceMetadataPublisherService datatype.....	21
4.2.2	ServiceMetadataPublisherServiceForParticipant datatype.....	21
4.2.3	ParticipantIdentifier datatype .....	22
4.2.4	ParticipantIdentifier format.....	22
4.2.5	ParticipantIdentifierPage datatype.....	22
4.2.6	MigrationRecord .....	23
5	Service Bindings .....	24
5.1	Services Provided as Web services - characteristics .....	24
5.2	ManageParticipantIdentifier service - binding .....	24

5.2.1	Transport binding.....	24
5.2.2	Security .....	24
5.3	ManageServiceMetadata service - binding .....	24
5.3.1	Transport binding.....	24
5.3.2	Security .....	24
6	DNS Spoof Mitigation.....	24
7	Appendix A: Schema .....	26
7.1	ServiceMetadataLocatorTypes.xsd .....	26
8	Appendix B: WSDLs .....	28
8.1	ManageParticipantIdentifierService.wsdl.....	28
8.2	ManageServiceMetadataService.wsdl.....	33

# 1 Document information

## 1.1 Document history

Date	Version	Initials	Changes
2009-02-12	0.1.0	GS	Created document
2009-02-13	0.1.1	GS	1 <sup>st</sup> outline of full content, XML Schema and WSDL added
2009-02-15	0.1.2	GS	Included logical data model + interface, renamed registries
2009-02-17	0.1.3	GS	Misc. edits
2009-02-17	0.5	GS	Minor edits
2009-04-01	0.7	MJE	Describe logical data model using pseudo-schema, change paging model for List operation, remove flow diagrams Describe that the authentication process defines the Service Metadata for the ManageBusinessIdentifier and ManageRegistryMetadata services Remove Update() operation for the ManageBusinessIdentifier interface Remove the "Business Key Hash" approach for BusinessIdentifiers XSD and WSDLs revised
2009-04-01	0.7	GS	Minor edits
2009-04-30	0.8	MJE	Changed name to Service Metadata Locator
2009-07-08	0.9	MJE	Added operations for migration of Metadata from one Service Metadata Publisher to another Added bulk creation and deletion operations
2009-07-30	0.9	MJE	Updated XSD & WSDL
2009-08-24	0.9	MJE	Addressed Feedback comments: 84, 85, 86, 89, 95, 96, 97, 100, 102, 104, 105, 106, 107, 108, 109, 111, 112, 113, 114, 118, 120, 121, 122, 123, 124, 125, 129, 130
2009-09-01	0.9	MJE	Updated namespaces for XSD and WSDLs to use the stem: <a href="http://busdox.org/serviceMetadata/">http://busdox.org/serviceMetadata/</a> Changed XSD to match agreements of 26 <sup>th</sup> Aug 2009 meeting
2009-09-30	0.9.0.1	MJE	Errata: Fixed WSDLs to use SOAP 1.1 rather than SOAP 1.2
2009-10-01	0.9.5	MJE	Completely new design for SML, using DNS lookups Significant changes to data structures and WSDL files. Interface descriptions in Section 4 updated to match XSD and WSDLs New diagrams for flows added in Section 3
2009-11-17	1.0.0	MJE	Added an Appendix section on DNS Spoof Mitigation
2009-11-25	1.0.0	MJE	XSD and WSDLs tweaked (eg "Business" replaced by "Participant") Removal of CertificateID and replacement by ServiceMetadataPublisherID Description of need for Hash conversion of Participant IDs
2009-11-26	1.0.0	GS	Renamed 'Business identifier' to 'participant identifier' everywhere. Removed DNS spoof mitigation description.
2009-12-21	1.0.0	MJE	Fixed typos. Added extra references to other specifications Extra text describing Migration process

## **1.2 Editors**

Mike Edwards, IBM

## **1.3 Contributors (alphabetically)**

Jens Jakob Andersen, NITA

Kenneth Bengtsson, Alfa1lab

Mikkel Hippe Brun, NITA

Mike Edwards, IBM

Paul Fremantle, WSO2

Thomas Gundel, IT Crew

Philip Helger, Bundesrechenzentrum

Hans Guldager Knudsen, Lenio

Christian Uldall Pedersen, Accenture

Carl-Markus Piswanger, Bundesrechenzentrum

Bergþór Skúlason, NITA

Dennis Jensen Søgaard, Accenture

Gert Sylvest, Avanade

## 1 **2 Introduction**

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

4 The Service Metadata Locator service exposes three interfaces:

- 5 • *Service Metadata discovery interface*. This is the lookup interface which enables senders to  
6 discover service metadata about specific target participants
- 7 • *Manage participant identifiers interface*. This is the interface for Service Metadata publishers for  
8 managing the metadata relating to specific participant identifiers that they make available.
- 9 • *Manage service metadata interface*. This is the interface for Service Metadata publishers for  
10 managing the metadata about their services, e.g. binding, interface profile and key information.

11 This document describes the physical bindings of the logical interfaces in section 04.1.

### 12 **2.1 Goals and non-goals**

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

### 16 **2.2 Terminology**

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

#### 18 **2.2.1 Notational conventions**

19 For a description of notational conventions, see the Common Definitions document [BDEN-CDEF].

#### 20 **2.2.2 Normative references**

21 [BDEN-START] Secure Trusted Asynchronous Reliable Transport (START), STARTProfile.pdf

22 [BDEN-SMP] Service Metadata Publishing, ServiceMetadataPublishing.pdf

23 [BDEN-CDEF] Business Document Exchange Network - Common Definitions, CommonDefinitions.pdf

24 [XML-DSIG] XML Signature Syntax and Processing (Second Edition)

25 <http://www.w3.org/TR/xmlsig-core/>

26 [RFC-2119] "Key words for use in RFCs to Indicate Requirement Levels", <http://www.ietf.org/rfc/rfc2119.txt>

27 [RFC3986] "Uniform Resource Identifier (URI): Generic Syntax", <http://tools.ietf.org/html/rfc3986>

#### 28 **2.2.3 Non-normative references**

29 [WSDL-2.0] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language",

30 <http://www.w3.org/TR/wsdl20/>

31 [WS-I BP] "WS-I Basic Profile Version 1.1"

32 <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>

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

35 [DNS-1034] "Domain Names - Concepts and Facilities"  
36 <http://tools.ietf.org/html/rfc1034>

37 [DNS-1035] "Domain Names - Implementation and Specification"  
38 <http://tools.ietf.org/html/rfc1035>

39 [MD5] The MD5 Message-Digest Algorithm  
40 <http://tools.ietf.org/html/rfc1321>

## 41 **2.3 Namespaces**

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

44

45

### 46 **3 The Service Discovery Process**

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

- 50 • Service Metadata Locator:
    - 51 ○ Discovery interface for senders
    - 52 ○ Management interface for SMPs
  - 53 • Service Metadata Publishers:
    - 54 ○ Discovery interface for senders
- 55

