



Service Metadata Locator Profile

*Physical interfaces and bindings for the Service Metadata
Locator Service*



Version 0.9.5

WP8 2009-10-01



Contents

1	Document information	4
1.1	Document history	4
1.2	Editors.....	4
1.3	Contributors (alphabetically).....	4
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	Data model and interfaces	12
4.1	Service Metadata Locator - data model	12
4.1.1	ServiceMetadataPublisherService datatype.....	12
4.1.2	ServiceMetadataPublisherServiceForBusiness datatype	12
4.1.3	SignedServiceMetadataPublisherService data type	13
4.1.4	BusinessIdentifier datatype	13
4.1.5	BusinessIdentifier format	13
4.1.6	BusinessIdentifierPage datatype	13
4.1.7	MigrationRecord.....	14
4.2	Service Metadata Locator Service, logical interface	14
4.2.1	ManageBusinessIdentifier interface.....	15
4.2.2	ManageServiceMetadata interface	19
5	Service Bindings.....	21
5.1	Services Provided as Web services - characteristics.....	21
5.2	ManageBusinessIdentifier service - binding.....	21
5.2.1	Transport binding	21

5.2.2	Security	21
5.3	ManageServiceMetadata service - binding	21
5.3.1	Transport binding	21
5.3.2	Security	21
6	Message signature.....	22
Appendix A: Schema.....		23
6.1	ServiceMetadataLocatorTypes.xsd.....	23
7	Appendix B: WSDLs.....	25
7.1	ManageBusinessIdentifierService.wsdl	25
7.2	ManageServiceMetadataService.wsdl	30

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 st 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: http://busdox.org/serviceMetadata/ Changed XSD to match agreements of 26 th 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

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 Sjøgaard, 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 businesses
- 7 • *Manage business identifiers interface*. This is the interface for Service Metadata publishers for
8 managing the metadata relating to specific business 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 04.2.

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 **2.3 Namespaces**

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

38

39

40 **3 The Service Discovery Process**

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

- 44 • Service Metadata Locator:
 - 45 ○ Discovery interface for senders
 - 46 ○ Management interface for SMPs
 - 47 • Service Metadata Publishers:
 - 48 ○ Discovery interface for senders
- 49

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

51 NOTE: This version of the Service Metadata Locator service specification introduces a new mechanism for
52 the Discovery interface for senders, which is a replacement for the mechanism described in Version 0.9 of
53 the SML specification (and in earlier versions). The new mechanism is based on the the use of DNS
54 (Domain Name System) lookups to find the address of the Service Metadata for a given business ID. The
55 principal reason to adopt this new approach is that it removes the need for a single central server to run
56 the Discovery interface, with its associated single point of failure. Instead the already distributed and
57 highly redundant

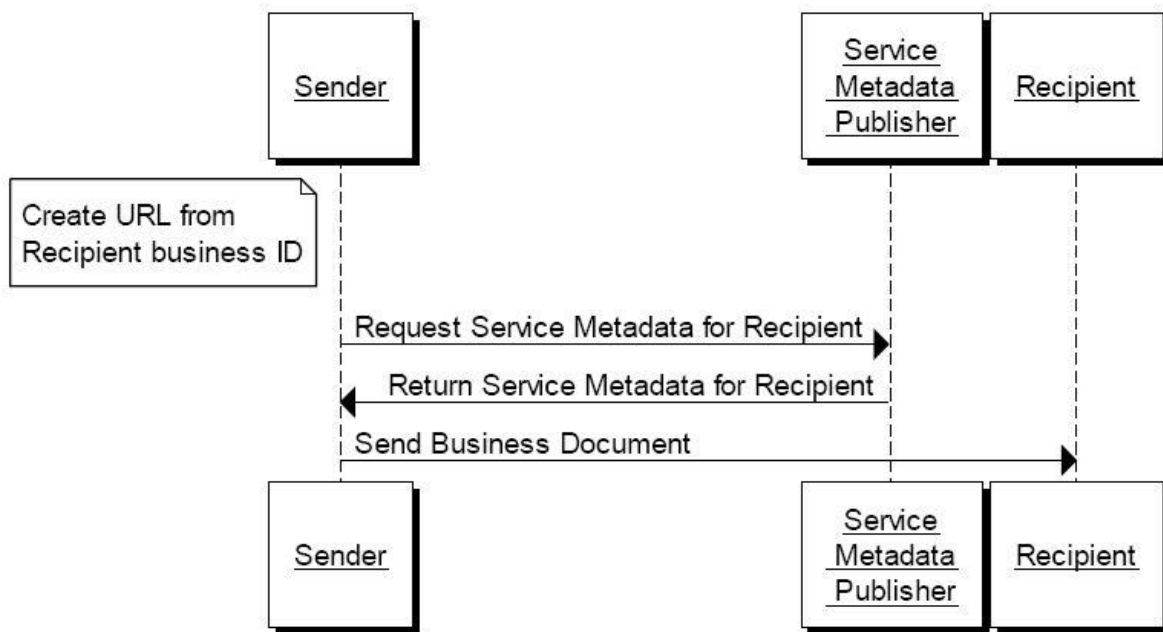
58 **3.1 Discovery flow**

59 For a sender, the first step in the Discovery process is to establish the location of the Service Metadata
60 relating to the particular Business Identifier to which the sender wants to transmit a message. Each
61 business identifier is registered with one and only one Service Metadata Publisher. The sender constructs
62 the address for the service metadata for a given recipient business identifier using a standard format, as
63 follows:

64 `http://<recipientID>.<schemeID>.<SML domain>/<recipientID>/services/<documentType>`

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

69 Note that the sender is required to know 2 pieces of information about the recipient - the recipient's
70 business ID and the ID of the Scheme of the business ID (i.e. the format or type of the business ID). This
71 provides for flexibility in the types of business identifier that can be used in the system.



