Specification

OpenPEPPOL AISBL

Peppol Transport Infrastructure
ICT - Models

Service Metadata Publishing (SMP)

Version: 1.2.0

Status: In use

Editors:

Gert Sylvest (NITA/Avanade)
Jens Jakob Andersen (NITA)
Klaus Vilstrup Pedersen (DIFI)
Mikkel Hippe Brun (NITA)
Paul Fremantle (NITA/WSO2)

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)	Mikkel Hippe Brun, NITA
1.0.1	2010-10-01	EC approved	Klaus Vilstrup Pedersen, DIFI
1.1.0	2012-08-15	Make room for alternative Transport Protocols e.g. AS2	Klaus Vilstrup Pedersen, DIFI
1.2.0	2020-02-20	Updated the references Improved layout Explicitly allowing Content-Type “application/xml” as it is equivalent to “text/xml” (chapter 5.1) Removing the requirement that the encoding attribute value is case sensitive (chapter 5.2) Change “is not” to “MUST NOT” in chapter 5.5 Replaced the references to the BusDox Common Definition document (BDEN-CEDF)	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This deliverable contains original unpublished work except where clearly indicated otherwise.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

Contributors

Organisations

DIFI (Direktoratet for forvaltning og IKT)[1], Norway, www.difi.no

NITA (IT- og Telestyrelsen)[2], Denmark, www.itst.dk

BRZ (Bundesrechenzentrum)[3], Austria, www.brz.gv.at

Consip, Italy

OpenPEPPOL

Persons

Bergthór Skúlason, NITA

Carl-Markus Piswanger, BRZ

Gert Sylvest, NITA/Avanade (editor)

Jens Jakob Andersen, NITA

Joakim Recht, NITA/Trifork

Kenneth Bengtsson, NITA/Alfa1lab

Klaus Vilstrup Pedersen, DIFI

Mike Edwards, NITA/IBM

Mikkel Hippe Brun, NITA

Paul Fremantle, NITA/WSO2

Philip Helger, BRZ/OpenPEPPOL OO

Thomas Gundel, NITA/IT Crew

[1] English: Agency for Public Management and eGovernment
[2] English: National IT- and Telecom Agency
[3] English: Austrian Federal Computing Centre

Table of contents

1 Introduction

1.1 Objective

This document describes the REST (Representational State Transfer) interface for Service Metadata Publication within the Business Document Exchange Network (BUSDOX). It describes the request/response exchanges between a Service Metadata Publisher and a client wishing to discover endpoint information. A client could be an end-user business application or an Access Point. It also defines the request processing that must happen at the client.

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.