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

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

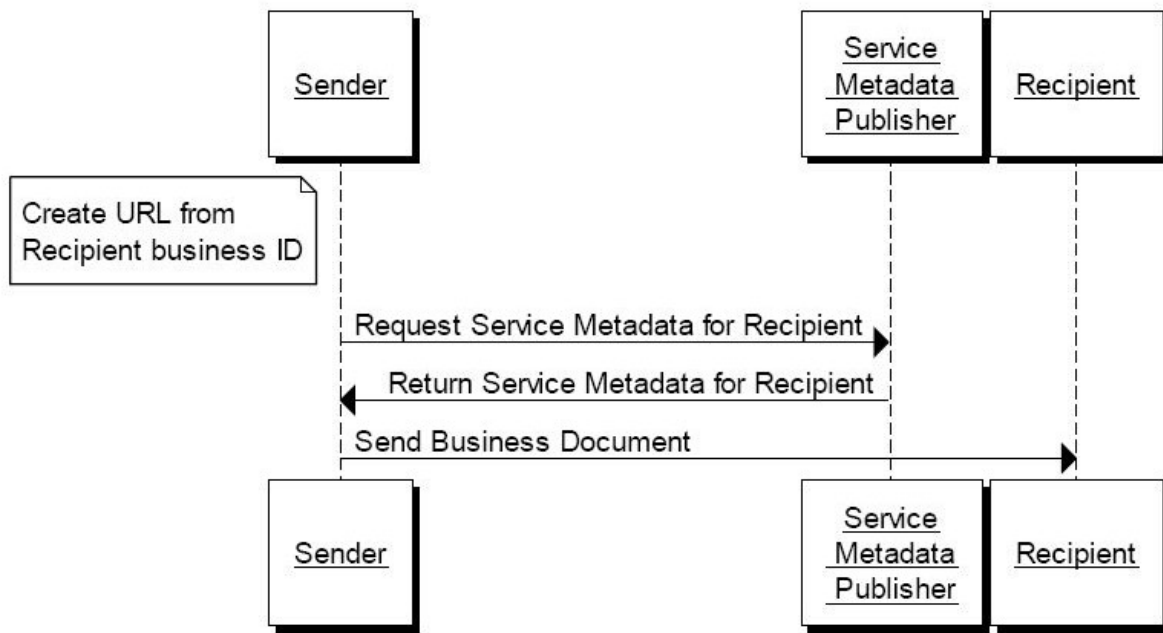
#### 63 **3.1 Discovery flow**

64 For a sender, the first step in the Discovery process is to establish the location of the Service Metadata  
65 relating to the particular Participant Identifier to which the sender wants to transmit a message. Each  
66 participant identifier is registered with one and only one Service Metadata Publisher. The sender  
67 constructs the address for the service metadata for a given recipient participant identifier using a standard  
68 format, as follows:

69 `http://<hash over recipientID>.<schemeID>.<SML domain>/<recipientID>/services/<documentType>`

70 The sender uses this URL in an HTTP GET operation which returns the metadata relating to that recipient  
71 and the specific document type (for details, see the Service Metadata Publishing specification [BDEN-SMP]).  
72 The sender can obtain the information necessary to transmit a message containing that document type to  
73 that recipient from the returned metadata. This sequence is shown in Figure 1.

74 Note that the sender is required to know 2 pieces of information about the recipient - the recipient's  
75 participant ID and the ID of the Scheme of the participant ID (i.e. the format or type of the participant ID).  
76 This provides for flexibility in the types of participant identifier that can be used in the system. Since in  
77 general a participant ID may not have a format that is acceptable in an HTTP URL, the ID is hashed into a  
78 string as described in section 4.1.1 Format of Participant Identifiers.