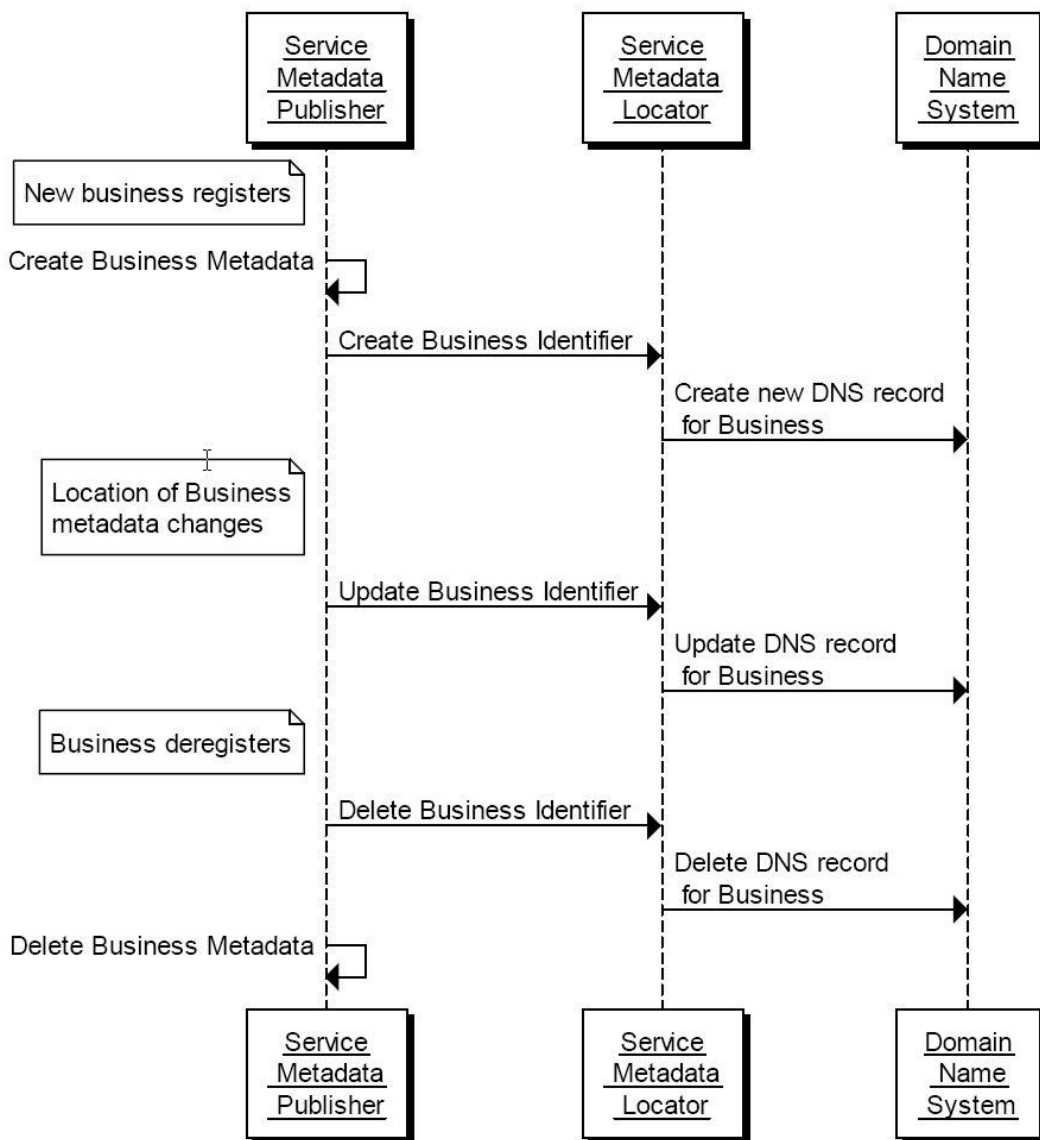
72

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

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

81 **3.2 Flows Relating to Service Metadata Publishers**

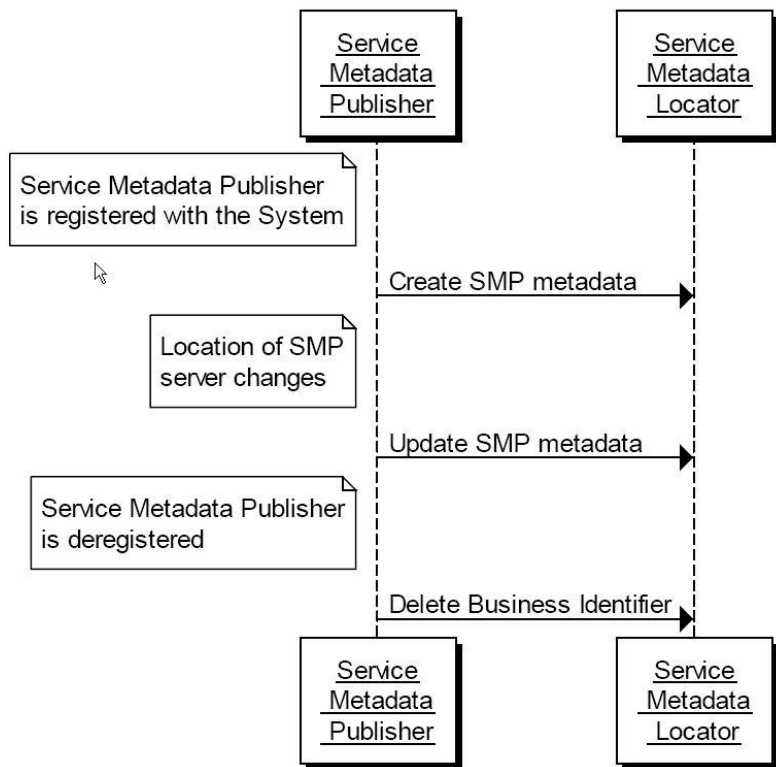
82 The management of the DNS CNAME records for a given business identifier is performed through the
 83 Management interface of the ServiceMetadataLocator. The management interface is primarily for use by
 84 the Service Metadata Publisher which controls the service metadata for a given business identifier. Note
 85 that the DNS CNAME records are *not* manipulated directly by the Service Metadata Publisher, but are
 86 manipulated by the ServiceMetadataLocator service following requests made to its Management interface.
 87 The basic process steps for the SMP to manipulate the metadata relating to a given business are shown in
 88 Figure 2.



89

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

91 Each Service Metadata Publisher is required to register the address of its server with the Service Metadata
 92 Locator. Only once this has been done can information relating to specific Business Identifiers be
 93 presented to the SML. The address for the metadata for a given business is tied to the address of the SMP
 94 with which the business is registered. For this purpose, the SMP uses the ManageServiceMetadata
 95 interface with flows as shown in Figure 3.



96

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

98

99 4 Data model and interfaces

100 This section outlines the data model and service interfaces.

101 4.1 Service Metadata Locator - data model

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

- 103 • ServiceMetadataPublisher
- 104 • SignedServiceMetadataPublisher
- 105 • RecipientBusinessIdentifier
- 106 • BusinessIdentifierPage
- 107 • MigrationRecord

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

109 4.1.1 ServiceMetadataPublisherService datatype

110 Represents a Metadata Publisher Service.

```
111 <ServiceMetadataPublisherService>  
112   <PublisherEndpoint/>  
113   <CertificateUID/>  
114 </ServiceMetadataPublisherService>
```

115 ServiceMetadataPublisherService has the following subelements:

- 116 • **PublisherEndpoint (1..1) : EndpointType** - the technical endpoint address of the Service Metadata
117 Publisher, which can be used to query information about particular business identifiers.
118 ServiceEndpointList is a type defined in the ServiceMetadataPublishingTypes Schema.
- 119 • **CertificateUID (1..1) : String** - Holds the Subject Unique Identifier of the certificate of the
120 destination SMP. A client SHOULD validate that the Subject Unique Identifier of the certificate used
121 to sign the SMP resource matches the Subject Unique Identifier published in the SML.