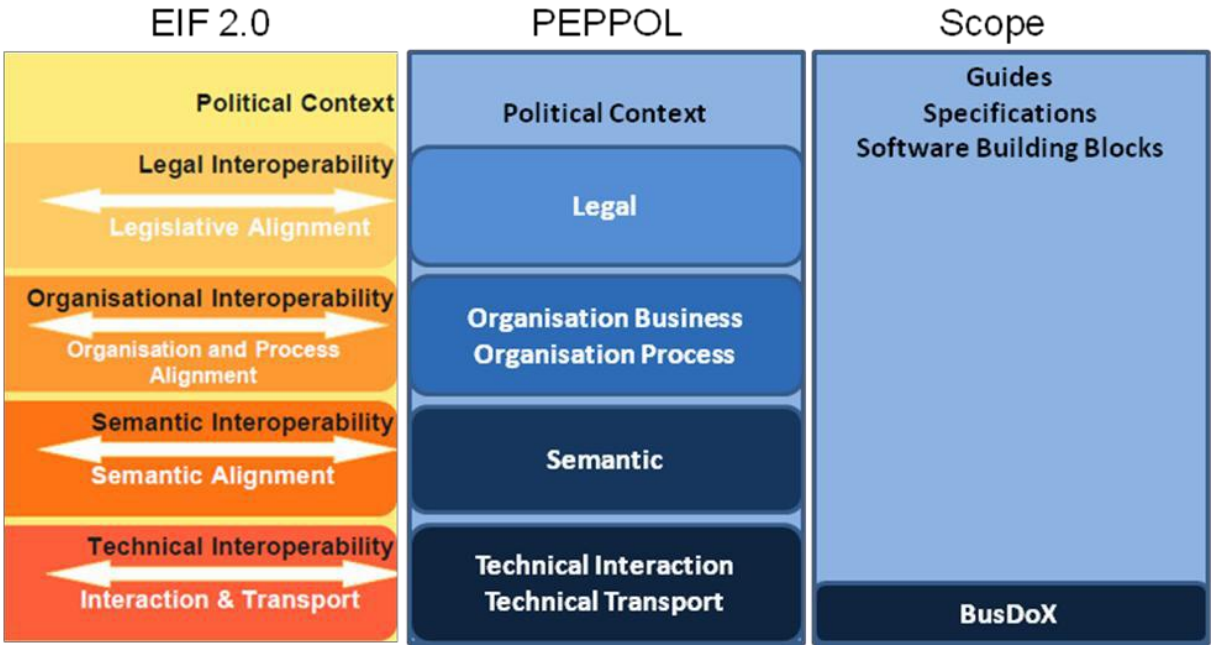

Fig. 1: Peppol Interoperability

1.3 Goals and non-goals

The goal of this document is to define the REST lookup interface that Service Metadata Publishers ("SMP") and clients must support. Decisions regarding physical data format and management interfaces are left to implementers of such a service.

Service Metadata Publishers may be subject to additional constraints of agreements and governance frameworks within instances of the BUSDOX infrastructure not covered in this specification, which only addresses the technical interface of such a service.

1.4 Terminology

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

1.4.1 Notational conventions

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

```
<!-- sample pseudo-schema -->
<defined element
    required_attribute_of_type_string="xs:string"
    optional_attribute_of_type_int="xs:int"? >
  <required_element />
  <optional_element />?
  <one_or_more_of_these_elements />+
  [ <choice_1 /> | <choice_2 /> ]*
</defined_element>
```

1.4.2 Normative references

[XML-DSIG] "XML Signature Syntax and Processing (Second Edition)", http://www.w3.org/TR/xmldsig-core/

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

[WSA-1.0] "Web Services Addressing 1.0 - Core" (http://www.w3.org/TR/2005/CR-ws-addrcore-20050817/) and "Web Services Addressing 1.0 - SOAP Binding", http://www.w3.org/TR/wsaddr-soap/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", http://www.ietf.org/rfc/rfc2119.txt

[PFUOI4] Policy for use of Identifiers 4.0, https://github.com/OpenPEPPOL/documentation/raw/master/TransportInfrastructure/PEPPOL-EDN-Policy-for-use-of-identifiers-4.0-2019-01-28.pdf

1.4.3 Non-normative references

[WSDL-2.0] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language", http://www.w3.org/TR/wsdl20/

[REST] “Architectural Styles and the Design of Network-based Software Architectures”, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[BDEN-SML] Service Metadata Locator Profile, ServiceMetadataLocator.pdf

1.5 Namespaces

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

Prefix	Namespace URI
ds	http://www.w3.org/2000/09/xmldsig#
ids	http://busdox.org/transport/identifiers/1.0/
smp	http://busdox.org/serviceMetadata/publishing/1.0/

wsa	http://www.w3.org/2005/08/addressing
xs	http://www.w3.org/2001/XMLSchema

2 The Service Discovery Process

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

- Service Metadata Locator:
  - DNS-based resolve mechanism to locate individual SMPs
  - Management interface towards SMPs
- Service Metadata Publishers:
  - Discovery interface towards senders

This specification only covers the discovery interface for Service Metadata Publication services.

