



# Specification



## OpenPeppol AISBL



## Peppol Transport Infrastructure ICT - Models

## Service Metadata Locator (SML)



**Version: 1.2.0**  
**Status: In use**



### Editors:

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

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



## Revision History

Version	Date	Description of changes	Author
1.0.0	2010-02-15	First version (pending EC approval)	Mike Edwards, NITA/IBM
1.0.1	2010-10-01	EC approved	Klaus Vilstrup Pedersen, DIFI
1.2.0	2021-05-13	Updated the references Improved layout Linking external XSD and WSDLs in the Appendix Updated rules for migration key Changed the service name from “ManageParticipant*” to “ManageBusiness*” to reflect the current situation	Philip Helger, OpenPeppol OO

### Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise.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

### Statement of copyright



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

*You are free to:*

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

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

## Contributors

### Organisations

DIFI (Direktoratet for forvaltning og IKT)<sup>1</sup>, Norway, [www.difi.no](http://www.difi.no)

NITA (IT- og Telestyrelsen)<sup>2</sup>, Denmark, [www.itst.dk](http://www.itst.dk)

BRZ (Bundesrechenzentrum)<sup>3</sup>, Austria, [www.brz.gv.at](http://www.brz.gv.at)

Consip, Italy

OpenPeppol

### Persons

Bergthór Skúlason, NITA

Carl-Markus Piswanger, BRZ

Christian Uldall Pedersen, NITA/Accenture

Dennis Jensen Søgaard, NITA/Accenture

Gert Sylvest, NITA/Avanade

Hans Guldager Knudsen, NITA/Lenio

Jens Jakob Andersen, NITA

Joakim Recht, NITA/Trifork

Kenneth Bengtsson, NITA/Alfa1lab

Klaus Vilstrup Pedersen, DIFI

Mike Edwards, NITA/IBM (editor)

Mikkel Hippe Brun, NITA

Paul Fremantle, NITA/WSO2

Philip Helger, BRZ/ OpenPeppol Operating Office

Thomas Gundel, NITA/IT Crew

---

<sup>1</sup> English: Agency for Public Management and eGovernment

<sup>2</sup> English: National IT- and Telecom Agency

<sup>3</sup> English: Austrian Federal Computing Centre

---

## Table of contents

Contributors .....	4
Table of contents.....	5
1 Introduction .....	6
1.1 Objective .....	6
1.2 Scope .....	6
1.3 Goals and non-goals .....	6
1.4 Terminology.....	7
1.4.1 Notational conventions .....	7
1.4.2 Normative references.....	7
1.4.3 Non-normative references .....	7
1.5 Namespaces .....	8
2 The Service Discovery Process .....	9
2.1 Discovery flow .....	9
2.2 Flows Relating to Service Metadata Publishers .....	10
3 Interfaces and Data Model .....	14
3.1 Service Metadata Locator Service, logical interface .....	14
3.1.1 Format of Participant Identifiers .....	14
3.1.2 ManageBusinessIdentifier interface .....	14
3.1.3 ManageServiceMetadata interface .....	18
3.1.4 Fault Descriptions.....	20
3.2 Service Metadata Locator - data model.....	21
3.2.1 ServiceMetadataPublisherService datatype .....	21
3.2.2 ServiceMetadataPublisherServiceForParticipant datatype .....	21
3.2.3 ParticipantIdentifier datatype .....	21
3.2.4 ParticipantIdentifier format .....	22
3.2.5 ParticipantIdentifierPage datatype .....	22
3.2.6 MigrationRecord.....	22
4 Service Bindings .....	24
4.1 Services Provided as Web services - characteristics .....	24
4.2 ManageBusinessIdentifier service - binding.....	24
4.2.1 Transport binding .....	24
4.2.2 Security .....	24
4.3 ManageServiceMetadata service - binding.....	24
4.3.1 Transport binding .....	24
4.3.2 Security .....	24
5 DNS Spoof Mitigation.....	25
6 Appendix A: XML Schema (non-normative).....	26
6.1 peppol-sml-types-v1.xsd .....	26
7 Appendix B: WSDLs (non-normative) .....	28
7.1 peppol-sml-manage-business-identifier-service-v1.wsdl .....	28
7.2 peppol-sml-manage-service-metadata-service-v1.wsdl .....	33

# 1 Introduction

## 1.1 Objective

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

The Service Metadata Locator service exposes three interfaces:

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

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

## 1.2 Scope

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

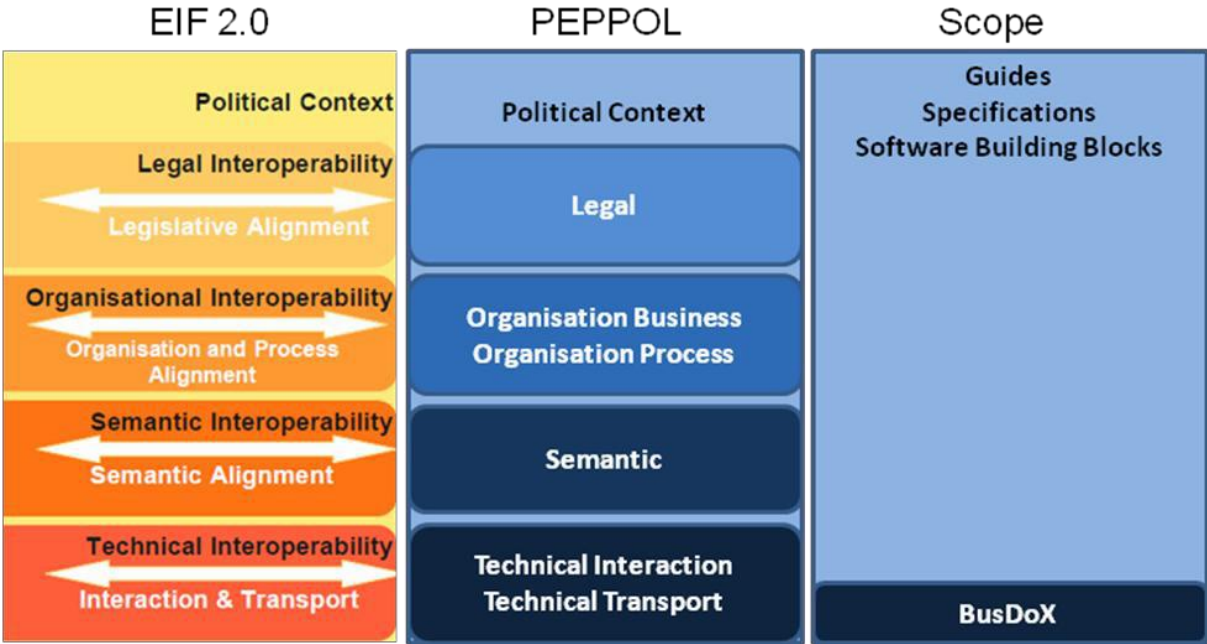


Fig. 1: Peppol Interoperability

## 1.3 Goals and non-goals

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

## 26 1.4 Terminology

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

### 30 1.4.1 Notational conventions

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

```
38 <!-- sample pseudo-schema -->  
39 <defined_element  
40     required_attribute_of_type_string="xs:string"  
41     optional_attribute_of_type_int="xs:int"? >  
42   <required_element />  
43   <optional_element />?  
44   <one_or_more_of_these_elements />+  
45   [ <choice_1 /> | <choice_2 /> ]*  
46 </defined_element>
```

### 47 1.4.2 Normative references

- 48 [BDEN-SMP] "Peppol Service Metadata Publishing (SMP) 1.2.0",  
49 [https://docs.peppol.eu/edelivery/smp/PEPPOL-EDN-Service-Metadata-Publishing-  
50 1.2.0-2021-02-24.pdf](https://docs.peppol.eu/edelivery/smp/PEPPOL-EDN-Service-Metadata-Publishing-1.2.0-2021-02-24.pdf)
- 51 [XML-DSIG] "XML Signature Syntax and Processing (Second Edition)",  
52 <http://www.w3.org/TR/xmlsig-core/>
- 53 [RFC-2119] "Key words for use in RFCs to Indicate Requirement Levels",  
54 <http://www.ietf.org/rfc/rfc2119.txt>
- 55 [RFC3986] "Uniform Resource Identifier (URI): Generic Syntax",  
56 <http://tools.ietf.org/html/rfc3986>
- 57 [PFUOI4] "Peppol Policy for use of Identifiers 4.1.0",  
58 [https://docs.peppol.eu/edelivery/policies/PEPPOL-EDN-Policy-for-use-of-identifiers-  
59 4.1.0-2020-03-11.pdf](https://docs.peppol.eu/edelivery/policies/PEPPOL-EDN-Policy-for-use-of-identifiers-4.1.0-2020-03-11.pdf)