122 4.1.2 ServiceMetadataPublisherServiceForBusiness datatype

123 Represents a Metadata Publisher Service containing information about a particular Business Identifier.

```
124 <ServiceMetadataPublisherServiceForBusiness>  
125   <CertificateUID/>  
126   <ids:BusinessIdentifier/>  
127 </ServiceMetadataPublisherServiceForBusiness>
```

128 ServiceMetadataPublisherService has the following subelements:

- 129 • **CertificateUID (1..1) : String** - Holds the Subject Unique Identifier of the certificate of the
130 destination SMP. A client SHOULD validate that the Subject Unique Identifier of the certificate used
131 to sign the SMP resource matches the Subject Unique Identifier published in the SML.

- 132 • ***Ids:BusinessIdentifier (1..1):ids: BusinessIdentifierType*** - the BusinessIdentifier which has its
133 services registered in the Service Metadata Publisher. See the “BusinessIdentifier” section on the
134 format.

135

136 **4.1.3 SignedServiceMetadataPublisherService data type**

137 A signed form of the ServiceMetadataPublisherService data type, where the signature guarantees that the
138 service metadata publisher is trusted (through an agreement between the service metadata publisher and
139 the Service Metadata Locator service) and that the public key is associated with the associated endpoint.

```
140 <SignedServiceMetadataPublisherService  
141     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
142     <ServiceMetadataPublisherService/>  
143     <ds:Signature/>  
144 </SignedServiceMetadataPublisherService>
```

145 SignedServiceMetadataPublisherService has the following sub elements:

- 146 • ***ServiceMetadataPublisherService (1..1): ServiceMetadataPublisherServiceForBusinessType*** - the
147 service metadata publisher information
- 148 • ***Signature (1..1): ds:Signature*** - the signature assuring the information in
149 ServiceMetadataPublisherService.

150 **4.1.4 BusinessIdentifier datatype**

151 Represents a BusinessIdentifier which has its service metadata held by a specific Service Metadata
152 Publisher.

```
153 <ids:BusinessIdentifier scheme="xs:anyUri">xs:string</ids:BusinessIdentifier>  
154
```

155 BusinessIdentifier has the following sub elements:

- 156 • ***BusinessIdentifier (1..1): xs:string*** - the business identifier
- 157 • ***@scheme (1..1): xs:anyURI*** - the format scheme of the business identifier

158 **4.1.5 BusinessIdentifier format**

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

161 **4.1.6 BusinessIdentifierPage datatype**

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

```
163 <BusinessIdentifierPage>  
164     <CertificateUID/>  
165     <BusinessIdentifier/*>  
166     <NextPageIdentifier/?>  
167 </BusinessIdentifierPage>
```

- 168 • **CertificateUID (1..1) : String** - Holds the Subject Unique Identifier of the certificate of the SMP
169 holding the service metadata for these business identifiers.
- 170 • **ids:BusinessIdentifier (1..1): xs:string** - the business identifier
- 171 • **NextPageIdentifier (0..1): xs:string** - an element containing a string identifying the next page of
172 BusinessIdentifiers:

```
173 <Page>  
174   [ Identifier for_Next_Page ]  
175 </Page>
```

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

178 4.1.7 MigrationRecord

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

```
181 <MigrationRecord>  
182   <CertificateUID/>  
183   <ids:BusinessIdentifier/>  
184   <MigrationKey/>  
185 </MigrationRecord>
```

186 MigrationRecord has the following sub elements:

- 187 • **CertificateUID (1..1) : String** - Holds the Subject Unique Identifier of the certificate of the
188 destination SMP. The SML will validate that the Subject Unique Identifier of the certificate used to
189 sign the migration record.
- 190 • **ids:BusinessIdentifier (1..1): xs:string** - the business identifier
- 191 • **MigrationKey (1..1): xs:string** - a string which is a unique key controlling the migration of the
192 metadata for a given BusinessIdentifier from one Service Metadata Publisher to another. The
193 MigrationKey string is a string of characters and numbers only, with a maximum length of 24
194 characters.

195 4.2 Service Metadata Locator Service, logical interface

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

- 197 • *Manage business identifiers interface*. This is the interface for Service Metadata Publishers for
198 managing the registered business identifiers they expose.
- 199 • *Manage service metadata interface*. This is the interface for Service Metadata Publishers for
200 managing the metadata about their metadata publishing service, e.g. binding, interface profile and
201 key information.

202

203 4.2.1 ManageBusinessIdentifier interface

204 The ManageBusinessIdentifier interface allows Service Metadata Publishers to manage the information in
205 the Service Metadata Locator Service relating to individual business identifiers for which they hold
206 metadata.

207 This interface requires authentication of the Service Metadata Publisher. The identity of the Service
208 Metadata Publisher derived from the authentication process identifies the Service Metadata Publisher
209 associated with the BusinessIdentifier(s) which are managed via this interface.

210 The ManageBusinessIdentifier interface has the following operations:

- 211 • **Create**
- 212 • **CreateList**
- 213 • **Delete**
- 214 • **DeleteList**
- 215 • **PrepareToMigrate**
- 216 • **Migrate**
- 217 • **List**

218 **Create()**

219 - creates an entry in the Service Metadata Locator Service for information relating to a specific business
220 identifier. Regardless of the number of services a recipient exposes, only one record corresponding to the
221 business identifier is created in the Service Metadata Locator Service by the Service Metadata Publisher
222 which exposes the services for that business.

- 223 • **Input CreateBusinessIdentifier: ServiceMetadataPublisherServiceForBusinessType** - contains the
224 BusinessIdentifier for a given business and the identifier of the SMP which holds its data
- 225 • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found
- 226 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Create operation
- 227 • **Fault: badRequestFault** - returned if the supplied CreateBusinessIdentifier does not contain
228 consistent data
- 229 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
230 reason

231 **CreateList()**

232 - creates a set of entries in the Service Metadata Locator Service for information relating to a list of
233 business identifiers. Regardless of the number of services a recipient exposes, only one record

234 corresponding to each business identifier is created in the Service Metadata Locator Service by the Service
235 Metadata Publisher which exposes the services for that business.

- 236 • **Input CreateList: BusinessIdentifierPage** - contains the list of BusinessIdentifiers for the businesses
237 which are added to the Service Metadata Locator Service. The NextPageIdentifier element is
238 absent.
- 239 • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found
- 240 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the CreateList
241 operation
- 242 • **Fault: badRequestFault** - returned if the supplied CreateList does not contain consistent data
- 243 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
244 reason

245 The maximum physical size of a single page of data supplied by CreateList() is 2Mb.

246 **Delete()**

247 - deletes the information that the Service Metadata Locator Service holds for a specific Business Identifier.