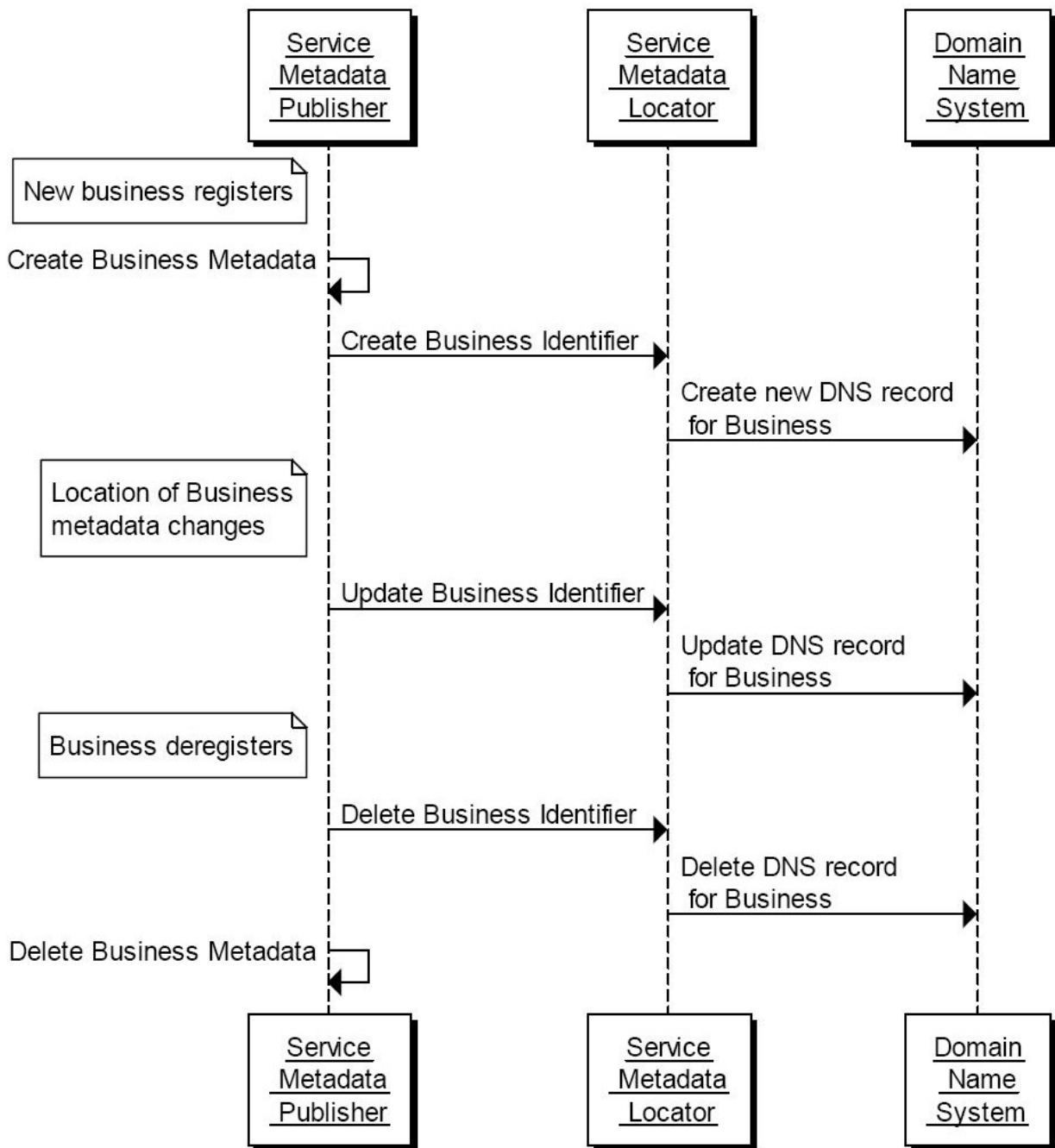
79

80 **Figure 1: Sequence Diagram for Sender transmitting Document to Recipient**

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

### 88 **3.2 Flows Relating to Service Metadata Publishers**

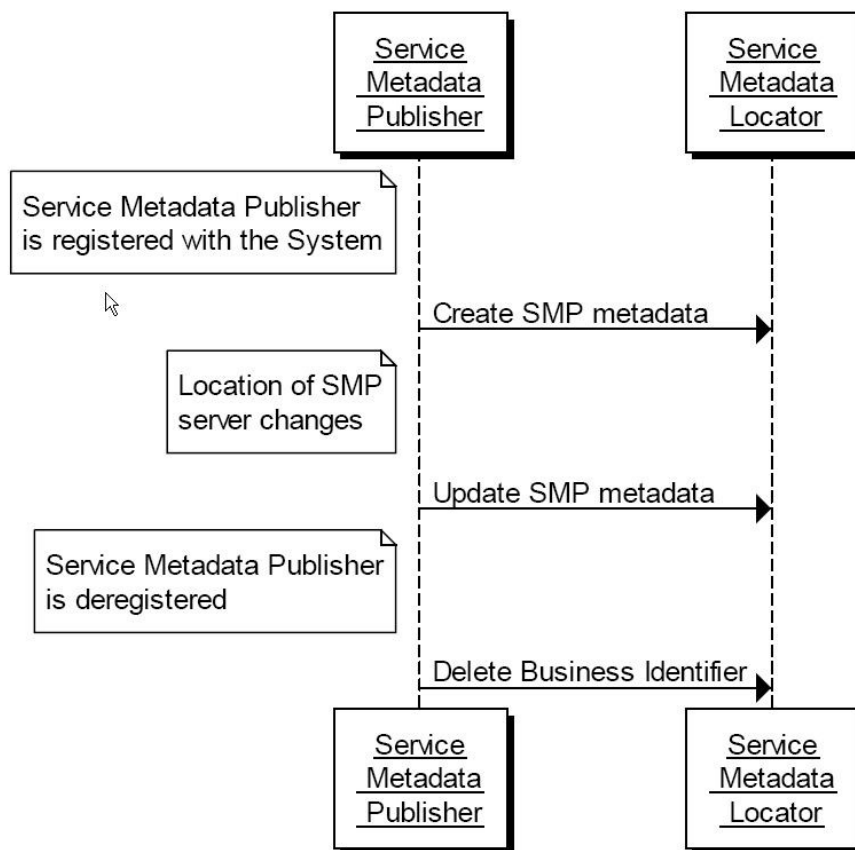
89 The management of the DNS CNAME records for a given participant identifier is performed through the  
 90 Management interface of the Service Metadata Locator. The management interface is primarily for use by  
 91 the Service Metadata Publisher which controls the service metadata for a given participant identifier. Note  
 92 that the DNS CNAME records are *not* manipulated directly by the Service Metadata Publisher, but are  
 93 manipulated by the Service Metadata Locator service following requests made to its Management  
 94 interface. The basic process steps for the SMP to manipulate the metadata relating to a given participant  
 95 are shown in Figure 2.



96

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

98 Each Service Metadata Publisher is required to register the address of its server with the Service Metadata  
 99 Locator. Only once this has been done can information relating to specific Participant Identifiers be  
 100 presented to the SML. The address for the metadata for a given participant is tied to the address of the  
 101 SMP with which the participant is registered. For this purpose, the SMP uses the ManageServiceMetadata  
 102 interface with flows as shown in Figure 3.

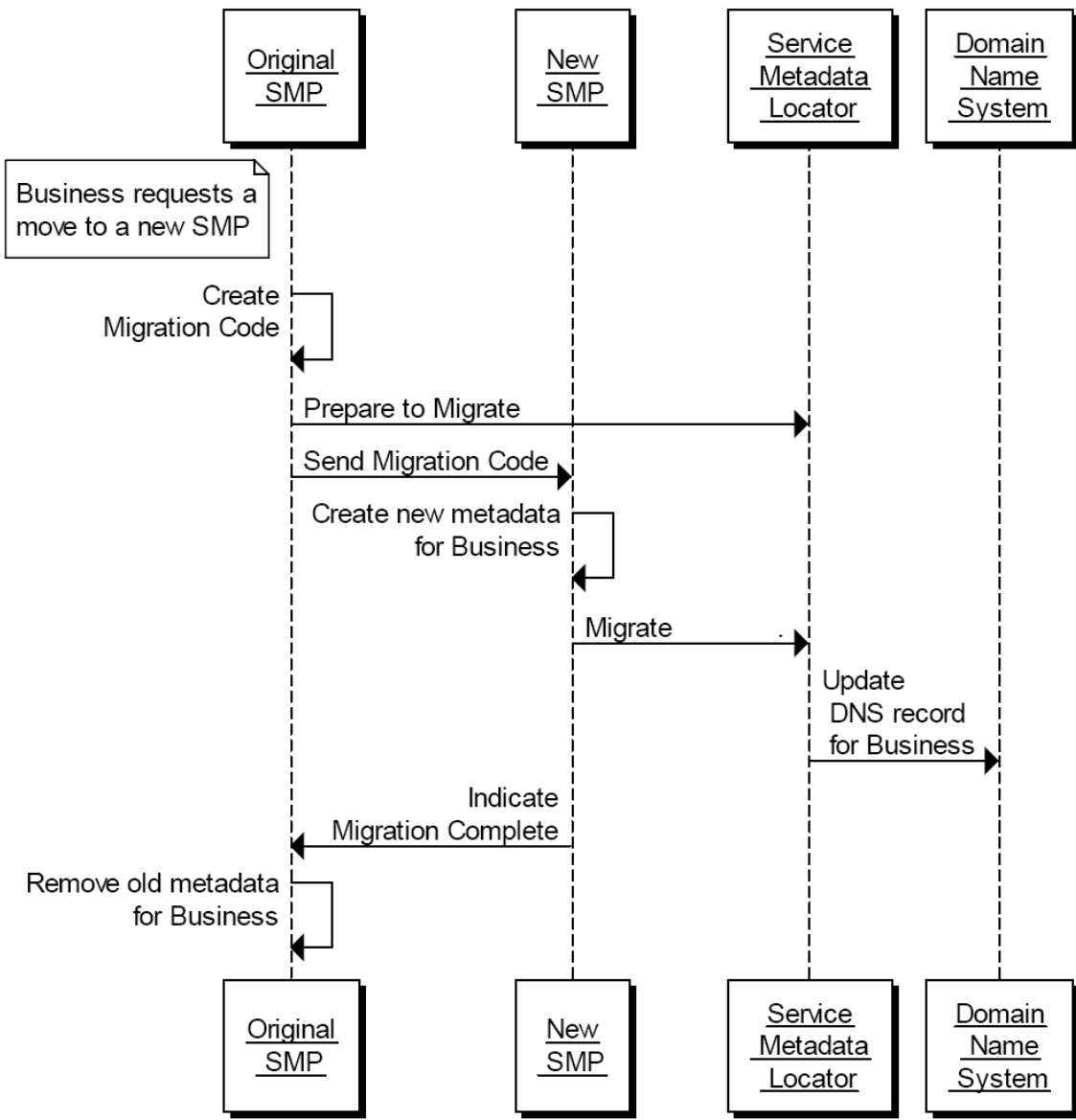


103

104 **Figure 3: Service Metadata Publisher use of the ManageServiceMetadata**

105 Another set of steps relating to SMPs and the SML relates to the migration of the metadata about a  
 106 participant from one SMP to another SMP (for example, the participant decides to change suppliers for this  
 107 function). There are interfaces to the SML to support migrations of this kind, which imply following a  
 108 sequence of steps along the lines shown in figure 4.