### 60 1.4.3 Non-normative references

- 61 [WSDL-2.0] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language",  
62 <http://www.w3.org/TR/wsdl20/>
- 63 [WS-I BP] "WS-I Basic Profile Version 1.1",  
64 <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
- 65 [WS-I BSP] "WS-I Basic Security Profile Version 1.0",  
66 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
- 67 [DNS-1034] "Domain Names - Concepts and Facilities",  
68 <http://tools.ietf.org/html/rfc1034>

- 69 [DNS-1035] “Domain Names - Implementation and Specification”,  
70 <http://tools.ietf.org/html/rfc1035>
- 71 [MD5] “The MD5 Message-Digest Algorithm”,  
72 <http://tools.ietf.org/html/rfc1321>

## 73 1.5 Namespaces

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

Prefix	Namespace URI
ids	<a href="http://busdox.org/transport/identifiers/1.0/">http://busdox.org/transport/identifiers/1.0/</a>
lrs	<a href="http://busdox.org/serviceMetadata/locator/1.0/">http://busdox.org/serviceMetadata/locator/1.0/</a>
soap	<a href="http://schemas.xmlsoap.org/wsdl/soap/">http://schemas.xmlsoap.org/wsdl/soap/</a>
wsdl	<a href="http://schemas.xmlsoap.org/wsdl/">http://schemas.xmlsoap.org/wsdl/</a>
xs	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>



## 76 2 The Service Discovery Process

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

- 80 • Service Metadata Locator:
  - 81 ○ Discovery interface for senders
  - 82 ○ Management interface for SMPs
- 83 • Service Metadata Publishers:
  - 84 ○ Discovery interface for senders

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

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

### 92 2.1 Discovery flow

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

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

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

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

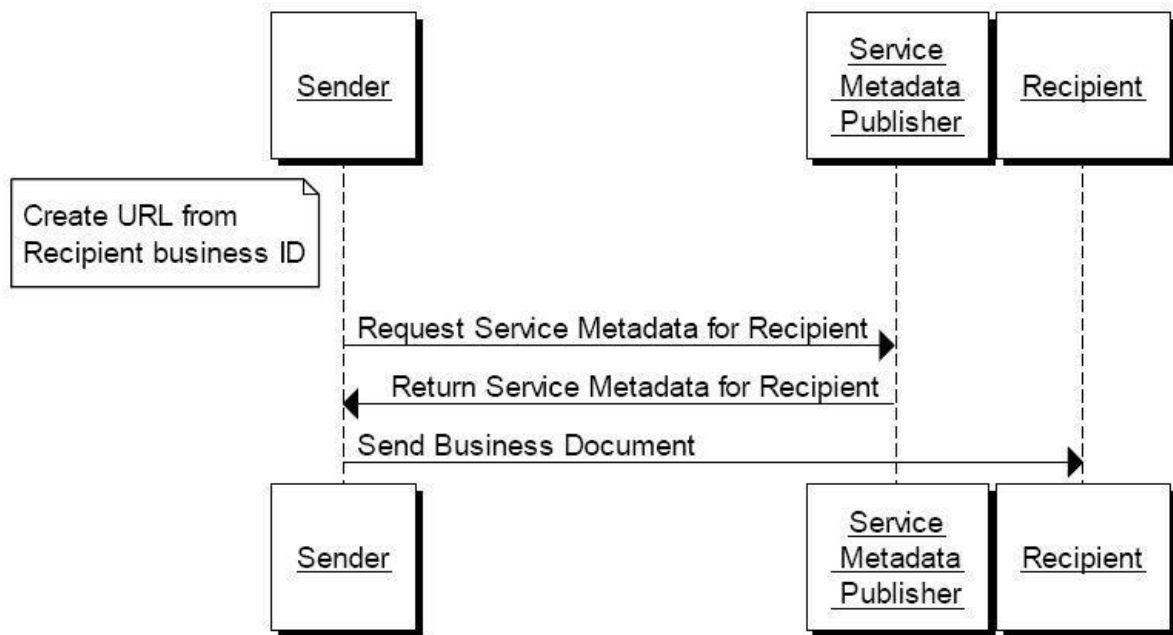


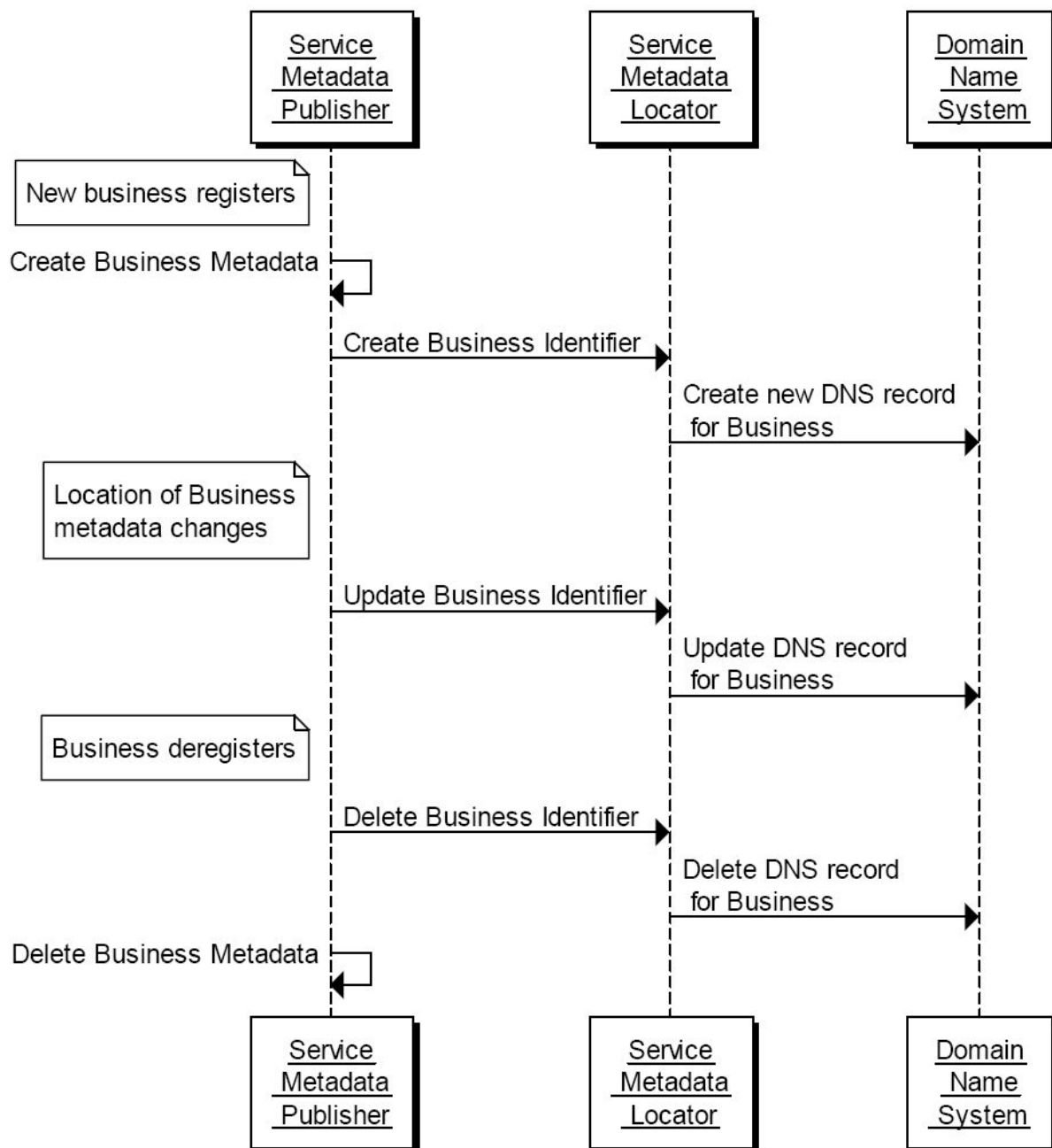
Fig. 2: Sequence Diagram for Sender transmitting Document to Recipient

110  
111

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

## 120 2.2 Flows Relating to Service Metadata Publishers

121 The management of the DNS CNAME records for a given participant identifier is performed through  
 122 the Management interface of the Service Metadata Locator. The management interface is primarily  
 123 for use by the Service Metadata Publisher which controls the service metadata for a given participant  
 124 identifier. Note that the DNS CNAME records are **not** manipulated directly by the Service Metadata  
 125 Publisher, but are manipulated by the Service Metadata Locator service following requests made to  
 126 its Management interface. The basic process steps for the SMP to manipulate the metadata relating  
 127 to a given participant are shown in Fig. 3.