- 248 • **Input DeleteBusinessIdentifier: ServiceMetadataPublisherServiceForBusinessType** - contains the
249 BusinessIdentifier for a given business and the identifier of the SMP that publishes its metadata
- 250 • **Fault: notFoundFault** - returned if the business identifier or the identifier of the SMP could not be
251 found
- 252 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Delete operation
- 253 • **Fault: badRequestFault** - returned if the supplied DeleteBusinessIdentifier does not contain
254 consistent data
- 255 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
256 reason

257 **DeleteList()**

258 - deletes the information that the Service Metadata Locator Service holds for a list of Business Identifiers.

- 259 • **Input DeleteList: BusinessIdentifier** - contains the list of BusinessIdentifiers for the businesses
260 which are removed from the Service Metadata Locator Service. The NextPageIdentifier element is
261 absent.
- 262 • **Fault: notFoundFault** - returned if one or more business identifiers or the identifier of the SMP
263 could not be found

- 264 • **Fault: *unauthorizedFault*** - returned if the caller is not authorized to invoke the DeleteList
265 operation
- 266 • **Fault: *badRequestFault*** - returned if the supplied DeleteList does not contain consistent data
- 267 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
268 reason

269 The maximum physical size of a single page of data supplied by DeleteList() is 2Mb.

270 **PrepareToMigrate()**

271 - prepares a BusinessIdentifier for migration to a new Service Metadata Publisher. This operation is called
272 by the Service Metadata Publisher which currently publishes the metadata for the BusinessIdentifier. The
273 Service Metadata Publisher supplies a MigrationCode which is used to control the migration process. The
274 MigrationCode must be passed (out of band) to the Service Metadata Publisher which is taking over the
275 publishing of the metadata for the BusinessIdentifier and which **MUST** be used on the invocation of the
276 Migrate() operation.

277 This operation can only be invoked by the Service Metadata Publisher which currently publishes the
278 metadata for the specified BusinessIdentifier.

- 279 • **Input *PrepareMigrationRecord: MigrationRecordType*** - contains the Migration Key and the
280 BusinessIdentifier which is about to be migrated from one Service Metadata Publisher to another.
- 281 • **Fault: *notFoundFault*** - returned if the business identifier or the identifier of the SMP could not be
282 found
- 283 • **Fault: *unauthorizedFault*** - returned if the caller is not authorized to invoke the PrepareToMigrate
284 operation
- 285 • **Fault: *badRequestFault*** - returned if the supplied PrepateMigrationRecord does not contain
286 consistent data
- 287 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
288 reason

289 **Migrate()**

290 - migrates a BusinessIdentifier already held by the Service Metadata Locator Service to target a new
291 ServiceMetadata Publisher. This operation is called by the Service Metadata Publisher which is taking over
292 the publishing for the BusinessIdentifier. The operation requires the new Service Metadata Publisher to
293 provide a migration code which was originally obtained from the old Service Metadata Publisher.

294 The PrepareToMigrate operation **MUST** have been previously invoked for the supplied BusinessIdentifier,
295 using the same MigrationCode, otherwise the Migrate() operation fails.

296

297 Following the successful invocation of this operation, the Locate() operation of the
298 ServiceMetadataDiscovery interface returns the new Service Metadata Publisher.

- 299 • **Input CompleteMigrationRecord: MigrationRecordType** - contains the Migration Key and the
300 BusinessIdentifier which is to be migrated from one Service Metadata Publisher to another.
- 301 • **Fault: notFoundFault** - returned if the migration key or the identifier of the SMP could not be found
- 302 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Migrate operation
- 303 • **Fault: badRequestFault** - returned if the supplied CompleteMigrationRecord does not contain
304 consistent data
- 305 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
306 reason

307

308 **List()**

309 - is used to retrieve a list of all business identifiers associated with a single Service Metadata Publisher, for
310 synchronization purposes. Since this list may be large, it is returned as pages of data, with each page being
311 linked from the previous page.

- 312 • **Input Page: PageRequest** - contains a PageRequest containing the Certificate UID of the SMP and
313 (if required) the URI for the page of data to retrieve. If the URI is null, the first page of data is
314 retrieved.
- 315 • **Output: BusinessIdentifierPage** - a page of BusinessIdentifier entries associated with the Service
316 Metadata Publisher, also containing a <Page/> element containing the URI to the next page, if any.
- 317 • **Fault: notFoundFault** - returned if the next page or the identifier of the SMP could not be found
- 318 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the List operation
- 319 • **Fault: badRequestFault** - returned if the supplied NextPage does not contain consistent data
- 320 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
321 reason

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

325 The maximum physical size of a single page of data returned by List() is 2Mb.

326

327 4.2.2 ManageServiceMetadata interface

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

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

334 The ManageServiceMetadata interface has the following operations:

- 335 • **Create**
- 336 • **Read**
- 337 • **Update**
- 338 • **Delete**

339 **Create()**

340 - establishes a Service Metadata Publisher metadata record, containing the metadata about the Service
341 Metadata Publisher- information as outlined in the *SignedServiceMetadataPublisherService* data type.

- 342 • **Input CreateServiceMetadataPublisherService: ServiceMetadataPublisherService** - contains the
343 service metadata publisher information
- 344 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Create operation
- 345 • **Fault: badRequestFault** - returned if the supplied CreateServiceMetadataPublisherService does not
346 contain consistent data
- 347 • **Fault: internalErrorFault** - returned if the SML service is unable to process the request for any
348 reason

349 **Read()**

350 - retrieves the Service Metadata Publisher record for the service metadata publisher

- 351 • **Input ReadServiceMetadataPublisherService: CertificateUID** - the certificate UID of the Service
352 Metadata Publisher for which the record is required
- 353 • **Output: SignedServiceMetadataPublisherService** - the service metadata publisher record, in the
354 form of a SignedServiceMetadataPublisherService data type
- 355 • **Fault: notFoundFault** - returned if the identifier of the SMP could not be found
- 356 • **Fault: unauthorizedFault** - returned if the caller is not authorized to invoke the Read operation

- 357 • **Fault: *badRequestFault*** - returned if the supplied parameter does not contain consistent data
- 358 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
- 359 reason

360 **Update()**

361 - updates the Service Metadata Publisher record for the service metadata publisher

- 362 • **Input *UpdateServiceMetadataPublisheService: ServiceMetadataPublisherService*** - contains the
- 363 service metadata for the service metadata publisher
- 364 • **Fault: *notFoundFault*** - returned if the identifier of the SMP could not be found
- 365 • **Fault: *unauthorizedFault*** - returned if the caller is not authorized to invoke the Update operation
- 366 • **Fault: *badRequestFault*** - returned if the supplied *UpdateServiceMetadataPublisheService* does
- 367 not contain consistent data
- 368 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
- 369 reason

370 **Delete()**

371 - deletes the Service Metadata Publisher record for the service metadata publisher

- 372 • **Input *DeleteServiceMetadataPublisherService: CertificateUID*** - the certificate UID of the Service
- 373 Metadata Publisher to delete
- 374 • **Fault: *notFoundFault*** - returned if the identifier of the SMP could not be found
- 375 • **Fault: *unauthorizedFault*** - returned if the caller is not authorized to invoke the Delete operation
- 376 • **Fault: *badRequestFault*** - returned if the supplied *DeleteServiceMetadataPublisherService* does not
- 377 contain consistent data
- 378 • **Fault: *internalErrorFault*** - returned if the SML service is unable to process the request for any
- 379 reason

380

381 **5 Service Bindings**

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

384 **5.1 Services Provided as Web services - characteristics**

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

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

388 **5.2 ManageBusinessIdentifier service - binding**

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

390 **5.2.1 Transport binding**

391 The 'manage business identifier' interface is bound to an HTTP SOAP 1.1 transport.

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

393 **5.2.2 Security**

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

400 Return messages are signed with XML-signature; see [the section on Message Signature](#) for details.

401 **5.3 ManageServiceMetadata service - binding**

402 Service Metadata Publishers use this interface to create or update metadata such as the endpoint address
403 for retrieval of metadata about specific business services.