109 In this sequence, the original SMP receives a request from a participant to migrate its metadata to a new  
 110 SMP (a step that is done out-of-band: there are no interfaces defined in these specifications for this). The  
 111 SMP generates a Migration Key which is a unique string containing characters and numbers only, with a  
 112 maximum length of 24 characters. The original SMP invokes the PrepareToMigrate operation of the SML  
 113 and then passes the migration key to the new SMP (the key passing is an out-of-band step not defined in  
 114 these specifications). When the new SMP has created the relevant metadata for the participant, it signals  
 115 that it is taking over by invoking the Migrate operation of the SML, which then causes the DNS record(s) for  
 116 that participant ID to be updated to point at the new SMP. Once this switch is complete, the original SMP  
 117 can remove the metadata which it holds for the participant.



118

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

120

## 121 **4 Interfaces and Data Model**

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

### 123 **4.1 Service Metadata Locator Service, logical interface**

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

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

#### 130 **4.1.1 Format of Participant Identifiers**

131 BUSDOX functions by means of logical addresses for the metadata of services offered by a participant, of  
132 the form:

133 `http://<hash over recipientID>.<schemeID>.<SML domain>/<recipientID>/services/<documentType>`

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

140 BUSDOX allocates schemeIDs to conform to this requirement. However, there is no guarantee that the  
141 participant IDs will conform to this requirement for any given scheme (remembering that in many cases the  
142 participant ID scheme will be a pre-existing scheme with its own format rules that might violate the  
143 requirements of a DNS name). Therefore a hash of the participant ID is always used, using the MD5 hash  
144 algorithm [MD5], and prefixed by "B-".

145 An example participant ID is "0010:5798000000001" (see the Common Definitions document for more  
146 details [BDEN-CDEF]), for which the MD5 hash is "e49b223851f6e97cbfce4f72c3402aac".

147

148 **4.1.2 ManageParticipantIdentifier interface**

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

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

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

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

159 <SMP domain> may either be the domain name associated with the SMP, or an alias for it.

160 This implies that all participant identifiers for that schemeID will have addresses that resolve to the single  
161 address of that one SMP - and that as result only one SMP can handle the metadata for all participant  
162 identifiers of that scheme. Wildcard records are indicated through the use of "\*" as the participant  
163 identifier in the operations of the ManageParticipantIdentifier interface.

164 The ManageParticipantIdentifier interface has the following operations:

- 165       • **Create**
- 166       • **CreateList**
- 167       • **Delete**
- 168       • **DeleteList**
- 169       • **PrepareToMigrate**
- 170       • **Migrate**
- 171       • **List**

172 **Create()**

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

- 177       • **Input CreateParticipantIdentifier: ServiceMetadataPublisherServiceForParticipantType** - contains  
178       the Participant Identifier for a given participant and the identifier of the SMP which holds its data
- 179       • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found

- 180 • **Fault: *unauthorizedFault*** - returned if the caller is not authorized to invoke the Create operation
- 181 • **Fault: *badRequestFault*** - returned if the supplied CreateParticipantIdentifier does not contain
- 182 consistent data
- 183 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
- 184 reason

#### 185 **CreateList()**

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

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

#### 199 **Delete()**

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

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

## 211 DeleteList()

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

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

## 223 PrepareToMigrate()

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

230 This operation can only be invoked by the Service Metadata Publisher which currently publishes the  
231 metadata for the specified Participant Identifier.

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

## 243 **Migrate()**

244 Migrates a Participant Identifier already held by the Service Metadata Locator Service to target a new  
245 Service Metadata Publisher. This operation is called by the Service Metadata Publisher which is taking over  
246 the publishing for the Participant Identifier. The operation requires the new Service Metadata Publisher to  
247 provide a migration code which was originally obtained from the old Service Metadata Publisher.

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

250

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

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

261

## 262 **List()**

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

- 266 • **Input Page: PageRequest** - contains a PageRequest containing the ServiceMetadataPublisherID of  
267 the SMP and (if required) an identifier representing the next page of data to retrieve. If the  
268 NextPageIdentifier is absent, the first page is returned.
- 269 • **Output: ParticipantIdentifierPage** - a page of Participant Identifier entries associated with the  
270 Service Metadata Publisher, also containing a <Page/> element containing the identifier that  
271 represents the next page, if any.
- 272 • **Fault: notFoundFault** - returned if the next page or the identifier of the SMP could not be found
- 273 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the List operation
- 274 • **Fault: badRequestFault** - returned if the supplied NextPage does not contain consistent data

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

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

#### 280 **4.1.3 ManageServiceMetadata interface**

281 The ManageServiceMetadata interface allows Service Metadata Publishers to manage the metadata held in  
282 the Service Metadata Locator Service about their service metadata publisher services, e.g. binding,  
283 interface profile and key information.

284 This interface requires authentication of the user. The identity of the user derived from the authentication  
285 process identifies the Service Metadata Publisher associated with the service metadata which is managed  
286 via this interface.

287 The ManageServiceMetadata interface has the following operations:

- 288 • **Create**
- 289 • **Read**
- 290 • **Update**
- 291 • **Delete**

#### 292 **Create()**

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

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

#### 304 **Read()**

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

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

#### 315 Update()

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

- 317 • **Input UpdateServiceMetadataPublisherService: ServiceMetadataPublisherService** - contains the  
318 service metadata for the service metadata publisher, which includes the logical and physical  
319 addresses for the SMP (Domain name and IP address)
- 320 • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found
- 321 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Update operation
- 322 • **Fault: badRequestFault** - returned if the supplied UpdateServiceMetadataPublisherService does  
323 not contain consistent data
- 324 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any  
325 reason

#### 326 Delete()

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

- 328 • **Input DeleteServiceMetadataPublisherService: ServiceMetadataPublisherID** - the unique ID of the  
329 Service Metadata Publisher to delete
- 330 • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found
- 331 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Delete operation
- 332 • **Fault: badRequestFault** - returned if the supplied DeleteServiceMetadataPublisherService does not  
333 contain consistent data
- 334 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any  
335 reason

336 **4.1.4 Fault Descriptions**

337

338 **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

339

340 **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

341

342 **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

343

344 **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

345

## 346 4.2 Service Metadata Locator - data model

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

- 348 • ServiceMetadataPublisher
- 349 • RecipientParticipantIdentifier
- 350 • ParticipantIdentifierPage
- 351 • MigrationRecord

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

### 353 4.2.1 ServiceMetadataPublisherService datatype

354 Represents a Metadata Publisher Service.

```

355 <ServiceMetadataPublisherService>
356   <PublisherEndpoint>
357     <EndpointAddress/>
358   </PublisherEndpoint>
359   <ServiceMetadataPublisherID/>
360 </ServiceMetadataPublisherService>

```

361 ServiceMetadataPublisherService has the following subelements:

- 362 • **PublisherEndpoint (1..1) : PublisherEndpointType** - the technical endpoint address of the Service  
363 Metadata Publisher, which can be used to query information about particular participant  
364 identifiers. ServiceEndpointList is a type defined in the ServiceMetadataPublishingTypes Schema.  
365 The PublisherEndpoint element may be a domain name or an IP address of the SMP, or a wildcard  
366 expression based on the domain name.
- 367 • **ServiceMetadataPublisherID (1..1) : xs:string** - Holds the Unique Identifier of the SMP. When  
368 creating a ServiceMetadataPublisherService record, it is assumed that the publisher ID has been  
369 obtained out of band.