128

129

**Fig. 3: Sequence Diagram for Service Metadata Publisher Adding, Updating and Removing Metadata for a Participant**

130

131

132

133

134

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

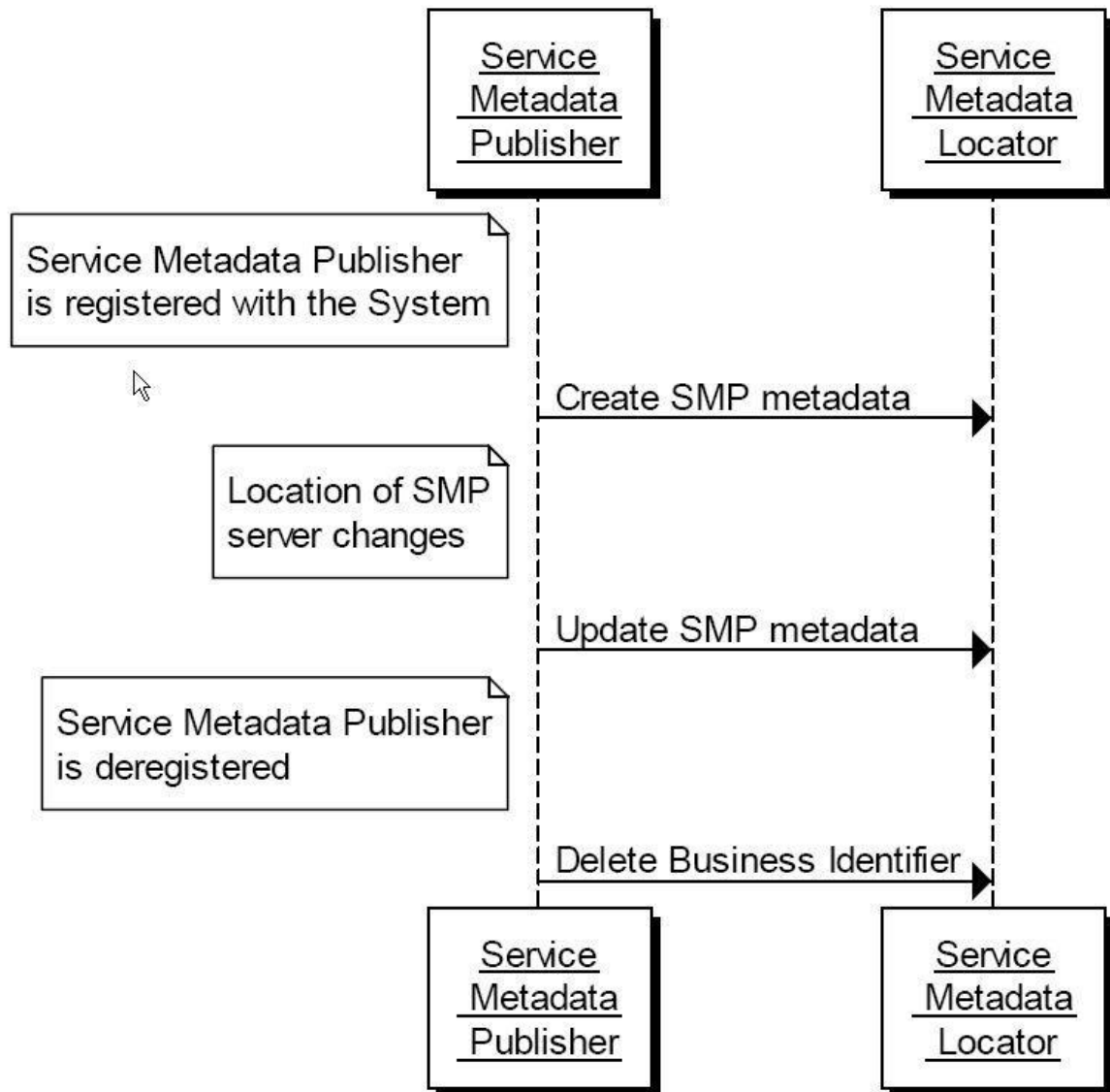


Fig. 4: Service Metadata Publisher use of the ManageServiceMetadata

135  
136

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

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

150 The following rules apply to the Migration Key

- 151
- MUST have at least 8 characters and not more than 24 characters

- 152 • MUST contain at least 2 lower case characters (a-z)
- 153 • MUST contain at least 2 upper case characters (A-Z)
- 154 • MUST contain at least 2 digits (0-9)
- 155 • MUST contain at least 2 characters from this set: "@" (ASCII code 64), "#" (35), "\$" (36), "%" (37), "(" (40), ")" (41), "[" (91), "]" (93), "{" (123), "}" (125), "\*" (42), "^" (94), "-" (45), "!" (33), "~" (126), "|" (124), "+" (43) and "=" (61)
- 156
- 157
- 158 • MUST NOT contain whitespace characters

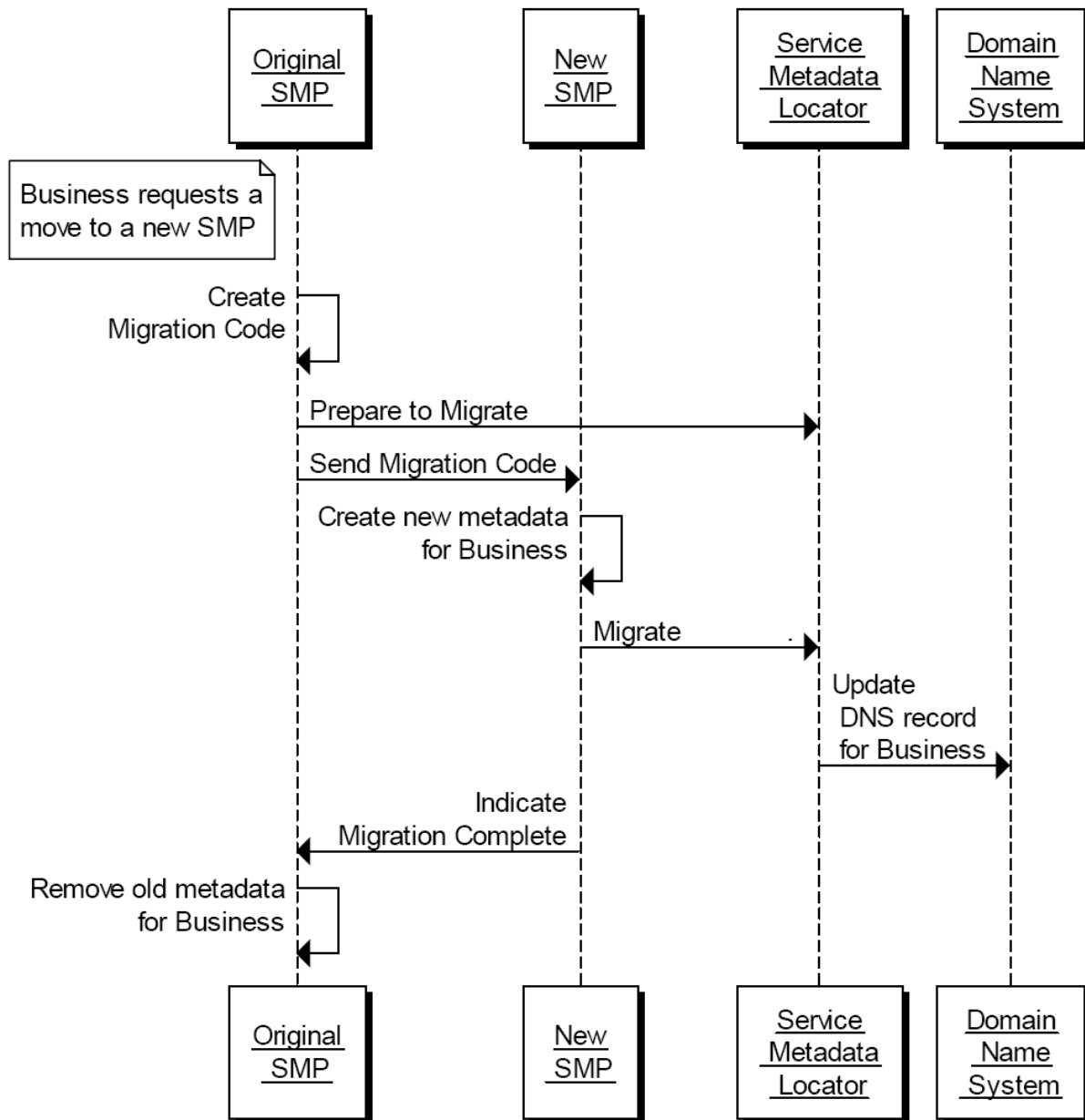


Fig. 5: Steps in Migrating Metadata for a Participant from one SMP to a new SMP

## 161 3 Interfaces and Data Model

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

### 163 3.1 Service Metadata Locator Service, logical interface

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

- 165 • Manage participant identifiers interface
- 166 This is the interface for Service Metadata Publishers for managing the registered participant
- 167 identifiers they expose.
- 168 • Manage service metadata interface
- 169 This is the interface for Service Metadata Publishers for managing the metadata about their
- 170 metadata publishing service, e.g. binding, interface profile and key information.