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

405 **5.3.1 Transport binding**

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

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

408 **5.3.2 Security**

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

411 Return messages are signed with XML-signature; see [the section on Message Signature](#) for details.

412

413 **6 Message signature**

414 The messages returned by the ServiceMetadataDiscovery ManageBusinessIdentifier and
415 ManageServiceMetadata services are signed by the Service Metadata Locator Service with XML-Signature
416 according to the standard <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>.

417 The signature MUST be an enveloped XML signature represented via an <ds:Signature> element embedded
418 in the <SignedServiceMetadataPublisherService> element. The <ds:Signature> element MUST be
419 constructed according to the following rules:

- 420 • The <Reference> MUST use exactly one Transform being:
421 “<http://www.w3.org/2000/09/xmlsig#envelopedsignature>”
- 422 • The <ds:KeyInfo> element MUST contain an <ds:X509Data> element with an <ds:X509Certificate>
423 sub-element containing the signer’s X.509 certificate as a base64 encoded value.
- 424 • The canonicalization algorithm MUST be <http://www.w3.org/2001/10/xml-exc-c14n#>
- 425 • The SignatureMethod MUST be <http://www.w3.org/2000/09/xmlsig#rsa-sha1>
- 426 • The DigestMethod MUST be <http://www.w3.org/2000/09/xmlsig#sha1>

427 When validating the signature, the requestor MUST check that the certificate used for signing the response
428 is identical to the one published (out-of-band) for the ServiceMetadataLocator service. In other words,
429 there is a *single* well-known certificate in the infrastructure that can be used for this purpose.

430

431 **Appendix A: Schema**

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

433 **6.1 ServiceMetadataLocatorTypes.xsd**

```
434 <?xml version="1.0" encoding="utf-8"?>
435 <xs:schema id="ServiceMetadataPublisherService"
436   targetNamespace="http://busdox.org/serviceMetadata/locator/1.0/"
437   elementFormDefault="qualified"
438   xmlns="http://busdox.org/serviceMetadata/locator/1.0/"
439   xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
440   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
441   xmlns:mstns="http://tempuri.org/ServiceMetadataPublisherService.xsd"
442   xmlns:wsa="http://www.w3.org/2005/08/addressing"
443   xmlns:xs="http://www.w3.org/2001/XMLSchema">
444
445   <xs:import schemaLocation="xmldsig-core-schema.xsd"
446     namespace="http://www.w3.org/2000/09/xmldsig#" />
447   <xs:import schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-
448 200401-wss-wssecurity-utility-1.0.xsd"
449     namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
450 wssecurity-utility-1.0.xsd"/>
451   <xs:import schemaLocation="ws-addr.xsd"
452     namespace="http://www.w3.org/2005/08/addressing" />
453   <xs:import schemaLocation="Identifiers-0.9.xsd"
454     namespace="http://busdox.org/transport/identifiers/1.0/" />
455
456   <xs:element name="CertificateUID" type="xs:string" />
457   <xs:element name="CreateServiceMetadataPublisherService"
458     type="ServiceMetadataPublisherServiceType" />
459   <xs:element name="ReadServiceMetadataPublisherService"
460     type="ServiceMetadataPublisherIdentifierType" />
461   <xs:element name="UpdateServiceMetadataPublisherService"
462     type="ServiceMetadataPublisherServiceType" />
463   <xs:element name="DeleteServiceMetadataPublisherService"
464     type="ServiceMetadataPublisherIdentifierType" />
465   <xs:complexType name="ServiceMetadataPublisherServiceType">
466     <xs:sequence>
467       <xs:element name="PublisherEndpoint" type="xs:anyURI" />
468       <xs:element ref="CertificateUID" />
469     </xs:sequence>
470   </xs:complexType>
471   <xs:complexType name="ServiceMetadataPublisherServiceForBusinessType">
472     <xs:sequence>
473       <xs:element ref="CertificateUID" />
474       <xs:element ref="ids:BusinessIdentifier" />
475     </xs:sequence>
476   </xs:complexType>
477   <xs:complexType name="ServiceMetadataPublisherIdentifierType">
478     <xs:sequence>
479       <xs:element ref="CertificateUID" />
480     </xs:sequence>
481   </xs:complexType>
482   <xs:element name="CreateBusinessIdentifier"
483     type="ServiceMetadataPublisherServiceForBusinessType" />
484   <xs:element name="DeleteBusinessIdentifier"
485     type="ServiceMetadataPublisherServiceForBusinessType" />
```

```

486 <xs:complexType name="SignedServiceMetadataPublisherServiceType">
487   <xs:sequence>
488     <xs:element name="ServiceMetadataPublisherService"
489       type="ServiceMetadataPublisherServiceType" />
490     <xs:element ref="ds:Signature" />
491   </xs:sequence>
492 </xs:complexType>
493 <xs:element name="SignedServiceMetadataPublisherService"
494   type="SignedServiceMetadataPublisherServiceType">
495 </xs:element>
496 <xs:element name="BusinessIdentifierPage" type="BusinessIdentifierPageType"/>
497 <xs:element name="CreateList" type="BusinessIdentifierPageType"/>
498 <xs:element name="DeleteList" type="BusinessIdentifierPageType"/>
499 <xs:complexType name="BusinessIdentifierPageType">
500   <xs:sequence>
501     <xs:element ref="CertificateUID"/>
502     <xs:element ref="ids:BusinessIdentifier" minOccurs="0"
503       maxOccurs="unbounded"/>
504     <xs:element ref="PageID" minOccurs="0"/>
505   </xs:sequence>
506 </xs:complexType>
507 <xs:element name="PageRequest" type="PageRequestType"/>
508 <xs:complexType name="PageRequestType">
509   <xs:sequence>
510     <xs:element ref="CertificateUID"/>
511     <xs:element ref="PageID" minOccurs="0"/>
512   </xs:sequence>
513 </xs:complexType>
514 <xs:element name="PageID" type="xs:anyURI"/>
515 <xs:element name="PrepareMigrationRecord" type="MigrationRecordType"/>
516 <xs:element name="CompleteMigrationRecord" type="MigrationRecordType"/>
517 <xs:complexType name="MigrationRecordType">
518   <xs:sequence>
519     <xs:element ref="CertificateUID"/>
520     <xs:element ref="ids:BusinessIdentifier" />
521     <xs:element name="MigrationKey" type="xs:string"/>
522   </xs:sequence>
523 </xs:complexType>
524 <xs:element name="BadRequestFault" type="FaultType"/>
525 <xs:element name="InternalServerError" type="FaultType"/>
526 <xs:element name="NotFoundFault" type="FaultType"/>
527 <xs:element name="UnauthorizedFault" type="FaultType"/>
528 <xs:complexType name="FaultType">
529   <xs:sequence>
530     <xs:element name="FaultMessage" type="xs:string" minOccurs="0"/>
531   </xs:sequence>
532 </xs:complexType>
533
534 </xs:schema>
535
536