### 370 4.2.2 ServiceMetadataPublisherServiceForParticipant datatype

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

```

372 <ServiceMetadataPublisherServiceForParticipant>
373     <ServiceMetadataPublisherID/>
374     <ids:ParticipantIdentifier/>
375 </ServiceMetadataPublisherServiceForParticipant>

```

376 ServiceMetadataPublisherService has the following subelements:

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

381

### 382 4.2.3 ParticipantIdentifier datatype

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

```

385 <ids:ParticipantIdentifier scheme="xs:string">
386     xs:string
387 </ids:ParticipantIdentifier>
388

```

389 ParticipantIdentifier has the following sub elements:

- 390 • **ParticipantIdentifier (1..1): xs:string** - the participant identifier
- 391 • **@scheme (1..1): xs:string** - the format scheme of the participant identifier

### 392 4.2.4 ParticipantIdentifier format

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

### 395 4.2.5 ParticipantIdentifierPage datatype

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

```

397 <ParticipantIdentifierPage>
398     <ServiceMetadataPublisherID/>
399     <ParticipantIdentifier/>*
400     <NextPageIdentifier/>?
401 </ParticipantIdentifierPage>

```

- 402 • **ServiceMetadataPublisherID (1..1) : xs:string** - Holds the Unique Identifier of the SMP.
- 403 • **ids:ParticipantIdentifier (1..1): xs:string** - the participant identifier
- 404 • **NextPageIdentifier (0..1): xs:string** - an element containing a string identifying the next page of  
405 ParticipantIdentifiers:

```

406 <NextPageIdentifier>
407     [ Identifier for_Next_Page ]

```

408 `</NextPageIdentifier>`

409

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

#### 411 **4.2.6 MigrationRecord**

412 The MigrationRecord represents the data required to control the process of migrating a

413 ParticipantIdentifier from the control of one Service Metadata Publisher to a different Service Metadata

414 Publisher.

415 `<MigrationRecord>`

416 `<ServiceMetadataPublisherID/>`

417 `<ParticipantIdentifier/>*`

418 `<MigrationKey/>?`

419 `</MigrationRecord>`

420 MigrationRecord has the following sub elements:

- 421     • **ServiceMetadataPublisherID (1..1) : xs:string** - Holds the Unique Identifier of the SMP.
- 422     • **ParticipantIdentifier (1..1): ids:ParticipantIdentifierType** - the participant identifier
- 423     • **MigrationKey (1..1): xs:string** - a string which is a unique key controlling the migration of the
- 424         metadata for a given ParticipantIdentifier from one Service Metadata Publisher to another. The
- 425         MigrationKey string is a string of characters and numbers only, with a maximum length of 24
- 426         characters.

427

## 428 **5 Service Bindings**

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

### 431 **5.1 Services Provided as Web services - characteristics**

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

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

### 435 **5.2 ManageParticipantIdentifier service - binding**

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

#### 437 **5.2.1 Transport binding**

438 The 'manage participant identifier' interface is bound to an HTTP SOAP 1.1 transport.

439 See a WSDL for this in "Appendix B: WSDLs".

#### 440 **5.2.2 Security**

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

### 447 **5.3 ManageServiceMetadata service - binding**

448 Service Metadata Publishers use this interface to create or update metadata such as the endpoint address  
449 for retrieval of metadata about specific participant services.

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

#### 451 **5.3.1 Transport binding**

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

453 See a WSDL for this in "Appendix B: WSDLs".

#### 454 **5.3.2 Security**

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

## 457 **6 DNS Spoof Mitigation**

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

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

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

470

## 471 7 Appendix A: Schema

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

### 473 7.1 ServiceMetadataLocatorTypes.xsd

```
474 <?xml version="1.0" encoding="utf-8"?>
475 <xs:schema id="ServiceMetadataPublisherService"
476   targetNamespace="http://busdox.org/serviceMetadata/locator/1.0/"
477   elementFormDefault="qualified"
478   xmlns="http://busdox.org/serviceMetadata/locator/1.0/"
479   xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
480   xmlns:mstns="http://tempuri.org/ServiceMetadataPublisherService.xsd"
481   xmlns:wsa="http://www.w3.org/2005/08/addressing"
482   xmlns:xs="http://www.w3.org/2001/XMLSchema" >
483
484   <xs:import schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-
485 200401-wss-wssecurity-utility-1.0.xsd"
486     namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
487 wssecurity-utility-1.0.xsd" />
488   <xs:import schemaLocation="ws-addr.xsd"
489     namespace="http://www.w3.org/2005/08/addressing" />
490   <xs:import schemaLocation="Identifiers-0.9.xsd"
491     namespace="http://busdox.org/transport/identifiers/1.0/" />
492
493   <xs:element name="ServiceMetadataPublisherID" type="xs:string" />
494   <xs:element name="CreateServiceMetadataPublisherService"
495     type="ServiceMetadataPublisherServiceType" />
496   <xs:element name="ReadServiceMetadataPublisherService"
497     type="ServiceMetadataPublisherIdentifierType" />
498   <xs:element name="UpdateServiceMetadataPublisherService"
499     type="ServiceMetadataPublisherServiceType" />
500   <xs:element name="DeleteServiceMetadataPublisherService"
501     ref="ServiceMetadataPublisherID" />
502   <xs:complexType name="ServiceMetadataPublisherServiceType" >
503     <xs:sequence>
504       <xs:element name="PublisherEndpoint" type="PublisherEndpointType" />
505       <xs:element ref="ServiceMetadataPublisherID" />
506     </xs:sequence>
507   </xs:complexType>
508   <xs:complexType name="PublisherEndpointType" >
509     <xs:sequence>
510       <xs:element name="EndpointAddress" type="xs:anyURI" />
511     </xs:sequence>
512   </xs:complexType>
513   <xs:complexType name="ServiceMetadataPublisherServiceForParticipantType" >
514     <xs:sequence>
515       <xs:element ref="ServiceMetadataPublisherID" />
516       <xs:element ref="ids:ParticipantIdentifier" />
517     </xs:sequence>
518   </xs:complexType>
519   <xs:complexType name="ServiceMetadataPublisherIdentifierType" >
520     <xs:sequence>
521       <xs:element ref="ServiceMetadataPublisherID" />
522     </xs:sequence>
523   </xs:complexType>
524   <xs:element name="CreateParticipantIdentifier"
525     type="ServiceMetadataPublisherServiceForParticipantType" />
```

```

526 <xs:element name="DeleteParticipantIdentifier"
527     type="ServiceMetadataPublisherServiceForParticipantType"/>
528 <xs:element name="ServiceMetadataPublisherService"
529     type="ServiceMetadataPublisherServiceType">
530 </xs:element>
531 <xs:element name="ParticipantIdentifierPage"
532 type="ParticipantIdentifierPageType"/>
533 <xs:element name="CreateList" type="ParticipantIdentifierPageType"/>
534 <xs:element name="DeleteList" type="ParticipantIdentifierPageType"/>
535 <xs:complexType name="ParticipantIdentifierPageType">
536     <xs:sequence>
537         <xs:element ref="ServiceMetadataPublisherID"/>
538         <xs:element ref="ids:ParticipantIdentifier" minOccurs="0"
539             maxOccurs="unbounded"/>
540         <xs:element ref="PageID" minOccurs="0"/>
541     </xs:sequence>
542 </xs:complexType>
543 <xs:element name="PageRequest" type="PageRequestType"/>
544 <xs:complexType name="PageRequestType">
545     <xs:sequence>
546         <xs:element ref="ServiceMetadataPublisherID"/>
547         <xs:element name="NextPageIdentifier" type="xs:string" minOccurs="0"/>
548     </xs:sequence>
549 </xs:complexType>
550 <xs:element name="PrepareMigrationRecord" type="MigrationRecordType"/>
551 <xs:element name="CompleteMigrationRecord" type="MigrationRecordType"/>
552 <xs:complexType name="MigrationRecordType">
553     <xs:sequence>
554         <xs:element ref="ServiceMetadataPublisherID"/>
555         <xs:element ref="ids:ParticipantIdentifier" />
556         <xs:element name="MigrationKey" type="xs:string"/>
557     </xs:sequence>
558 </xs:complexType>
559 <xs:element name="BadRequestFault" type="FaultType"/>
560 <xs:element name="InternalServerError" type="FaultType"/>
561 <xs:element name="NotFoundFault" type="FaultType"/>
562 <xs:element name="UnauthorizedFault" type="FaultType"/>
563 <xs:complexType name="FaultType">
564     <xs:sequence>
565         <xs:element name="FaultMessage" type="xs:string" minOccurs="0"/>
566     </xs:sequence>
567 </xs:complexType>
568 </xs:schema>
569