#### 171 3.1.1 Format of Participant Identifiers

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

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

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

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

187 An example participant ID is `0010:5798000000001`, for which the MD5 hash is  
188 `e49b223851f6e97cbfce4f72c3402aac`. See POLICY 7 of the [PFUOI4] for details.

#### 189 3.1.2 ManageBusinessIdentifier interface

190 The ManageBusinessIdentifier interface allows Service Metadata Publishers to manage the  
191 information in the Service Metadata Locator Service relating to individual participant identifiers for  
192 which they hold metadata.

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

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

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

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

201 This implies that all participant identifiers for that `schemeID` will have addresses that resolve to the  
202 single address of that one SMP - and that as result only one SMP can handle the metadata for all  
203 participant identifiers of that scheme. Wildcard records are indicated through the use of "\*" as the  
204 participant identifier in the operations of the `ManageBusinessIdentifier` interface.

205 The `ManageBusinessIdentifier` interface has the following operations:

- 206 • `Create`
- 207 • `CreateList`
- 208 • `Delete`
- 209 • `DeleteList`
- 210 • `PrepareToMigrate`
- 211 • `Migrate`
- 212 • `List`

### 213 **Create()**

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

- 218 • Input `CreateParticipantIdentifier`:  
219 `ServiceMetadataPublisherServiceForParticipantType`  
220 contains the Participant Identifier for a given participant and the identifier of the SMP which  
221 holds its data
- 222 • `Fault: notFoundFault`  
223 returned if the identifier of the SMP could not be found
- 224 • `Fault: unauthorizedFault`  
225 returned if the caller is not authorized to invoke the `Create` operation
- 226 • `Fault: badRequestFault`  
227 returned if the supplied `CreateParticipantIdentifier` does not contain consistent  
228 data
- 229 • `Fault: internalErrorFault`  
230 returned if the SML service is unable to process the request for any reason

### 231 **CreateList()**

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

- 236 • Input `CreateList`: `ParticipantIdentifierPage`  
237 contains the list of Participant Identifiers for the participants which are added to the Service  
238 Metadata Locator Service. The `NextPageIdentifier` element is absent.
- 239 • `Fault: notFoundFault`  
240 returned if the identifier of the SMP could not be found

241       • `Fault: unauthorizedFault`  
242       returned if the caller is not authorized to invoke the `CreateList` operation

243       • `Fault: badRequestFault`  
244       returned if the supplied `CreateList` does not contain consistent data

245       • `Fault: internalErrorFault`  
246       returned if the SML service is unable to process the request for any reason

#### 247 **Delete()**

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

249       • Input `DeleteParticipantIdentifier`:  
250       `ServiceMetadataPublisherServiceForParticipantType`  
251       contains the Participant Identifier for a given participant and the identifier of the SMP that  
252       publishes its metadata

253       • `Fault: notFoundFault`  
254       returned if the participant identifier or the identifier of the SMP could not be found

255       • `Fault: unauthorizedFault`  
256       returned if the caller is not authorized to invoke the `Delete` operation

257       • `Fault: badRequestFault`  
258       returned if the supplied `DeleteParticipantIdentifier` does not contain consistent  
259       data

260       • `Fault: internalErrorFault`  
261       returned if the SML service is unable to process the request for any reason

#### 262 **DeleteList()**

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

264       • Input `DeleteList: ParticipantIdentifier`  
265       contains the list of Participant Identifiers for the participants which are removed from the  
266       Service Metadata Locator Service. The `NextPageIdentifier` element is absent.

267       • `Fault: notFoundFault`  
268       returned if one or more participant identifiers or the identifier of the SMP could not be found

269       • `Fault: unauthorizedFault`  
270       returned if the caller is not authorized to invoke the `DeleteList` operation

271       • `Fault: badRequestFault`  
272       returned if the supplied `DeleteList` does not contain consistent data

273       • `Fault: internalErrorFault`  
274       returned if the SML service is unable to process the request for any reason

#### 275 **PrepareToMigrate()**

276 Prepares a Participant Identifier for migration to a new Service Metadata Publisher. This operation is  
277 called by the Service Metadata Publisher which currently publishes the metadata for the Participant  
278 Identifier. The Service Metadata Publisher supplies a Migration Code which is used to control the  
279 migration process. The Migration Code must be passed (out of band) to the Service Metadata  
280 Publisher which is taking over the publishing of the metadata for the Participant Identifier and which  
281 MUST be used on the invocation of the `Migrate()` operation.



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

284     • Input `PrepareMigrationRecord`: `MigrationRecordType`  
285         contains the Migration Key and the Participant Identifier which is about to be migrated from  
286         one Service Metadata Publisher to another.

287     • Fault: `notFoundFault`  
288         returned if the participant identifier or the identifier of the SMP could not be found

289     • Fault: `unauthorizedFault`  
290         returned if the caller is not authorized to invoke the `PrepareToMigrate` operation

291     • Fault: `badRequestFault`  
292         returned if the supplied `PrepateMigrationRecord` does not contain consistent data

293     • Fault: `internalErrorFault`  
294         returned if the SML service is unable to process the request for any reason

### 295 **Migrate()**

296 Migrates a Participant Identifier already held by the Service Metadata Locator Service to target a new  
297 Service Metadata Publisher. This operation is called by the Service Metadata Publisher which is  
298 taking over the publishing for the Participant Identifier. The operation requires the new Service  
299 Metadata Publisher to provide a migration code which was originally obtained from the old Service  
300 Metadata Publisher.

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

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

306     • Input `CompleteMigrationRecord`: `MigrationRecordType`  
307         contains the Migration Key and the Participant Identifier which is to be migrated from one  
308         Service Metadata Publisher to another.

309     • Fault: `notFoundFault`  
310         returned if the migration key or the identifier of the SMP could not be found

311     • Fault: `unauthorizedFault`  
312         returned if the caller is not authorized to invoke the `Migrate` operation

313     • Fault: `badRequestFault`  
314         returned if the supplied `CompleteMigrationRecord` does not contain consistent data

315     • Fault: `internalErrorFault`  
316         returned if the SML service is unable to process the request for any reason

### 317 **List()**

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

321     • Input `Page`: `PageRequest`  
322         contains a `PageRequest` containing the `ServiceMetadataPublisherID` of the SMP

- 323 and (if required) an identifier representing the next page of data to retrieve. If the  
324 `NextPageIdentifier` is absent, the first page is returned.
- 325 • Output: `ParticipantIdentifierPage`  
326 a page of Participant Identifier entries associated with the Service Metadata Publisher, also  
327 containing a `<Page/>` element containing the identifier that represents the next page, if  
328 any.
  - 329 • Fault: `notFoundFault`  
330 returned if the next page or the identifier of the SMP could not be found
  - 331 • Fault: `unauthorizedFault`  
332 returned if the caller is not authorized to invoke the `List` operation
  - 333 • Fault: `badRequestFault`  
334 returned if the supplied `NextPage` does not contain consistent data
  - 335 • Fault: `internalErrorFault`  
336 returned if the SML service is unable to process the request for any reason
- 337 Note that the underlying data may be updated between one invocation of `List()` and a  
338 subsequent invocation of `List()`, so that a set of retrieved pages of participant identifiers may not  
339 represent a consistent set of data.

### 340 3.1.3 **ManageServiceMetadata interface**

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

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

347 The `ManageServiceMetadata` interface has the following operations:

- 348 • Create
- 349 • Read
- 350 • Update
- 351 • Delete

#### 352 **Create()**

353 Establishes a Service Metadata Publisher metadata record, containing the metadata about the  
354 Service Metadata Publisher, as outlined in the `ServiceMetadataPublisherService` data  
355 type.