2.1 Discovery flow

For a sender, the first step in the Discovery process is to establish the location of the Service Metadata relating to the particular Participant Identifier to which the sender wants to transmit a message. Each participant identifier is registered with one and only one Service Metadata Publisher. The sender looks up the endpoint for the Service Metadata Publisher using the DNS-based Service Metadata Locator service (this is a regular DNS resolve). The sender can then retrieve the metadata associated with the Participant Identifier. This metadata includes the information necessary to transmit the message to the recipient endpoint.

The diagram below represents the lookup flow for a sender contacting both the Service Metadata Locator and the SMP.

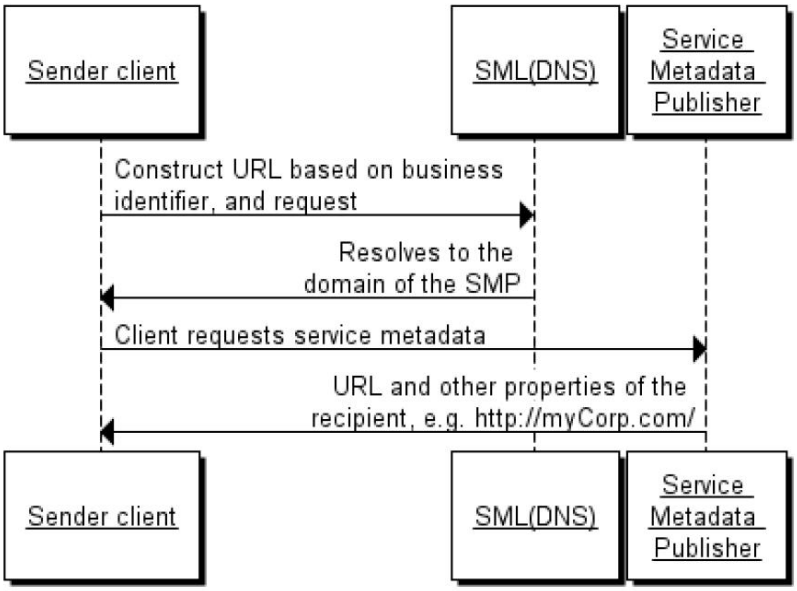

Fig. 2: Endpoint lookup with Service Metadata

Note: For optimization reasons, the discovery doesn't have to be performed for every transfer if the necessary information for transfer is already cached from previous transmissions. Though necessary exception handling has to be in place i.e. new lookup has to be performed if the sending shows that information is outdated e.g. old endpoint address.

2.1.1 Discovering services associated with a Participant Identifier

In addition to the direct lookup of Service Metadata based on participant identifier and document type, a sender may want to discover what document types can be handled by a specific participant identifier. Such discovery is relevant for applications supporting several equivalent business processes. Knowing the capabilities of the recipient is valuable information to a sender application and ultimately to an end user. E.g. the end user may be presented with a choice between a "simple" and a "rich" business process.

This is enabled by a pattern where the sender first retrieves the *ServiceGroup* entity, which holds a list of references to the *ServiceMetadata* resources associated with it. The *SignedServiceMetadata* in turn holds the metadata information that describes the capabilities associated with the recipient participant identifier

2.2 Service Metadata Publisher Redirection

For each participant identifier, the SML may only point to a single SMP. There are cases however where the owner of a participant identifier may want to use different SMPs for different document types or processes. This is supported by Service Metadata Publisher Redirection.