```

## 570 8 Appendix B: WSDLs

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

### 572 8.1 ManageParticipantIdentifierService.wsdl

```
573 <wsdl:definitions
574     xmlns:tns="http://busdox.org/serviceMetadata/
575         ManageParticipantIdentifierService/1.0/"
576     xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
577     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
578     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
579     xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
580     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
581     xmlns:s="http://www.w3.org/2001/XMLSchema"
582     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
583     name="ManageParticipantIdentifierService"
584     targetNamespace=
585         "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/"
586     xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" >
587 <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
588 <wsdl:types>
589     <s:schema elementFormDefault="qualified"
590         targetNamespace=
591             "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/
592                 1.0/Schema/" >
593         <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
594             schemaLocation="ServiceMetadataLocatorTypes.xsd" />
595     </s:schema>
596 </wsdl:types>
597 <wsdl:message name="createIn">
598     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
599     <wsdl:part name="messagePart" element="lrs:CreateParticipantIdentifier" />
600 </wsdl:message>
601 <wsdl:message name="createOut">
602     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
603 </wsdl:message>
604 <wsdl:message name="deleteIn">
605     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
606     <wsdl:part name="messagePart" element="lrs>DeleteParticipantIdentifier" />
607 </wsdl:message>
608 <wsdl:message name="deleteOut">
609     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
610 </wsdl:message>
611 <wsdl:message name="listIn">
612     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
613     <wsdl:part name="messagePart" element="lrs:PageRequest" />
614 </wsdl:message>
615 <wsdl:message name="listOut">
616     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
617     <wsdl:part name="messagePart" element="lrs:ParticipantIdentifierPage" />
618 </wsdl:message>
619 <wsdl:message name="prepareMigrateIn">
620     <wsdl:part name="prepareMigrateIn" element="lrs:PrepareMigrationRecord" />
621 </wsdl:message>
622 <wsdl:message name="prepareMigrateOut">
623     <wsdl:documentation xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" />
```

```

624 </wsdl:message>
625 <wsdl:message name="migrateIn">
626     <wsdl:part name="migrateIn" element="lrs:CompleteMigrationRecord"/>
627 </wsdl:message>
628 <wsdl:message name="migrateOut">
629     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
630 </wsdl:message>
631 <wsdl:message name="createListIn">
632     <wsdl:part name="createListIn" element="lrs:CreateList"/>
633 </wsdl:message>
634 <wsdl:message name="createListOut">
635     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
636 </wsdl:message>
637 <wsdl:message name="deleteListIn">
638     <wsdl:part name="deleteListIn" element="lrs>DeleteList"/>
639 </wsdl:message>
640 <wsdl:message name="deleteListOut">
641     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
642 </wsdl:message>
643 <wsdl:message name="badRequestFault">
644     <wsdl:part name="fault" element="lrs:BadRequestFault"/>
645 </wsdl:message>
646 <wsdl:message name="internalErrorFault">
647     <wsdl:part name="fault" element="lrs:InternalErrorFault"/>
648 </wsdl:message>
649 <wsdl:message name="notFoundFault">
650     <wsdl:part name="fault" element="lrs:NotFoundFault"/>
651 </wsdl:message>
652 <wsdl:message name="unauthorizedFault">
653     <wsdl:part name="fault" element="lrs:UnauthorizedFault"/>
654 </wsdl:message>
655 <wsdl:portType name="ManageParticipantIdentifierServiceSoap">
656     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
657     <wsdl:operation name="Create">
658         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
659         <wsdl:input message="tns:createIn" />
660         <wsdl:output message="tns:createOut" />
661         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
662         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
663         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
664         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
665     </wsdl:operation>
666     <wsdl:operation name="Delete">
667         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
668         <wsdl:input message="tns:deleteIn" />
669         <wsdl:output message="tns:deleteOut" />
670         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
671         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
672         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
673         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
674     </wsdl:operation>
675     <wsdl:operation name="List">
676         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
677         <wsdl:input message="tns:listIn" />
678         <wsdl:output message="tns:listOut" />
679         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
680         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
681         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
682         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>

```

```

683     </wsdl:operation>
684     <wsdl:operation name="PrepareToMigrate">
685         <wsdl:input message="tns:prepareMigrateIn"/>
686         <wsdl:output message="tns:prepareMigrateOut" />
687         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
688         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
689         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
690         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
691     </wsdl:operation>
692     <wsdl:operation name="Migrate">
693         <wsdl:input message="tns:migrateIn"/>
694         <wsdl:output message="tns:migrateOut"/>
695         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
696         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
697         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
698         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
699     </wsdl:operation>
700     <wsdl:operation name="CreateList">
701         <wsdl:input message="tns:createListIn"/>
702         <wsdl:output message="tns:createListOut" />
703         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
704         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
705         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
706         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
707     </wsdl:operation>
708     <wsdl:operation name="DeleteList">
709         <wsdl:input message="tns:deleteListIn"/>
710         <wsdl:output message="tns:deleteListOut" />
711         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
712         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
713         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
714         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
715     </wsdl:operation>
716 </wsdl:portType>
717 <wsdl:binding name="ManageParticipantIdentifierServiceSoap"
718     type="tns:ManageParticipantIdentifierServiceSoap">
719     <soap11:binding transport="http://schemas.xmlsoap.org/soap/http" />
720     <wsdl:operation name="Create">
721         <soap11:operation soapAction=
722
723 "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
724     :createIn"
725     style="document" />
726     <wsdl:input>
727         <soap11:body use="literal" />
728     </wsdl:input>
729     <wsdl:output>
730         <soap11:body use="literal" />
731     </wsdl:output>
732     <wsdl:fault name="UnauthorizedFault">
733         <soap:fault name="UnauthorizedFault" use="literal"/>
734     </wsdl:fault>
735     <wsdl:fault name="InternalErrorFault">
736         <soap:fault name="InternalErrorFault" use="literal"/>
737     </wsdl:fault>
738     <wsdl:fault name="BadRequestFault">
739         <soap:fault name="BadRequestFault" use="literal"/>
740     </wsdl:fault>
741 </wsdl:operation>