```

537 7 Appendix B: WSDLs

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

539 7.1 ManageBusinessIdentifierService.wsdl

```
540 <wsdl:definitions
541     xmlns:tns="http://busdox.org/serviceMetadata/
542         ManageBusinessIdentifierService/1.0/"
543     xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
544     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
545     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
546     xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
547     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
548     xmlns:s="http://www.w3.org/2001/XMLSchema"
549     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
550     name="ManageBusinessIdentifierService"
551     targetNamespace=
552         "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/"
553     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
554     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
555     <wsdl:types>
556         <s:schema elementFormDefault="qualified"
557             targetNamespace=
558                 "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/
559                 1.0/Schema/">
560             <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
561                 schemaLocation="ServiceMetadataLocatorTypes-0.95.xsd"/>
562         </s:schema>
563     </wsdl:types>
564     <wsdl:message name="createIn">
565         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
566         <wsdl:part name="messagePart" element="lrs:CreateBusinessIdentifier" />
567     </wsdl:message>
568     <wsdl:message name="createOut">
569         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
570     </wsdl:message>
571     <wsdl:message name="deleteIn">
572         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
573         <wsdl:part name="messagePart" element="lrs>DeleteBusinessIdentifier" />
574     </wsdl:message>
575     <wsdl:message name="deleteOut">
576         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
577     </wsdl:message>
578     <wsdl:message name="listIn">
579         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
580         <wsdl:part name="messagePart" element="lrs:PageRequest" />
581     </wsdl:message>
582     <wsdl:message name="listOut">
583         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
584         <wsdl:part name="messagePart" element="lrs:BusinessIdentifierPage" />
585     </wsdl:message>
586     <wsdl:message name="prepareMigrateIn">
587         <wsdl:part name="prepareMigrateIn" element="lrs:PrepareMigrationRecord"/>
588     </wsdl:message>
589     <wsdl:message name="prepareMigrateOut">
590         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
591     </wsdl:message>
```

```

592 <wsdl:message name="migrateIn">
593     <wsdl:part name="migrateIn" element="lrs:CompleteMigrationRecord"/>
594 </wsdl:message>
595 <wsdl:message name="migrateOut">
596     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
597 </wsdl:message>
598 <wsdl:message name="createListIn">
599     <wsdl:part name="createListIn" element="lrs:CreateList"/>
600 </wsdl:message>
601 <wsdl:message name="createListOut">
602     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
603 </wsdl:message>
604 <wsdl:message name="deleteListIn">
605     <wsdl:part name="deleteListIn" element="lrs>DeleteList"/>
606 </wsdl:message>
607 <wsdl:message name="deleteListOut">
608     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
609 </wsdl:message>
610 <wsdl:message name="badRequestFault">
611     <wsdl:part name="fault" element="lrs:BadRequestFault"/>
612 </wsdl:message>
613 <wsdl:message name="internalErrorFault">
614     <wsdl:part name="fault" element="lrs:InternalErrorFault"/>
615 </wsdl:message>
616 <wsdl:message name="notFoundFault">
617     <wsdl:part name="fault" element="lrs:NotFoundFault"/>
618 </wsdl:message>
619 <wsdl:message name="unauthorizedFault">
620     <wsdl:part name="fault" element="lrs:UnauthorizedFault"/>
621 </wsdl:message>
622 <wsdl:portType name="ManageBusinessIdentifierServiceSoap">
623     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
624     <wsdl:operation name="Create">
625         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
626         <wsdl:input message="tns:createIn" />
627         <wsdl:output message="tns:createOut" />
628         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
629         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
630         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
631         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
632     </wsdl:operation>
633     <wsdl:operation name="Delete">
634         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
635         <wsdl:input message="tns:deleteIn" />
636         <wsdl:output message="tns:deleteOut" />
637         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
638         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
639         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
640         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
641     </wsdl:operation>
642     <wsdl:operation name="List">
643         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
644         <wsdl:input message="tns:listIn" />
645         <wsdl:output message="tns:listOut" />
646         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
647         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
648         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
649         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
650     </wsdl:operation>

```

```

651 <wsdl:operation name="PrepareToMigrate">
652   <wsdl:input message="tns:prepareMigrateIn"/>
653   <wsdl:output message="tns:prepareMigrateOut" />
654   <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
655   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
656   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
657   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
658 </wsdl:operation>
659 <wsdl:operation name="Migrate">
660   <wsdl:input message="tns:migrateIn"/>
661   <wsdl:output message="tns:migrateOut"/>
662   <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
663   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
664   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
665   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
666 </wsdl:operation>
667 <wsdl:operation name="CreateList">
668   <wsdl:input message="tns:createListIn"/>
669   <wsdl:output message="tns:createListOut" />
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="DeleteList">
676   <wsdl:input message="tns:deleteListIn"/>
677   <wsdl:output message="tns:deleteListOut" />
678   <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
679   <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
680   <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
681   <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
682 </wsdl:operation>
683 </wsdl:portType>
684 <wsdl:binding name="ManageBusinessIdentifierServiceSoap"
685   type="tns:ManageBusinessIdentifierServiceSoap">
686   <soap11:binding transport="http://schemas.xmlsoap.org/soap/http" />
687   <wsdl:operation name="Create">
688     <soap11:operation soapAction=
689       "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
690       :createIn"
691       style="document" />
692     <wsdl:input>
693       <soap11:body use="literal" />
694     </wsdl:input>
695     <wsdl:output>
696       <soap11:body use="literal" />
697     </wsdl:output>
698     <wsdl:fault name="UnauthorizedFault">
699       <soap:fault name="UnauthorizedFault" use="literal"/>
700     </wsdl:fault>
701     <wsdl:fault name="InternalErrorFault">
702       <soap:fault name="InternalErrorFault" use="literal"/>
703     </wsdl:fault>
704     <wsdl:fault name="BadRequestFault">
705       <soap:fault name="BadRequestFault" use="literal"/>
706     </wsdl:fault>
707   </wsdl:operation>
708   <wsdl:operation name="CreateList">
709     <soap11:operation soapAction=