- 356 • Input `CreateServiceMetadataPublisherService`:  
357 `ServiceMetadataPublisherService`  
358 contains the service metadata publisher information, which includes the logical and physical  
359 addresses for the SMP (Domain name and IP address). It is assumed that the  
360 `ServiceMetadataPublisherID` has been assigned to the calling user out-of-bands.
- 361 • Fault: `unauthorizedFault`  
362 returned if the caller is not authorized to invoke the `Create` operation

363       • `Fault: badRequestFault`  
364        returned if the supplied `CreateServiceMetadataPublisherService` does not  
365        contain consistent data

366       • `Fault: internalErrorFault`  
367        returned if the SML service is unable to process the request for any reason

#### 368 **Read()**

369 Retrieves the Service Metadata Publisher record for the Service Metadata Publisher.

370       • Input `ReadServiceMetadataPublisherService:`  
371        `ServiceMetadataPublisherID`  
372        the unique ID of the Service Metadata Publisher for which the record is required

373       • Output: `ServiceMetadataPublisherService`  
374        the service metadata publisher record, in the form of a  
375        `ServiceMetadataPublisherService` data type

376       • `Fault: notFoundFault`  
377        returned if the identifier of the SMP could not be found

378       • `Fault: unauthorizedFault`  
379        returned if the caller is not authorized to invoke the `Read` operation

380       • `Fault: badRequestFault`  
381        returned if the supplied parameter does not contain consistent data

382       • `Fault: internalErrorFault`  
383        returned if the SML service is unable to process the request for any reason

#### 384 **Update()**

385 Updates the Service Metadata Publisher record for the Service Metadata Publisher

386       • Input `UpdateServiceMetadataPublisherService:`  
387        `ServiceMetadataPublisherService`  
388        contains the service metadata for the service metadata publisher, which includes the logical  
389        and physical addresses for the SMP (Domain name and IP address)

390       • `Fault: notFoundFault`  
391        returned if the identifier of the SMP could not be found

392       • `Fault: unauthorizedFault`  
393        returned if the caller is not authorized to invoke the `Update` operation

394       • `Fault: badRequestFault`  
395        returned if the supplied `UpdateServiceMetadataPublisherService` does not  
396        contain consistent data

397       • `Fault: internalErrorFault`  
398        returned if the SML service is unable to process the request for any reason

#### 399 **Delete()**

400 Deletes the Service Metadata Publisher record for the Service Metadata Publisher

401       • Input `DeleteServiceMetadataPublisherService:`  
402        `ServiceMetadataPublisherID`  
403        the unique ID of the Service Metadata Publisher to delete

- 404       • `Fault: notFoundFault`
- 405       returned if the identifier of the SMP could not be found
- 406       • `Fault: unauthorizedFault`
- 407       returned if the caller is not authorized to invoke the `Delete` operation
- 408       • `Fault: badRequestFault`
- 409       returned if the supplied `DeleteServiceMetadataPublisherService` does not
- 410       contain consistent data
- 411       • `Fault: internalErrorFault`
- 412       returned if the SML service is unable to process the request for any reason

413   **3.1.4 Fault Descriptions**

414   **SMP Not Found Fault**

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

415   **Unauthorized Fault**

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

416   **Bad Request Fault**

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

417   **Internal Error Fault**

[action]	<a href="http://busdox.org/2010/02/locator/fault">http://busdox.org/2010/02/locator/fault</a>
Code	Sender
Subcode	internalErrorFault
Reason	The SML encountered an error while processing the request
Detail	As detailed by the SML

## 418 3.2 Service Metadata Locator - data model

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

- 420 • ServiceMetadataPublisher
- 421 • RecipientParticipantIdentifier
- 422 • ParticipantIdentifierPage
- 423 • MigrationRecord

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

### 425 3.2.1 ServiceMetadataPublisherService datatype

426 Represents a Metadata Publisher Service.

```
427 <ServiceMetadataPublisherService>
428   <PublisherEndpoint>
429     <EndpointAddress/>
430   </PublisherEndpoint>
431   <ServiceMetadataPublisherID/>
432 </ServiceMetadataPublisherService>
```

433 ServiceMetadataPublisherService has the following sub-elements:

- 434 • PublisherEndpoint (1..1) : PublisherEndpointType  
435 the technical endpoint address of the Service Metadata Publisher, which can be used to  
436 query information about particular participant identifiers. ServiceEndpointList is a type  
437 defined in the ServiceMetadataPublishingTypes Schema. The PublisherEndpoint  
438 element may be a domain name or an IP address of the SMP, or a wildcard expression based  
439 on the domain name.
- 440 • ServiceMetadataPublisherID (1..1) : xs:string  
441 holds the Unique Identifier of the SMP. When creating a  
442 ServiceMetadataPublisherService record, it is assumed that the publisher ID has  
443 been obtained out of band.

### 444 3.2.2 ServiceMetadataPublisherServiceForParticipant datatype

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

```
447 <ServiceMetadataPublisherServiceForParticipant>
448   <ServiceMetadataPublisherID/>
449   <ids:ParticipantIdentifier/>
450 </ServiceMetadataPublisherServiceForParticipant>
```

451 ServiceMetadataPublisherService has the following subelements:

- 452 • ServiceMetadataPublisherID (1..1) : xs:string  
453 holds the Unique Identifier of the SMP.
- 454 • ParticipantIdentifier (1..1) : ids:ParticipantIdentifierType  
455 the Participant Identifier which has its services registered in the Service Metadata Publisher.  
456 See the “ParticipantIdentifier” section on the format.

### 457 3.2.3 ParticipantIdentifier datatype

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

---

```

460 <ids:ParticipantIdentifier scheme="xs:string">
461   xs:string
462 </ids:ParticipantIdentifier>
    
```

463 ParticipantIdentifier has the following sub elements:

- 464 • ParticipantIdentifier (1..1): xs:string  
465 the participant identifier
- 466 • @scheme (1..1): xs:string  
467 the format scheme of the participant identifier

### 468 3.2.4 ParticipantIdentifier format

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

### 471 3.2.5 ParticipantIdentifierPage datatype

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

```

474 <ParticipantIdentifierPage>
475   <ServiceMetadataPublisherID/>
476   <ParticipantIdentifier/>*
477   <NextPageIdentifier/?>
478 </ParticipantIdentifierPage>
    
```

- 479 • ServiceMetadataPublisherID (1..1) : xs:string  
480 holds the Unique Identifier of the SMP
- 481 • ids:ParticipantIdentifier (1..1): xs:string  
482 the participant identifier
- 483 • NextPageIdentifier (0..1): xs:string  
484 an element containing a string identifying the next page of ParticipantIdentifiers:

```

485 <NextPageIdentifier>
486   [ Identifier for_Next_Page ]
487 </NextPageIdentifier>
    
```

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

### 489 3.2.6 MigrationRecord

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

```

493 <MigrationRecord>
494   <ServiceMetadataPublisherID/>
495   <ParticipantIdentifier/>*
496   <MigrationKey/?>
497 </MigrationRecord>
    
```

498 MigrationRecord has the following sub elements:

- 499 • ServiceMetadataPublisherID (1..1) : xs:string  
500 holds the Unique Identifier of the SMP.

- 501 • `ParticipantIdentifier (1..1)` : `ids:ParticipantIdentifierType`  
502 the participant identifier
  
- 503 • `MigrationKey (1..1)` : `xs:string`  
504 a string which is a unique key controlling the migration of the metadata for a given  
505 `ParticipantIdentifier` from one Service Metadata Publisher to another. The  
506 `MigrationKey` string is a string of characters and numbers only, with a maximum length  
507 of 24 characters.

## 508 **4 Service Bindings**

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

### 511 **4.1 Services Provided as Web services - characteristics**

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

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

### 515 **4.2 ManageBusinessIdentifier service - binding**

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

#### 517 **4.2.1 Transport binding**

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

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

#### 520 **4.2.2 Security**

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

### 526 **4.3 ManageServiceMetadata service - binding**

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

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

#### 530 **4.3.1 Transport binding**

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

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

#### 533 **4.3.2 Security**

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



## 536 **5 DNS Spoof Mitigation**

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

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

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

## 550 6 Appendix A: XML Schema (non-normative)

551 This section defines the XML Schema types used in the interfaces. The normative version of the file is  
552 published together with this specification.