```

```

742     <wsdl:operation name="CreateList">
743         <soap11:operation soapAction=
744
745 "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
746         :createListIn"
747         style="document" />
748     <wsdl:input>
749         <soap11:body use="literal" />
750     </wsdl:input>
751     <wsdl:output>
752         <soap11:body use="literal" />
753     </wsdl:output>
754     <wsdl:fault name="NotFoundFault">
755         <soap:fault name="NotFoundFault" use="literal"/>
756     </wsdl:fault>
757     <wsdl:fault name="UnauthorizedFault">
758         <soap:fault name="UnauthorizedFault" use="literal"/>
759     </wsdl:fault>
760     <wsdl:fault name="InternalServerErrorFault">
761         <soap:fault name="InternalServerErrorFault" use="literal"/>
762     </wsdl:fault>
763     <wsdl:fault name="BadRequestFault">
764         <soap:fault name="BadRequestFault" use="literal"/>
765     </wsdl:fault>
766 </wsdl:operation>
767 <wsdl:operation name="Delete">
768     <soap11:operation soapAction=
769
770 "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
771         :deleteIn"
772         style="document" />
773     <wsdl:input>
774         <soap11:body use="literal" />
775     </wsdl:input>
776     <wsdl:output>
777         <soap11:body use="literal" />
778     </wsdl:output>
779     <wsdl:fault name="NotFoundFault">
780         <soap:fault name="NotFoundFault" use="literal"/>
781     </wsdl:fault>
782     <wsdl:fault name="UnauthorizedFault">
783         <soap:fault name="UnauthorizedFault" use="literal"/>
784     </wsdl:fault>
785     <wsdl:fault name="InternalServerErrorFault">
786         <soap:fault name="InternalServerErrorFault" use="literal"/>
787     </wsdl:fault>
788     <wsdl:fault name="BadRequestFault">
789         <soap:fault name="BadRequestFault" use="literal"/>
790     </wsdl:fault>
791 </wsdl:operation>
792 <wsdl:operation name="DeleteList">
793     <soap11:operation soapAction=
794
795 "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
796         :deleteListIn"
797         style="document" />
798     <wsdl:input>
799         <soap11:body use="literal" />
800 </wsdl:input>

```

```

801     <wsdl:output>
802         <soap11:body use="literal" />
803     </wsdl:output>
804     <wsdl:fault name="NotFoundFault">
805         <soap:fault name="NotFoundFault" use="literal"/>
806     </wsdl:fault>
807     <wsdl:fault name="UnauthorizedFault">
808         <soap:fault name="UnauthorizedFault" use="literal"/>
809     </wsdl:fault>
810     <wsdl:fault name="InternalServerErrorFault">
811         <soap:fault name="InternalServerErrorFault" use="literal"/>
812     </wsdl:fault>
813     <wsdl:fault name="BadRequestFault">
814         <soap:fault name="BadRequestFault" use="literal"/>
815     </wsdl:fault>
816 </wsdl:operation>
817 <wsdl:operation name="List">
818     <soap11:operation soapAction=
819         "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
820         :listIn"
821         style="document" />
822     <wsdl:input>
823         <soap11:body use="literal" />
824     </wsdl:input>
825     <wsdl:output>
826         <soap11:body use="literal" />
827     </wsdl:output>
828     <wsdl:fault name="NotFoundFault">
829         <soap:fault name="NotFoundFault" use="literal"/>
830     </wsdl:fault>
831     <wsdl:fault name="UnauthorizedFault">
832         <soap:fault name="UnauthorizedFault" use="literal"/>
833     </wsdl:fault>
834     <wsdl:fault name="InternalServerErrorFault">
835         <soap:fault name="InternalServerErrorFault" use="literal"/>
836     </wsdl:fault>
837     <wsdl:fault name="BadRequestFault">
838         <soap:fault name="BadRequestFault" use="literal"/>
839     </wsdl:fault>
840 </wsdl:operation>
841 <wsdl:operation name="PrepareToMigrate">
842     <soap11:operation soapAction=
843         "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
844         :prepareMigrateIn"
845         style="document" />
846     <wsdl:input>
847         <soap11:body use="literal" />
848     </wsdl:input>
849     <wsdl:output>
850         <soap11:body use="literal" />
851     </wsdl:output>
852     <wsdl:fault name="NotFoundFault">
853         <soap:fault name="NotFoundFault" use="literal"/>
854     </wsdl:fault>
855     <wsdl:fault name="UnauthorizedFault">
856         <soap:fault name="UnauthorizedFault" use="literal"/>
857     </wsdl:fault>
858     <wsdl:fault name="InternalServerErrorFault">
859         <soap:fault name="InternalServerErrorFault" use="literal"/>

```

```

860     </wsdl:fault>
861     <wsdl:fault name="BadRequestFault">
862         <soap:fault name="BadRequestFault" use="literal"/>
863     </wsdl:fault>
864 </wsdl:operation>
865 <wsdl:operation name="Migrate">
866     <soap11:operation soapAction=
867         "http://busdox.org/serviceMetadata/ManageParticipantIdentifierService/1.0/
868         :migrateIn"
869         style="document" />
870     <wsdl:input>
871         <soap11:body use="literal" />
872     </wsdl:input>
873     <wsdl:output>
874         <soap11:body use="literal" />
875     </wsdl:output>
876     <wsdl:fault name="NotFoundFault">
877         <soap:fault name="NotFoundFault" use="literal"/>
878     </wsdl:fault>
879     <wsdl:fault name="UnauthorizedFault">
880         <soap:fault name="UnauthorizedFault" use="literal"/>
881     </wsdl:fault>
882     <wsdl:fault name="InternalServerErrorFault">
883         <soap:fault name="InternalServerErrorFault" use="literal"/>
884     </wsdl:fault>
885     <wsdl:fault name="BadRequestFault">
886         <soap:fault name="BadRequestFault" use="literal"/>
887     </wsdl:fault>
888 </wsdl:operation>
889 </wsdl:binding>
890 </wsdl:definitions>

```

## 891 **8.2 ManageServiceMetadataService.wsdl**

```

892 <wsdl:definitions
893     xmlns:tns="http://busdox.org/serviceMetadata/
894     ManageServiceMetadataService/1.0/"
895     xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
896     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
897     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
898     xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
899     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
900     xmlns:s="http://www.w3.org/2001/XMLSchema"
901     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
902     name="ManageServiceMetadataService"
903     targetNamespace=
904         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/"
905     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
906     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
907     <wsdl:types>
908         <s:schema elementFormDefault="qualified"
909             targetNamespace=
910                 "http://busdox.org/serviceMetadata/
911                 ManageServiceMetadataService/1.0/Schema/">
912             <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
913                 schemaLocation="ServiceMetadataLocatorTypes.xsd"/>
914         </s:schema>
915     </wsdl:types>
916     <wsdl:message name="createIn">