In this pattern, the sender is redirected by the SMP to a secondary, remote SMP where the actual *SignedServiceMetadata* can be found. A special element within the *SignedServiceMetadata* record of the SMP points to the SMP that has the actual Service Metadata and certificate information for that SMP. The diagram below shows this flow:

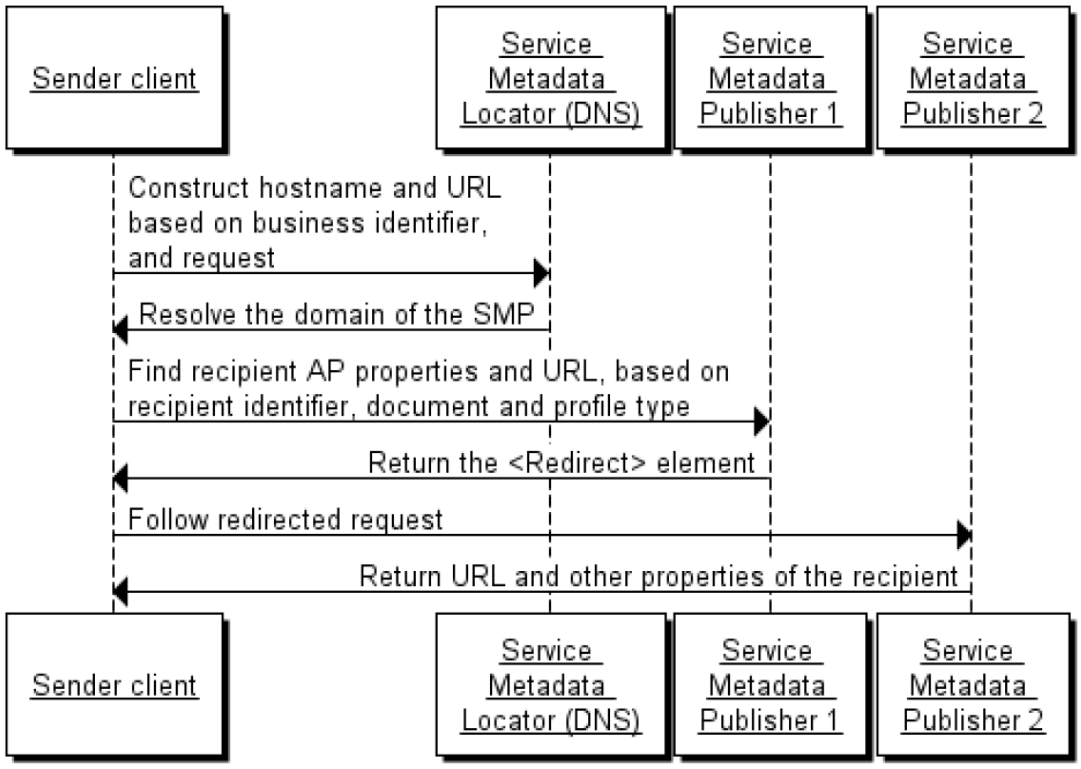

Fig. 3: Service Metadata Redirection

Note that only one degree of redirect is allowed; clients are not required to follow more than one redirect, i.e. a redirect resource cannot point to another redirect resource. Allowing one level of redirect permits the described use case to be realized, while avoiding the possibility of cyclic references and long chains of redirects

3 Interface model

This specification defines a REST-based interface for retrieving Service Metadata, but does not specify interfaces for creating, updating, deleting and managing Service Metadata, or any internal data storage formats.

The goal is to allow the interface in this specification to expose data from many different Service Metadata back-ends, which may be based on any suitable technology such as for example RDBMS, LDAP, or UDDI.

Note that when adding or deleting Participant Identifiers in the SMP, an implementation of the SMP will need to reflect its custody of a Participant Identifier in the SML. Please see the SML specification [BDEN-SML] for a description of the processes and interfaces for doing this.

4 Data model

This section outlines the data model of the interface. The data model comprises the following main data types:

- ServiceGroup
- ServiceMetadata / SignedServiceMetadata

Supporting data types for these main types are:

- ServiceInformation
- ServiceEndpointList
- ParticipantIdentifier
- DocumentIdentifier
- Redirect
- Process
- ProcessList
- Endpoint

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

4.1 On extension points

For each major entity, extension points have been added with the optional `<smp:Extension>` element.