```

```

710     "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
711     :createListIn"
712     style="document" />
713 <wsdl:input>
714     <soap11:body use="literal" />
715 </wsdl:input>
716 <wsdl:output>
717     <soap11:body use="literal" />
718 </wsdl:output>
719 <wsdl:fault name="NotFoundFault">
720     <soap:fault name="NotFoundFault" use="literal"/>
721 </wsdl:fault>
722 <wsdl:fault name="UnauthorizedFault">
723     <soap:fault name="UnauthorizedFault" use="literal"/>
724 </wsdl:fault>
725 <wsdl:fault name="InternalServerErrorFault">
726     <soap:fault name="InternalServerErrorFault" use="literal"/>
727 </wsdl:fault>
728 <wsdl:fault name="BadRequestFault">
729     <soap:fault name="BadRequestFault" use="literal"/>
730 </wsdl:fault>
731 </wsdl:operation>
732 <wsdl:operation name="Delete">
733     <soap11:operation soapAction=
734         "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
735         :deleteIn"
736         style="document" />
737 <wsdl:input>
738     <soap11:body use="literal" />
739 </wsdl:input>
740 <wsdl:output>
741     <soap11:body use="literal" />
742 </wsdl:output>
743 <wsdl:fault name="NotFoundFault">
744     <soap:fault name="NotFoundFault" use="literal"/>
745 </wsdl:fault>
746 <wsdl:fault name="UnauthorizedFault">
747     <soap:fault name="UnauthorizedFault" use="literal"/>
748 </wsdl:fault>
749 <wsdl:fault name="InternalServerErrorFault">
750     <soap:fault name="InternalServerErrorFault" use="literal"/>
751 </wsdl:fault>
752 <wsdl:fault name="BadRequestFault">
753     <soap:fault name="BadRequestFault" use="literal"/>
754 </wsdl:fault>
755 </wsdl:operation>
756 <wsdl:operation name="DeleteList">
757     <soap11:operation soapAction=
758         "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
759         :deleteListIn"
760         style="document" />
761 <wsdl:input>
762     <soap11:body use="literal" />
763 </wsdl:input>
764 <wsdl:output>
765     <soap11:body use="literal" />
766 </wsdl:output>
767 <wsdl:fault name="NotFoundFault">
768     <soap:fault name="NotFoundFault" use="literal"/>

```

```

769     </wsdl:fault>
770     <wsdl:fault name="UnauthorizedFault">
771       <soap:fault name="UnauthorizedFault" use="literal"/>
772     </wsdl:fault>
773     <wsdl:fault name="InternalServerErrorFault">
774       <soap:fault name="InternalServerErrorFault" use="literal"/>
775     </wsdl:fault>
776     <wsdl:fault name="BadRequestFault">
777       <soap:fault name="BadRequestFault" use="literal"/>
778     </wsdl:fault>
779   </wsdl:operation>
780   <wsdl:operation name="List">
781     <soap11:operation soapAction=
782       "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
783       :listIn"
784       style="document" />
785     <wsdl:input>
786       <soap11:body use="literal" />
787     </wsdl:input>
788     <wsdl:output>
789       <soap11:body use="literal" />
790     </wsdl:output>
791     <wsdl:fault name="NotFoundFault">
792       <soap:fault name="NotFoundFault" use="literal"/>
793     </wsdl:fault>
794     <wsdl:fault name="UnauthorizedFault">
795       <soap:fault name="UnauthorizedFault" use="literal"/>
796     </wsdl:fault>
797     <wsdl:fault name="InternalServerErrorFault">
798       <soap:fault name="InternalServerErrorFault" use="literal"/>
799     </wsdl:fault>
800     <wsdl:fault name="BadRequestFault">
801       <soap:fault name="BadRequestFault" use="literal"/>
802     </wsdl:fault>
803   </wsdl:operation>
804   <wsdl:operation name="PrepareToMigrate">
805     <soap11:operation soapAction=
806       "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
807       :prepareMigrateIn"
808       style="document" />
809     <wsdl:input>
810       <soap11:body use="literal" />
811     </wsdl:input>
812     <wsdl:output>
813       <soap11:body use="literal" />
814     </wsdl:output>
815     <wsdl:fault name="NotFoundFault">
816       <soap:fault name="NotFoundFault" use="literal"/>
817     </wsdl:fault>
818     <wsdl:fault name="UnauthorizedFault">
819       <soap:fault name="UnauthorizedFault" use="literal"/>
820     </wsdl:fault>
821     <wsdl:fault name="InternalServerErrorFault">
822       <soap:fault name="InternalServerErrorFault" use="literal"/>
823     </wsdl:fault>
824     <wsdl:fault name="BadRequestFault">
825       <soap:fault name="BadRequestFault" use="literal"/>
826     </wsdl:fault>
827   </wsdl:operation>

```

```

828     <wsdl:operation name="Migrate">
829         <soap11:operation soapAction=
830             "http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
831             :migrateIn"
832             style="document" />
833         <wsdl:input>
834             <soap11:body use="literal" />
835         </wsdl:input>
836         <wsdl:output>
837             <soap11:body use="literal" />
838         </wsdl:output>
839         <wsdl:fault name="NotFoundFault">
840             <soap:fault name="NotFoundFault" use="literal"/>
841         </wsdl:fault>
842         <wsdl:fault name="UnauthorizedFault">
843             <soap:fault name="UnauthorizedFault" use="literal"/>
844         </wsdl:fault>
845         <wsdl:fault name="InternalServerErrorFault">
846             <soap:fault name="InternalServerErrorFault" use="literal"/>
847         </wsdl:fault>
848         <wsdl:fault name="BadRequestFault">
849             <soap:fault name="BadRequestFault" use="literal"/>
850         </wsdl:fault>
851     </wsdl:operation>
852 </wsdl:binding>
853 </wsdl:definitions>

```

854 7.2 ManageServiceMetadataService.wsdl

```

855 <wsdl:definitions
856     xmlns:tns="http://busdox.org/serviceMetadata/
857     ManageServiceMetadataService/1.0/"
858     xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
859     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
860     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
861     xmlns:lrs="http://busdox.org/serviceMetadata/locator/1.0/"
862     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
863     xmlns:s="http://www.w3.org/2001/XMLSchema"
864     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
865     name="ManageServiceMetadataService"
866     targetNamespace=
867         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/"
868     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
869     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
870     <wsdl:types>
871         <s:schema elementFormDefault="qualified"
872             targetNamespace=
873                 "http://busdox.org/serviceMetadata/
874                 ManageServiceMetadataService/1.0/Schema/">
875             <s:import namespace="http://busdox.org/serviceMetadata/locator/1.0/"
876                 schemaLocation="ServiceMetadataLocatorTypes-0.95.xsd"/>
877         </s:schema>
878     </wsdl:types>
879     <wsdl:message name="createIn">
880         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
881         <wsdl:part name="messagePart"
882             element="lrs:CreateServiceMetadataPublisherService" />
883     </wsdl:message>
884     <wsdl:message name="createOut">