### 553 6.1 peppol-sml-types-v1.xsd

```

554 <?xml version="1.0" encoding="utf-8"?>
555 <xs:schema id="ServiceMetadataPublisherService"
556           targetNamespace="http://busdox.org/serviceMetadata/Locator/1.0/"
557           elementFormDefault="qualified"
558           xmlns="http://busdox.org/serviceMetadata/Locator/1.0/"
559           xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
560           xmlns:xs="http://www.w3.org/2001/XMLSchema">
561   <xs:import schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
562 wss-wssecurity-utility-1.0.xsd"
563             namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
564 wsswssecurity-utility-1.0.xsd"/>
565   <xs:import schemaLocation="ws-addr.xsd"
566             namespace="http://www.w3.org/2005/08/addressing"/>
567   <xs:import schemaLocation="peppol-identifiers-v1.xsd"
568             namespace="http://busdox.org/transport/identifiers/1.0"/>
569
570   <xs:element name="ServiceMetadataPublisherID" type="xs:string"/>
571   <xs:element name="CreateServiceMetadataPublisherService"
572 type="ServiceMetadataPublisherServiceType"/>
573   <xs:element name="ReadServiceMetadataPublisherService"
574 type="ServiceMetadataPublisherIdentifierType"/>
575   <xs:element name="UpdateServiceMetadataPublisherService"
576 type="ServiceMetadataPublisherServiceType"/>
577   <xs:element name="DeleteServiceMetadataPublisherService"
578 ref="ServiceMetadataPublisherID"/>
579
580   <xs:complexType name="ServiceMetadataPublisherServiceType">
581     <xs:sequence>
582       <xs:element name="PublisherEndpoint" type="PublisherEndpointType"/>
583       <xs:element ref="ServiceMetadataPublisherID"/>
584     </xs:sequence>
585   </xs:complexType>
586
587   <xs:complexType name="PublisherEndpointType">
588     <xs:sequence>
589       <xs:element name="EndpointAddress" type="xs:anyURI"/>
590     </xs:sequence>
591   </xs:complexType>
592
593   <xs:complexType name="ServiceMetadataPublisherServiceForParticipantType">
594     <xs:sequence>
595       <xs:element ref="ServiceMetadataPublisherID"/>
596       <xs:element ref="ids:ParticipantIdentifier"/>
597     </xs:sequence>
598   </xs:complexType>
599
600   <xs:complexType name="ServiceMetadataPublisherIdentifierType">
601     <xs:sequence>
602       <xs:element ref="ServiceMetadataPublisherID"/>
603     </xs:sequence>
604   </xs:complexType>

```

```
605
606     <xs:element name="CreateParticipantIdentifier"
607 type="ServiceMetadataPublisherServiceForParticipantType" />
608     <xs:element name="DeleteParticipantIdentifier"
609 type="ServiceMetadataPublisherServiceForParticipantType" />
610     <xs:element name="ServiceMetadataPublishersService"
611 type="ServiceMetadataPublisherServiceType" />
612
613     <xs:element name="ParticipantIdentifierPage"
614 type="ParticipantIdentifierPageType" />
615     <xs:element name="CreateList" type="ParticipantIdentifierPageType" />
616     <xs:element name="DeleteList" type="ParticipantIdentifierPageType" />
617     <xs:complexType name="ParticipantIdentifierPageType">
618         <xs:sequence>
619             <xs:element ref="ServiceMetadataPublisherID" />
620             <xs:element ref="ids:ParticipantIdentifier" minOccurs="0"
621 maxOccurs="unbounded" />
622             <xs:element ref="PageID" minOccurs="0" />
623         </xs:sequence>
624     </xs:complexType>
625
626     <xs:element name="PageRequest" type="PageRequestType" />
627     <xs:complexType name="PageRequestType">
628         <xs:sequence>
629             <xs:element ref="ServiceMetadataPublisherID" />
630             <xs:element name="NextPageIdentifier" type="xs:string" minOccurs="0" />
631         </xs:sequence>
632     </xs:complexType>
633
634     <xs:element name="PrepareMigrationRecord" type="MigrationRecordType" />
635     <xs:element name="CompleteMigrationRecord" type="MigrationRecordType" />
636     <xs:complexType name="MigrationRecordType">
637         <xs:sequence>
638             <xs:element ref="ServiceMetadataPublisherID" />
639             <xs:element ref="ids:ParticipantIdentifier" />
640             <xs:element name="MigrationKey" type="xs:string" />
641         </xs:sequence>
642     </xs:complexType>
643
644     <xs:element name="BadRequestFault" type="FaultType" />
645     <xs:element name="InternalServerError" type="FaultType" />
646     <xs:element name="NotFoundFault" type="FaultType" />
647     <xs:element name="UnauthorizedFault" type="FaultType" />
648     <xs:complexType name="FaultType">
649         <xs:sequence>
650             <xs:element name="FaultMessage" type="xs:string" minOccurs="0" />
651         </xs:sequence>
652     </xs:complexType>
653 </xs:schema>
```

## 654 7 Appendix B: WSDLs (non-normative)

655 This section defines the WSDLs for the services offered as Web services. The normative versions of  
656 the files are published together with this specification.

### 657 7.1 peppol-sml-manage-business-identifier-service-v1.wsdl

```

658 <?xml version="1.0" encoding="utf-8"?>
659 <wsdl:definitions
660 xmlns:tns="http://busdoux.org/serviceMetadata/ManageBusinessIdentifierService/1.0/"
661     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
662     xmlns:lrs="http://busdoux.org/serviceMetadata/Locator/1.0/"
663     xmlns:xs="http://www.w3.org/2001/XMLSchema"
664     name="ManageBusinessIdentifierService"
665     targetNamespace=
666     "http://busdoux.org/serviceMetadata/ManageBusinessIdentifierService/1.0/"
667     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
668 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
669 <wsdl:types>
670 <xs:schema elementFormDefault="qualified" targetNamespace=
671     "http://busdoux.org/serviceMetadata/ManageBusinessIdentifierService/1.0/Schema/">
672 <xs:import namespace="http://busdoux.org/serviceMetadata/Locator/1.0/"
673     schemaLocation="peppol-sml-types-v1.xsd"/>
674 </xs:schema>
675 </wsdl:types>
676 <wsdl:message name="createIn">
677 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
678 <wsdl:part name="messagePart" element="Lrs:CreateParticipantIdentifier"/>
679 </wsdl:message>
680 <wsdl:message name="createOut">
681 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
682 </wsdl:message>
683 <wsdl:message name="deleteIn">
684 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
685 <wsdl:part name="messagePart" element="Lrs:DeleteParticipantIdentifier"/>
686 </wsdl:message>
687 <wsdl:message name="deleteOut">
688 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
689 </wsdl:message>
690 <wsdl:message name="listIn">
691 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
692 <wsdl:part name="messagePart" element="Lrs:PageRequest"/>
693 </wsdl:message>
694 <wsdl:message name="listOut">
695 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
696 <wsdl:part name="messagePart" element="Lrs:ParticipantIdentifierPage"/>
697 </wsdl:message>
698 <wsdl:message name="prepareMigrateIn">
699 <wsdl:part name="prepareMigrateIn" element="Lrs:PrepareMigrationRecord"/>
700 </wsdl:message>
701 <wsdl:message name="prepareMigrateOut">
702 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
703 </wsdl:message>
704 <wsdl:message name="migrateIn">
705 <wsdl:part name="migrateIn" element="Lrs:CompleteMigrationRecord"/>
706 </wsdl:message>
707 <wsdl:message name="migrateOut">
708 <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