4.1.1 Semantics and use

Child elements of the `<smp:Extension>` element are known as “custom extension elements”. Extension points may be used for optional extensions of service metadata. This implies:

- Extension elements added to a specific Service Metadata resource MUST be ignorable by any client of the transport infrastructure. The ability to parse and adjust client behaviour based on an extension element MUST NOT be a prerequisite for a client to locate a service, or to make a successful request at the referenced service.
- A client MAY ignore any extension element added to specific service metadata resource instances.

4.2 ServiceGroup

The *ServiceGroup* structure represents a set of services associated with a specific participant identifier that is handled by a specific SMP. The *ServiceGroup* structure holds a list of references to *SignedServiceMetadata* resources in the *ServiceList* structure.

Pseudo-schema for *ServiceGroup*:

```
<smp:ServiceGroup>
  <ids:ParticipantIdentifier scheme=”xs:string”>
    xs:string
  </ids:ParticipantIdentifier>
  <smp:ServiceMetadataReferenceCollection>
    <smp:ServiceMetadataReference href=”xs:anyURI” />*
    </smp:ServiceMetadataReferenceCollection>
  <smp:Extension>xs:any</smp:Extension>?
</smp:ServiceGroup>
```

Description of the individual fields (elements and attributes).

Field	Description
ServiceGroup	Document element
ParticipantIdentifier	Represents the business level endpoint key and key type, e.g. a DUNS or GLN number that is associated with a group of services. See [PFUOI4] for information on this data type.
ServiceMetadataReferenceCollection	This structure holds a list of references to *SignedServiceMetadata* structures. From this list, a sender can follow the references to get each *SignedServiceMetadata* structure.
ServiceMetadataReference (0..*)	Contains the URL to a specific *SignedServiceMetadata* instance - see the REST binding section for details on the URL format. Note that references MUST refer to *SignedServiceMetadata* records that are signed by the certificate of the SMP. It MUST NOT point to *SignedServiceMetadata* resources published by external SMPs.
Extension	The extension element may contain any XML element. Clients MAY ignore this element. It can be used to add extended metadata to individual references to Service Metadata resources.

4.2.1 Non-normative example

Non-normative example of a *ServiceGroup* resource:

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
This sample assumes that the service metadata publisher resides at
"http://serviceMetadata.eu/".
It assumes that the business identifier is "0010:5798000000001".
-->
<ServiceGroup xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:ids="http://busdox.org/transport/identifiers/1.0/">
  <ids:ParticipantIdentifier scheme="busdox-actorid-upis">
    0010:5798000000001
  </ids:ParticipantIdentifier>
  <ServiceMetadataReferenceCollection>
    <ServiceMetadataReference href="http://serviceMetadata.eu/busdox-actorid-
upis%3A%3A0010%3A5798000000001/services/busdox-docid-
qns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AInvoice-
2%3A%3AInvoice%23%23UBL-2.0"/>
  </ServiceMetadataReferenceCollection>
  <Extension>
    <ex:Test xmlns:ex="http://test.eu">Test</ex:Test>
  </Extension>
</ServiceGroup>
```

4.3 ServiceMetadata

This data structure represents Metadata about a specific electronic service. The role of the *ServiceMetadata* structure is to associate a participant identifier with the ability to receive a specific

document type over a specific transport. It also describes which business processes a document can participate in, and various operational data such as service activation and expiration times.

The *ServiceMetadata* resource contains all the metadata about a service that a sender Access Point needs to know in order to send a message to that service.

4.3.1 Redirection

For recipients that want to associate more than one SMP with their participant identifier, they may redirect senders to an alternative SMP for specific document types. To achieve this, the *ServiceMetadata* element defines the optional element `Redirect`. This element holds the URL of the alternative SMP, as well as the Subject Unique Identifier of the destination SMPs certificate used to sign its resources.

In the case where a client encounters such a redirection element, the client MUST follow the first redirect reference to the alternative SMP. If the *SignedServiceMetadata* resource at the alternative SMP also contains a redirection element, the client SHOULD NOT follow that redirect. It is the responsibility of the client to enforce this constraint.

Pseudo-schema for this data type:

```
<smp:ServiceMetadata>
  [<smp:ServiceInformation /> | <smp:Redirect />]
</smp:ServiceMetadata>
```

Pseudo-schema for the `ServiceInformation` data type:

```
<smp:ServiceInformation>
  <ids:ParticipantIdentifier scheme=”xs:string”>xs:string
  </ids:ParticipantIdentifier>
  <ids:DocumentIdentifier scheme=”xs:string” />
  <smp:ProcessList>
    <smp:Process>+
      <ids:ProcessIdentifier scheme=”xs:string” />
      <smp:ServiceEndpointList>
        <smp:Endpoint transportProfile=”xs:string”>+
          <wsa:EndpointReference />
          <smp:RequireBusinessLevelSignature>xs:boolean
          </smp:RequireBusinessLevelSignature>
          <smp:MinimumAuthenticationLevel>xs:string
          </smp:MinimumAuthenticationLevel >?
          <smp:ServiceActivationDate>xs:dateTime
          </smp:ServiceActivationDate>?
          <smp:ServiceExpirationDate>xs:dateTime
          </smp:ServiceExpirationDate>?
          <smp:Certificate>xs:string</smp:Certificate>
          <smp:ServiceDescription>xs:string
          </smp:ServiceDescription>
          <smp:TechnicalContactUrl>xs:anyURI
          </smp:TechnicalContactUrl>
          <smp:TechnicalInformationUrl>xs:anyURI
          </smp:TechnicalInformationUrl>?
          <smp:Extension>xs:any</smp:Extension>?
        </smp:Endpoint>
      </smp:ServiceEndpointList>
      <smp:Extension>xs:any</smp:Extension>?
    </smp:Process>
  </smp:ProcessList>
  <smp:Extension>xs:any</smp:Extension>?
</smp:ServiceInformation>
```

Pseudo-schema for the `Redirect` data type:

```
<smp:Redirect href="xs:anyURI">
  <smp:CertificateUID>xs:string</smp:CertificateUID>
  <smp:Extension>xs:any</smp:Extension>?
<smp:Redirect>
```

The Extension element may contain any XML element. Clients MAY ignore this element. It can be used to add extension metadata to the service metadata.

The `href` attribute of the `Redirect` element contains the full address of the destination SMP record that the client is redirected to.

For example, assume that an SMP called "SMP1" has the address `http://smp1.eu`, and another SMP called "SMP2" has the address `http://smp2.eu`, and a client requests a resource with the following URL (note that these examples have been percent encoded):

```
http://smp1.eu/busdox-actorid-
upis%3A%3A0010%3A5798000000001/services/busdox-docid-
qns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AInvoice
- 2%3A%3AInvoice%23%23UBL-2.0
```

We now assume that the owner of these metadata has moved them to SMP2. SMP1 would then return a *SignedServiceMetadata* resource with a `Redirect` child element that has the `href` attribute set to