```

```

885     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
886 </wsdl:message>
887 <wsdl:message name="readIn">
888     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
889     <wsdl:part name="messagePart"
890         element="lrs:ReadServiceMetadataPublisherService" />
891 </wsdl:message>
892 <wsdl:message name="readOut">
893     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
894     <wsdl:part name="messagePart"
895         element="lrs:SignedServiceMetadataPublisherService" />
896 </wsdl:message>
897 <wsdl:message name="updateIn">
898     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
899     <wsdl:part name="messagePart"
900         element="lrs:UpdateServiceMetadataPublisherService" />
901 </wsdl:message>
902 <wsdl:message name="updateOut">
903     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
904 </wsdl:message>
905 <wsdl:message name="deleteIn">
906     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
907     <wsdl:part name="messagePart"
908         element="lrs>DeleteServiceMetadataPublisherService" />
909 </wsdl:message>
910 <wsdl:message name="deleteOut">
911     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
912 </wsdl:message>
913 <wsdl:message name="badRequestFault">
914     <wsdl:part name="fault" element="lrs:BadRequestFault"/>
915 </wsdl:message>
916 <wsdl:message name="internalErrorFault">
917     <wsdl:part name="fault" element="lrs:InternalErrorFault"/>
918 </wsdl:message>
919 <wsdl:message name="notFoundFault">
920     <wsdl:part name="fault" element="lrs:NotFoundFault"/>
921 </wsdl:message>
922 <wsdl:message name="unauthorizedFault">
923     <wsdl:part name="fault" element="lrs:UnauthorizedFault"/>
924 </wsdl:message>
925 <wsdl:portType name="ManageServiceMetadataServiceSoap">
926     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
927     <wsdl:operation name="Create">
928         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
929         <wsdl:input message="tns:createIn" />
930         <wsdl:output message="tns:createOut" />
931         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
932         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
933         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
934     </wsdl:operation>
935     <wsdl:operation name="Read">
936         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
937         <wsdl:input message="tns:readIn" />
938         <wsdl:output message="tns:readOut" />
939         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
940         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
941         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
942         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
943     </wsdl:operation>

```

```

944     <wsdl:operation name="Update">
945         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
946         <wsdl:input message="tns:updateIn" />
947         <wsdl:output message="tns:updateOut" />
948         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
949         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
950         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
951         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
952     </wsdl:operation>
953     <wsdl:operation name="Delete">
954         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
955         <wsdl:input message="tns:deleteIn" />
956         <wsdl:output message="tns:deleteOut" />
957         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
958         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
959         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
960         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
961     </wsdl:operation>
962 </wsdl:portType>
963 <wsdl:binding name="ManageServiceMetadataServiceSoap"
964     type="tns:ManageServiceMetadataServiceSoap">
965     <soap11:binding transport="http://schemas.xmlsoap.org/soap/http" />
966     <wsdl:operation name="Create">
967         <soap11:operation soapAction=
968             "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
969             :createIn"
970             style="document" />
971         <wsdl:input>
972             <soap11:body use="literal" />
973         </wsdl:input>
974         <wsdl:output>
975             <soap11:body use="literal" />
976         </wsdl:output>
977         <wsdl:fault name="UnauthorizedFault">
978             <soap:fault name="UnauthorizedFault" use="literal"/>
979         </wsdl:fault>
980         <wsdl:fault name="InternalErrorFault">
981             <soap:fault name="InternalErrorFault" use="literal"/>
982         </wsdl:fault>
983         <wsdl:fault name="BadRequestFault">
984             <soap:fault name="BadRequestFault" use="literal"/>
985         </wsdl:fault>
986     </wsdl:operation>
987     <wsdl:operation name="Read">
988         <soap11:operation soapAction=
989             "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
990             :readIn"
991             style="document" />
992         <wsdl:input>
993             <soap11:body use="literal" />
994         </wsdl:input>
995         <wsdl:output>
996             <soap11:body use="literal" />
997         </wsdl:output>
998         <wsdl:fault name="NotFoundFault">
999             <soap:fault name="NotFoundFault" use="literal"/>
1000     </wsdl:fault>
1001     <wsdl:fault name="UnauthorizedFault">
1002         <soap:fault name="UnauthorizedFault" use="literal"/>

```

```

1003     </wsdl:fault>
1004     <wsdl:fault name="InternalServerErrorFault">
1005         <soap:fault name="InternalServerErrorFault" use="literal"/>
1006     </wsdl:fault>
1007     <wsdl:fault name="BadRequestFault">
1008         <soap:fault name="BadRequestFault" use="literal"/>
1009     </wsdl:fault>
1010 </wsdl:operation>
1011     <wsdl:operation name="Update">
1012     <soap11:operation soapAction=
1013         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1014         :updateIn"
1015         style="document" />
1016     <wsdl:input>
1017         <soap11:body use="literal" />
1018     </wsdl:input>
1019     <wsdl:output>
1020         <soap11:body use="literal" />
1021     </wsdl:output>
1022     <wsdl:fault name="NotFoundFault">
1023         <soap:fault name="NotFoundFault" use="literal"/>
1024     </wsdl:fault>
1025     <wsdl:fault name="UnauthorizedFault">
1026         <soap:fault name="UnauthorizedFault" use="literal"/>
1027     </wsdl:fault>
1028     <wsdl:fault name="InternalServerErrorFault">
1029         <soap:fault name="InternalServerErrorFault" use="literal"/>
1030     </wsdl:fault>
1031     <wsdl:fault name="BadRequestFault">
1032         <soap:fault name="BadRequestFault" use="literal"/>
1033     </wsdl:fault>
1034 </wsdl:operation>
1035 <wsdl:operation name="Delete">
1036     <soap11:operation soapAction=
1037         "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/
1038         :deleteIn"
1039         style="document" />
1040     <wsdl:input>
1041         <soap11:body use="literal" />
1042     </wsdl:input>
1043     <wsdl:output>
1044         <soap11:body use="literal" />
1045     </wsdl:output>
1046     <wsdl:fault name="NotFoundFault">
1047         <soap:fault name="NotFoundFault" use="literal"/>
1048     </wsdl:fault>
1049     <wsdl:fault name="UnauthorizedFault">
1050         <soap:fault name="UnauthorizedFault" use="literal"/>
1051     </wsdl:fault>
1052     <wsdl:fault name="InternalServerErrorFault">
1053         <soap:fault name="InternalServerErrorFault" use="literal"/>
1054     </wsdl:fault>
1055     <wsdl:fault name="BadRequestFault">
1056         <soap:fault name="BadRequestFault" use="literal"/>
1057     </wsdl:fault>
1058 </wsdl:operation>
1059 </wsdl:binding>
1060 </wsdl:definitions>

```