```

```
709     </wsdl:message>
710     <wsdl:message name="createListIn">
711         <wsdl:part name="createListIn" element="Lrs:CreateList"/>
712     </wsdl:message>
713     <wsdl:message name="createListOut">
714         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
715     </wsdl:message>
716     <wsdl:message name="deleteListIn">
717         <wsdl:part name="deleteListIn" element="Lrs:DeleteList"/>
718     </wsdl:message>
719     <wsdl:message name="deleteListOut">
720         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
721     </wsdl:message>
722     <wsdl:message name="badRequestFault">
723         <wsdl:part name="fault" element="Lrs:BadRequestFault"/>
724     </wsdl:message>
725     <wsdl:message name="internalErrorFault">
726         <wsdl:part name="fault" element="Lrs:InternalErrorFault"/>
727     </wsdl:message>
728     <wsdl:message name="notFoundFault">
729         <wsdl:part name="fault" element="Lrs:NotFoundFault"/>
730     </wsdl:message>
731     <wsdl:message name="unauthorizedFault">
732         <wsdl:part name="fault" element="Lrs:UnauthorizedFault"/>
733     </wsdl:message>
734     <wsdl:portType name="ManageBusinessIdentifierServiceSoap">
735         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
736         <wsdl:operation name="Create">
737             <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
738             <wsdl:input message="tns:createIn"/>
739             <wsdl:output message="tns:createOut"/>
740             <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
741             <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
742             <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
743             <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
744         </wsdl:operation>
745         <wsdl:operation name="Delete">
746             <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
747             <wsdl:input message="tns:deleteIn"/>
748             <wsdl:output message="tns:deleteOut"/>
749             <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
750             <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
751             <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
752             <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
753         </wsdl:operation>
754         <wsdl:operation name="List">
755             <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
756             <wsdl:input message="tns:listIn"/>
757             <wsdl:output message="tns:listOut"/>
758             <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
759             <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
760             <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
761             <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
762         </wsdl:operation>
763         <wsdl:operation name="PrepareToMigrate">
764             <wsdl:input message="tns:prepareMigrateIn"/>
765             <wsdl:output message="tns:prepareMigrateOut"/>
766             <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
```

```

767     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
768     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
769     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
770 </wsdl:operation>
771 <wsdl:operation name="Migrate">
772     <wsdl:input message="tns:migrateIn"/>
773     <wsdl:output message="tns:migrateOut"/>
774     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
775     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
776     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
777     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
778 </wsdl:operation>
779 <wsdl:operation name="CreateList">
780     <wsdl:input message="tns:createListIn"/>
781     <wsdl:output message="tns:createListOut"/>
782     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
783     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
784     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
785     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
786 </wsdl:operation>
787 <wsdl:operation name="DeleteList">
788     <wsdl:input message="tns:deleteListIn"/>
789     <wsdl:output message="tns:deleteListOut"/>
790     <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
791     <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
792     <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
793     <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
794 </wsdl:operation>
795 </wsdl:portType>
796 <wsdl:binding name="ManageBusinessIdentifierServiceSoap"
797 type="tns:ManageBusinessIdentifierServiceSoap">
798     <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
799     <wsdl:operation name="Create">
800 <!--
801 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
802 -->
803     <soap:operation
804 soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
805 :createIn" style="document"/>
806     <wsdl:input>
807         <soap:body use="Literal"/>
808     </wsdl:input>
809     <wsdl:output>
810         <soap:body use="Literal"/>
811     </wsdl:output>
812     <wsdl:fault name="UnauthorizedFault">
813         <soap:fault name="UnauthorizedFault" use="Literal"/>
814     </wsdl:fault>
815     <wsdl:fault name="InternalErrorFault">
816         <soap:fault name="InternalErrorFault" use="Literal"/>
817     </wsdl:fault>
818     <wsdl:fault name="BadRequestFault">
819         <soap:fault name="BadRequestFault" use="Literal"/>
820     </wsdl:fault>
821 </wsdl:operation>
822 <wsdl:operation name="CreateList">
823 <!--
824 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!

```

```
825 -->
826     <soap:operation
827 soapAction="http://busdoo.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
828 :createListIn" style="document"/>
829     <wsdl:input>
830         <soap:body use="Literal"/>
831     </wsdl:input>
832     <wsdl:output>
833         <soap:body use="Literal"/>
834     </wsdl:output>
835     <wsdl:fault name="NotFoundFault">
836         <soap:fault name="NotFoundFault" use="Literal"/>
837     </wsdl:fault>
838     <wsdl:fault name="UnauthorizedFault">
839         <soap:fault name="UnauthorizedFault" use="Literal"/>
840     </wsdl:fault>
841     <wsdl:fault name="InternalServerError">
842         <soap:fault name="InternalServerError" use="Literal"/>
843     </wsdl:fault>
844     <wsdl:fault name="BadRequestFault">
845         <soap:fault name="BadRequestFault" use="Literal"/>
846     </wsdl:fault>
847 </wsdl:operation>
848 <wsdl:operation name="Delete">
849 <!--
850 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
851 -->
852     <soap:operation
853 soapAction="http://busdoo.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
854 :deleteIn" style="document"/>
855     <wsdl:input>
856         <soap:body use="Literal"/>
857     </wsdl:input>
858     <wsdl:output>
859         <soap:body use="Literal"/>
860     </wsdl:output>
861     <wsdl:fault name="NotFoundFault">
862         <soap:fault name="NotFoundFault" use="Literal"/>
863     </wsdl:fault>
864     <wsdl:fault name="UnauthorizedFault">
865         <soap:fault name="UnauthorizedFault" use="Literal"/>
866     </wsdl:fault>
867     <wsdl:fault name="InternalServerError">
868         <soap:fault name="InternalServerError" use="Literal"/>
869     </wsdl:fault>
870     <wsdl:fault name="BadRequestFault">
871         <soap:fault name="BadRequestFault" use="Literal"/>
872     </wsdl:fault>
873 </wsdl:operation>
874 <wsdl:operation name="DeleteList">
875 <!--
876 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
877 -->
878     <soap:operation
879 soapAction="http://busdoo.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
880 :deleteListIn" style="document"/>
881     <wsdl:input>
882         <soap:body use="Literal"/>
```

```
883     </wsdl:input>
884     <wsdl:output>
885         <soap:body use="Literal"/>
886     </wsdl:output>
887     <wsdl:fault name="NotFoundFault">
888         <soap:fault name="NotFoundFault" use="Literal"/>
889     </wsdl:fault>
890     <wsdl:fault name="UnauthorizedFault">
891         <soap:fault name="UnauthorizedFault" use="Literal"/>
892     </wsdl:fault>
893     <wsdl:fault name="InternalServerErrorFault">
894         <soap:fault name="InternalServerErrorFault" use="Literal"/>
895     </wsdl:fault>
896     <wsdl:fault name="BadRequestFault">
897         <soap:fault name="BadRequestFault" use="Literal"/>
898     </wsdl:fault>
899 </wsdl:operation>
900 <wsdl:operation name="List">
901 <!--
902 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
903 -->
904     <soap:operation
905 soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
906 :listIn" style="document"/>
907     <wsdl:input>
908         <soap:body use="Literal"/>
909     </wsdl:input>
910     <wsdl:output>
911         <soap:body use="Literal"/>
912     </wsdl:output>
913     <wsdl:fault name="NotFoundFault">
914         <soap:fault name="NotFoundFault" use="Literal"/>
915     </wsdl:fault>
916     <wsdl:fault name="UnauthorizedFault">
917         <soap:fault name="UnauthorizedFault" use="Literal"/>
918     </wsdl:fault>
919     <wsdl:fault name="InternalServerErrorFault">
920         <soap:fault name="InternalServerErrorFault" use="Literal"/>
921     </wsdl:fault>
922     <wsdl:fault name="BadRequestFault">
923         <soap:fault name="BadRequestFault" use="Literal"/>
924     </wsdl:fault>
925 </wsdl:operation>
926 <wsdl:operation name="PrepareToMigrate">
927 <!--
928 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
929 -->
930     <soap:operation
931 soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
932 :prepareMigrateIn" style="document"/>
933     <wsdl:input>
934         <soap:body use="Literal"/>
935     </wsdl:input>
936     <wsdl:output>
937         <soap:body use="Literal"/>
938     </wsdl:output>
939     <wsdl:fault name="NotFoundFault">
940         <soap:fault name="NotFoundFault" use="Literal"/>
```



```

941     </wsdl:fault>
942     <wsdl:fault name="UnauthorizedFault">
943         <soap:fault name="UnauthorizedFault" use="literal"/>
944     </wsdl:fault>
945     <wsdl:fault name="InternalErrorFault">
946         <soap:fault name="InternalErrorFault" use="literal"/>
947     </wsdl:fault>
948     <wsdl:fault name="BadRequestFault">
949         <soap:fault name="BadRequestFault" use="literal"/>
950     </wsdl:fault>
951 </wsdl:operation>
952 <wsdl:operation name="Migrate">
953 <!--
954 The 9 blanks in @soapAction are unfortunate but implemented like this in CEF SML!
955 -->
956     <soap:operation
957 soapAction="http://busdox.org/serviceMetadata/ManageBusinessIdentifierService/1.0/
958 :migrateIn" style="document"/>
959     <wsdl:input>
960         <soap:body use="literal"/>
961     </wsdl:input>
962     <wsdl:output>
963         <soap:body use="literal"/>
964     </wsdl:output>
965     <wsdl:fault name="NotFoundFault">
966         <soap:fault name="NotFoundFault" use="literal"/>
967     </wsdl:fault>
968     <wsdl:fault name="UnauthorizedFault">
969         <soap:fault name="UnauthorizedFault" use="literal"/>
970     </wsdl:fault>
971     <wsdl:fault name="InternalErrorFault">
972         <soap:fault name="InternalErrorFault" use="literal"/>
973     </wsdl:fault>
974     <wsdl:fault name="BadRequestFault">
975         <soap:fault name="BadRequestFault" use="literal"/>
976     </wsdl:fault>
977 </wsdl:operation>
978 </wsdl:binding>
979 <wsdl:service name="ManageBusinessIdentifierService">
980     <wsdl:port name="ManageBusinessIdentifierServicePort"
981 binding="tns:ManageBusinessIdentifierServiceSoap">
982         <soap:address location="unknown"/>
983     </wsdl:port>
984 </wsdl:service>
985 </wsdl:definitions>