```

```

917     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
918     <wsdl:part name="messagePart"
919         element="lrs:CreateServiceMetadataPublisherService" />
920 </wsdl:message>
921 <wsdl:message name="createOut">
922     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
923 </wsdl:message>
924 <wsdl:message name="readIn">
925     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
926     <wsdl:part name="messagePart"
927         element="lrs:ReadServiceMetadataPublisherService" />
928 </wsdl:message>
929 <wsdl:message name="readOut">
930     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
931     <wsdl:part name="messagePart"
932         element="lrs:ServiceMetadataPublisherService" />
933 </wsdl:message>
934 <wsdl:message name="updateIn">
935     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
936     <wsdl:part name="messagePart"
937         element="lrs:UpdateServiceMetadataPublisherService" />
938 </wsdl:message>
939 <wsdl:message name="updateOut">
940     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
941 </wsdl:message>
942 <wsdl:message name="deleteIn">
943     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
944     <wsdl:part name="messagePart"
945         element="lrs>DeleteServiceMetadataPublisherService" />
946 </wsdl:message>
947 <wsdl:message name="deleteOut">
948     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
949 </wsdl:message>
950 <wsdl:message name="badRequestFault">
951     <wsdl:part name="fault" element="lrs:BadRequestFault"/>
952 </wsdl:message>
953 <wsdl:message name="internalErrorFault">
954     <wsdl:part name="fault" element="lrs:InternalErrorFault"/>
955 </wsdl:message>
956 <wsdl:message name="notFoundFault">
957     <wsdl:part name="fault" element="lrs:NotFoundFault"/>
958 </wsdl:message>
959 <wsdl:message name="unauthorizedFault">
960     <wsdl:part name="fault" element="lrs:UnauthorizedFault"/>
961 </wsdl:message>
962 <wsdl:portType name="ManageServiceMetadataServiceSoap">
963     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
964     <wsdl:operation name="Create">
965         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
966         <wsdl:input message="tns:createIn" />
967         <wsdl:output message="tns:createOut" />
968         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
969         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
970         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
971     </wsdl:operation>
972     <wsdl:operation name="Read">
973         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
974         <wsdl:input message="tns:readIn" />
975         <wsdl:output message="tns:readOut" />

```

```

976     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
977     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
978     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
979     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
980 </wsdl:operation>
981 <wsdl:operation name="Update">
982     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
983     <wsdl:input message="tns:updateIn" />
984     <wsdl:output message="tns:updateOut" />
985     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
986     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
987     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
988     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
989 </wsdl:operation>
990 <wsdl:operation name="Delete">
991     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
992     <wsdl:input message="tns:deleteIn" />
993     <wsdl:output message="tns:deleteOut" />
994     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
995     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
996     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
997     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
998 </wsdl:operation>
999 </wsdl:portType>
1000 <wsdl:binding name="ManageServiceMetadataServiceSoap"
1001     type="tns:ManageServiceMetadataServiceSoap">
1002     <soap11:binding transport="http://schemas.xmlsoap.org/soap/http" />
1003     <wsdl:operation name="Create">
1004         <soap11:operation soapAction=
1005             "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1006                 :createIn"
1007             style="document" />
1008         <wsdl:input>
1009             <soap11:body use="literal" />
1010         </wsdl:input>
1011         <wsdl:output>
1012             <soap11:body use="literal" />
1013         </wsdl:output>
1014         <wsdl:fault name="UnauthorizedFault">
1015             <soap:fault name="UnauthorizedFault" use="literal"/>
1016         </wsdl:fault>
1017         <wsdl:fault name="InternalErrorFault">
1018             <soap:fault name="InternalErrorFault" use="literal"/>
1019         </wsdl:fault>
1020         <wsdl:fault name="BadRequestFault">
1021             <soap:fault name="BadRequestFault" use="literal"/>
1022         </wsdl:fault>
1023     </wsdl:operation>
1024     <wsdl:operation name="Read">
1025         <soap11:operation soapAction=
1026             "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1027                 :readIn"
1028             style="document" />
1029         <wsdl:input>
1030             <soap11:body use="literal" />
1031         </wsdl:input>
1032         <wsdl:output>
1033             <soap11:body use="literal" />
1034         </wsdl:output>

```

```

1035     <wsdl:fault name="NotFoundFault">
1036         <soap:fault name="NotFoundFault" use="literal"/>
1037     </wsdl:fault>
1038     <wsdl:fault name="UnauthorizedFault">
1039         <soap:fault name="UnauthorizedFault" use="literal"/>
1040     </wsdl:fault>
1041     <wsdl:fault name="InternalServerErrorFault">
1042         <soap:fault name="InternalServerErrorFault" use="literal"/>
1043     </wsdl:fault>
1044     <wsdl:fault name="BadRequestFault">
1045         <soap:fault name="BadRequestFault" use="literal"/>
1046     </wsdl:fault>
1047 </wsdl:operation>
1048     <wsdl:operation name="Update">
1049     <soap11:operation soapAction=
1050         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1051         :updateIn"
1052         style="document" />
1053     <wsdl:input>
1054         <soap11:body use="literal" />
1055     </wsdl:input>
1056     <wsdl:output>
1057         <soap11:body use="literal" />
1058     </wsdl:output>
1059     <wsdl:fault name="NotFoundFault">
1060         <soap:fault name="NotFoundFault" use="literal"/>
1061     </wsdl:fault>
1062     <wsdl:fault name="UnauthorizedFault">
1063         <soap:fault name="UnauthorizedFault" use="literal"/>
1064     </wsdl:fault>
1065     <wsdl:fault name="InternalServerErrorFault">
1066         <soap:fault name="InternalServerErrorFault" use="literal"/>
1067     </wsdl:fault>
1068     <wsdl:fault name="BadRequestFault">
1069         <soap:fault name="BadRequestFault" use="literal"/>
1070     </wsdl:fault>
1071 </wsdl:operation>
1072 <wsdl:operation name="Delete">
1073     <soap11:operation soapAction=
1074         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1075         :deleteIn"
1076         style="document" />
1077     <wsdl:input>
1078         <soap11:body use="literal" />
1079     </wsdl:input>
1080     <wsdl:output>
1081         <soap11:body use="literal" />
1082     </wsdl:output>
1083     <wsdl:fault name="NotFoundFault">
1084         <soap:fault name="NotFoundFault" use="literal"/>
1085     </wsdl:fault>
1086     <wsdl:fault name="UnauthorizedFault">
1087         <soap:fault name="UnauthorizedFault" use="literal"/>
1088     </wsdl:fault>
1089     <wsdl:fault name="InternalServerErrorFault">
1090         <soap:fault name="InternalServerErrorFault" use="literal"/>
1091     </wsdl:fault>
1092     <wsdl:fault name="BadRequestFault">
1093         <soap:fault name="BadRequestFault" use="literal"/>

```

```
1094         </wsdl:fault>
1095     </wsdl:operation>
1096 </wsdl:binding>
1097 </wsdl:definitions>
```