```

## 986 7.2 peppol-sml-manage-service-metadata-service-v1.wsdl

```

987 <?xml version="1.0" encoding="utf-8"?>
988 <wsdl:definitions
989 xmlns:tns="http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/"
990     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
991     xmlns:lrns="http://busdox.org/serviceMetadata/Locator/1.0/"
992     xmlns:xs="http://www.w3.org/2001/XMLSchema"
993     name="ManageServiceMetadataService"
994     targetNamespace=
995 "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/"
996     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

```

```
997     <wsdl:types>
998         <xs:schema elementFormDefault="qualified" targetNamespace=
999             "http://busdox.org/serviceMetadata/ManageServiceMetadataService/1.0/Schema/">
1000             <xs:import namespace="http://busdox.org/serviceMetadata/Locator/1.0/"
1001                 schemaLocation="peppol-sml-types-v1.xsd"/>
1002         </xs:schema>
1003     </wsdl:types>
1004     <wsdl:message name="createIn">
1005         <wsdl:part name="messagePart"
1006             element="Lrs:CreateServiceMetadataPublisherService" />
1007     </wsdl:message>
1008     <wsdl:message name="createOut">
1009     </wsdl:message>
1010     <wsdl:message name="readIn">
1011         <wsdl:part name="messagePart"
1012             element="Lrs:ReadServiceMetadataPublisherService" />
1013     </wsdl:message>
1014     <wsdl:message name="readOut">
1015         <wsdl:part name="messagePart" element="Lrs:ServiceMetadataPublisherService" />
1016     </wsdl:message>
1017     <wsdl:message name="updateIn">
1018         <wsdl:part name="messagePart"
1019             element="Lrs:UpdateServiceMetadataPublisherService"/>
1020     </wsdl:message>
1021     <wsdl:message name="updateOut">
1022     </wsdl:message>
1023     <wsdl:message name="deleteIn">
1024         <wsdl:part name="messagePart" element="Lrs:ServiceMetadataPublisherID" />
1025     </wsdl:message>
1026     <wsdl:message name="deleteOut">
1027     </wsdl:message>
1028     <wsdl:message name="badRequestFault">
1029         <wsdl:part name="fault" element="Lrs:BadRequestFault"/>
1030     </wsdl:message>
1031     <wsdl:message name="internalErrorFault">
1032         <wsdl:part name="fault" element="Lrs:InternalErrorFault"/>
1033     </wsdl:message>
1034     <wsdl:message name="notFoundFault">
1035         <wsdl:part name="fault" element="Lrs:NotFoundFault"/>
1036     </wsdl:message>
1037     <wsdl:message name="unauthorizedFault">
1038         <wsdl:part name="fault" element="Lrs:UnauthorizedFault"/>
1039     </wsdl:message>
1040     <wsdl:portType name="ManageServiceMetadataServiceSoap">
1041         <wsdl:operation name="Create">
1042             <wsdl:input message="tns:createIn"/>
1043             <wsdl:output message="tns:createOut"/>
1044             <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
1045             <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
1046             <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
1047         </wsdl:operation>
1048         <wsdl:operation name="Read">
1049             <wsdl:input message="tns:readIn"/>
1050             <wsdl:output message="tns:readOut"/>
1051             <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
1052             <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
1053             <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
1054             <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>

```

```
1055     </wsdl:operation>
1056     <wsdl:operation name="Update">
1057         <wsdl:input message="tns:updateIn"/>
1058         <wsdl:output message="tns:updateOut"/>
1059         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
1060         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
1061         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
1062         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
1063     </wsdl:operation>
1064     <wsdl:operation name="Delete">
1065         <wsdl:input message="tns:deleteIn"/>
1066         <wsdl:output message="tns:deleteOut"/>
1067         <wsdl:fault message="tns:notFoundFault" name="NotFoundFault"/>
1068         <wsdl:fault message="tns:unauthorizedFault" name="UnauthorizedFault"/>
1069         <wsdl:fault message="tns:internalErrorFault" name="InternalErrorFault"/>
1070         <wsdl:fault message="tns:badRequestFault" name="BadRequestFault"/>
1071     </wsdl:operation>
1072 </wsdl:portType>
1073 <wsdl:binding name="ManageServiceMetadataServiceSoap"
1074 type="tns:ManageServiceMetadataServiceSoap">
1075     <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
1076     <wsdl:operation name="Create">
1077         <soap:operation soapAction=
1078 "http://busdoox.org/serviceMetadata/ManageServiceMetadataService/1.0/:createIn"
1079 style="document"/>
1080         <wsdl:input>
1081             <soap:body use="Literal"/>
1082         </wsdl:input>
1083         <wsdl:output>
1084             <soap:body use="Literal"/>
1085         </wsdl:output>
1086         <wsdl:fault name="UnauthorizedFault">
1087             <soap:fault name="UnauthorizedFault" use="Literal"/>
1088         </wsdl:fault>
1089         <wsdl:fault name="InternalErrorFault">
1090             <soap:fault name="InternalErrorFault" 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:operation name="Read">
1097         <soap:operation soapAction=
1098 "http://busdoox.org/serviceMetadata/ManageServiceMetadataService/1.0/:readIn"
1099 style="document"/>
1100         <wsdl:input>
1101             <soap:body use="Literal"/>
1102         </wsdl:input>
1103         <wsdl:output>
1104             <soap:body use="Literal"/>
1105         </wsdl:output>
1106         <wsdl:fault name="NotFoundFault">
1107             <soap:fault name="NotFoundFault" use="Literal"/>
1108         </wsdl:fault>
1109         <wsdl:fault name="UnauthorizedFault">
1110             <soap:fault name="UnauthorizedFault" use="Literal"/>
1111         </wsdl:fault>
1112         <wsdl:fault name="InternalErrorFault">
```

```
1113     <soap:fault name="InternalErrorFault" use="literal"/>
1114 </wsdl:fault>
1115 <wsdl:fault name="BadRequestFault">
1116     <soap:fault name="BadRequestFault" use="literal"/>
1117 </wsdl:fault>
1118 </wsdl:operation>
1119 <wsdl:operation name="Update">
1120     <soap:operation soapAction=
1121 "http://busdoox.org/serviceMetadata/ManageServiceMetadataService/1.0/:updateIn"
1122 style="document"/>
1123     <wsdl:input>
1124         <soap:body use="literal"/>
1125     </wsdl:input>
1126     <wsdl:output>
1127         <soap:body use="literal"/>
1128     </wsdl:output>
1129     <wsdl:fault name="NotFoundFault">
1130         <soap:fault name="NotFoundFault" use="literal"/>
1131     </wsdl:fault>
1132     <wsdl:fault name="UnauthorizedFault">
1133         <soap:fault name="UnauthorizedFault" use="literal"/>
1134     </wsdl:fault>
1135     <wsdl:fault name="InternalErrorFault">
1136         <soap:fault name="InternalErrorFault" use="literal"/>
1137     </wsdl:fault>
1138     <wsdl:fault name="BadRequestFault">
1139         <soap:fault name="BadRequestFault" use="literal"/>
1140     </wsdl:fault>
1141 </wsdl:operation>
1142 <wsdl:operation name="Delete">
1143     <soap:operation soapAction=
1144 "http://busdoox.org/serviceMetadata/ManageServiceMetadataService/1.0/:deleteIn"
1145 style="document"/>
1146     <wsdl:input>
1147         <soap:body use="literal"/>
1148     </wsdl:input>
1149     <wsdl:output>
1150         <soap:body use="literal"/>
1151     </wsdl:output>
1152     <wsdl:fault name="NotFoundFault">
1153         <soap:fault name="NotFoundFault" use="literal"/>
1154     </wsdl:fault>
1155     <wsdl:fault name="UnauthorizedFault">
1156         <soap:fault name="UnauthorizedFault" use="literal"/>
1157     </wsdl:fault>
1158     <wsdl:fault name="InternalErrorFault">
1159         <soap:fault name="InternalErrorFault" use="literal"/>
1160     </wsdl:fault>
1161     <wsdl:fault name="BadRequestFault">
1162         <soap:fault name="BadRequestFault" use="literal"/>
1163     </wsdl:fault>
1164 </wsdl:operation>
1165 </wsdl:binding>
1166 <wsdl:service name="ManageServiceMetadataService">
1167     <wsdl:port name="ManageServiceMetadataServicePort"
1168 binding="tns:ManageServiceMetadataServiceSoap">
1169         <soap:address location="unknown"/>
1170     </wsdl:port>
```

```
1171 </wsdl:service>  
1172 </wsdl:definitions>
```