```
http://smp2.eu/busdox-actorid-
upis%3A%3A0010%3A5798000000001/services/busdox-docid-
qns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AInvoice
- 2%3A%3AInvoice%23%23UBL-2.0
```

For the list of endpoints under each `Endpoint` element in the `ServiceEndpointList`, each endpoint MUST have different values of the `transportProfile` attribute, i.e. represent bindings to different transports.

Description of the individual fields (elements and attributes).

Field	Description
/ServiceMetadata	Document element
ServiceMetadata/Redirect	The direct child element of `ServiceMetadata` is either the `Redirect` element or the `ServiceInformation` element. The `Redirect` element indicates that a client must follow the URL of the `href` attribute of this element.
Redirect/CertificateUID	Holds the Subject Unique Identifier of the certificate of the destination SMP. A client SHOULD validate that the Subject Unique Identifier of the certificate used to sign the resource at the destination SMP matches the Subject Unique Identifier published in the redirecting SMP.
Redirect/Extension	The `Extension` element may contain any XML element. Clients MAY ignore this element. It can be used to add extension metadata to the `Redirect`.
ServiceMetadata/ServiceInformation	The direct child element of `ServiceMetadata` is either the `Redirect` element or the

Field	Description
	`ServiceInformation` element. The `ServiceInformation` element contains service information for an actual service registration, rather than a redirect to another SMP.
ServiceInformation/ParticipantIdentifier	The participant identifier. Comprises the identifier, and an identifier scheme. This identifier MUST have the same value of the {id} part of the URI of the enclosing *ServiceMetadata* resource. See the ParticipantIdentifier section of the 'Policy for use of identifiers' document [PFUOI4] for information on this data type.
ServiceInformation/DocumentIdentifier	Represents the type of document that the recipient is able to handle. The document type is represented by an identifier (identifying the document type) and an identifier scheme, which the format of the identifier itself. See the DocumentTypeIdentifier section of the 'Policy for use of identifiers' document [PFUOI4] for information on this data type.
ServiceInformation/ProcessList	Represents the processes that a specific document type can participate in, and endpoint address and binding information. Each process element describes a specific business process that accepts this type of document as input and holds a list of endpoint addresses (in the case that the service supports multiple transports) of services that implement the business process, plus information about the transport used for each endpoint. See the Process section of the 'Policy for use of identifiers' document [PFUOI4] for information on the identifier format.
Process/ProcessIdentifier	The identifier of the process. See the 'Policy for use of identifiers' document for a definition of process identifiers [PFUOI4]
Process/ServiceEndpointList	List of one or more endpoints that support this process.
ServiceEndpointList/Endpoint	`Endpoint` represents the technical endpoint and address type of the recipient, as an URL.
Endpoint/EndpointReference	The address of an endpoint, as a WS-Addressing Endpoint Reference (EPR).
Endpoint/@transportProfile	Indicates the type of transport protocol that is being used between access points, e.g. the Peppol AS4 profile (`peppol-transport-as4-v2_0`). A list of valid transport protocols is referenced from the 'Policy for use of identifiers' document [PFUOI4].

Field	Description
Endpoint/RequireBusinessLevelSignature	Set to `true` if the recipient requires business-level signatures for the message, meaning a signature applied to the business message before the message is put on the transport. This is independent of the transport-level signatures that a specific transport profile, such as the Peppol AS4 profile, might mandate. This flag does not indicate which type of business-level signature might be required. Setting or consuming business-level signatures would typically be the responsibility of the final senders and receivers of messages, rather than a set of APs.
Endpoint/MinimumAuthenticationLevel	Indicates the minimum authentication level that recipient requires. The specific semantics of this field is defined in a specific instance of the BUSDOX infrastructure. It could for example reflect the value of the "urn:eu:busdox:attribute:assurance-level" SAML attribute defined in the START specification.
Endpoint/ServiceActivationDate	Activation date of the service. Senders SHOULD ignore services that are not yet activated. Format of ServiceActivationDate date is `xs:dateTime`.
Endpoint/ServiceExpirationDate	Expiration date of the service. Senders SHOULD ignore services that are expired. Format of ServiceExpirationDate date is `xs:dateTime`.
Endpoint/Certificate	Holds the complete signing certificate of the recipient AP, as a PEM (base 64) encoded X509 DER formatted value.
Endpoint/ServiceDescription	A human readable description of the service.
Endpoint/TechnicalContactUrl	Represents a link to human readable contact information. This might also be an email address.
Endpoint/TechnicalInformationUrl	A URL to human readable documentation of the service format. This could for example be a web site containing links to XML Schemas, WSDLs, Schematrons and other relevant resources.
Process/Extension	The `Extension` element may contain any XML element. Clients MAY ignore this element. It can be used to add extension metadata to the process metadata block as a whole.
ServiceInformation/Extension	The `Extension` element may contain any XML element. Clients MAY ignore this element. It can be used to add extension metadata to the service metadata.

4.3.2 Non-normative example

For a non-normative example of a *ServiceMetadata* resource, see the *SignedServiceMetadata* non-normative example below.

4.4 SignedServiceMetadata

The *SignedServiceMetadata* structure is a *ServiceMetadata* structure that has been signed by the SMP, according to governance policies that are not covered by this document. Pseudo-schema for this data type:

```
<smp:SignedServiceMetadata>
  <smp:ServiceMetadata />
  <ds:Signature />
</smp:SignedServiceMetadata>
```

- `ServiceMetadata` is the `ServiceMetadata` element covered by the signature.
- `Signature` represents an enveloped XML signature over the `SignedServiceMetadata` element.

4.4.1 Non-normative example

Non-normative example of a *SignedServiceMetadata* resource.

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
This sample assumes that the service metadata publisher resides at
"http://serviceMetadata.eu/".
It assumes that the business identifier is "0010:5798000000001".
-->
<SignedServiceMetadata xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:ids="http://busdox.org/transport/identifiers/1.0/">
  <ServiceMetadata xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-
utility-1.0.xsd">
    <ServiceInformation>
      <ids:ParticipantIdentifier scheme="busdox-actorid-
upis">0010:5798000000001</ids:ParticipantIdentifier>
      <ids:DocumentIdentifier scheme="busdox-docid-
qns">urn:oasis:names:specification:ubl:schema:xsd:Invoice-2::Invoice##UBL-
2.02</ids:DocumentIdentifier>
      <ProcessList>
        <Process>
          <ids:ProcessIdentifier scheme="cenbii-procid-
ubl">BII04</ids:ProcessIdentifier>
          <ServiceEndpointList>
            <Endpoint transportProfile="busdox-transport-start">
              <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
                <Address>http://busdox.org/sampleService/</Address>
              </EndpointReference>
              <RequireBusinessLevelSignature>false</RequireBusinessLevelSignature>
              <MinimumAuthenticationLevel>2</MinimumAuthenticationLevel>
              <ServiceActivationDate>2009-05-01T09:00:00</ServiceActivationDate>
              <ServiceExpirationDate>2016-05-01T09:00:00</ServiceExpirationDate>
              <Certificate>TlRMTVNTUAABAAAAt7IY4gk....</Certificate>
              <ServiceDescription>invoice service</ServiceDescription>
              <TechnicalContactUrl>https://example.com</TechnicalContactUrl>

<TechnicalInformationUrl>http://example.com/info</TechnicalInformationUrl>
            </Endpoint>
          </ServiceEndpointList>
        </Process>
        <Process>
```

```
          <ids:ProcessIdentifier scheme="cenbii-procid-
ubl">BII07</ids:ProcessIdentifier>
          <ServiceEndpointList>
            <Endpoint transportProfile="busdox-transport-start">
              <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
                <Address>http://busdox.org/sampleService/</Address>
              </EndpointReference>
              <RequireBusinessLevelSignature>true</RequireBusinessLevelSignature>
              <MinimumAuthenticationLevel>1</MinimumAuthenticationLevel>
              <ServiceActivationDate>2009-05-01T09:00:00</ServiceActivationDate>
              <ServiceExpirationDate>2016-05-01T09:00:00</ServiceExpirationDate>
              <Certificate>TlRMTVNTUAABAAAAt7IY4gk....</Certificate>
              <ServiceDescription>invoice service</ServiceDescription>
              <TechnicalContactUrl>https://example.com</TechnicalContactUrl>

<TechnicalInformationUrl>http://example.com/info</TechnicalInformationUrl>
              <Extension>
                <ex:Test xmlns:ex="http://test.eu">Test</ex:Test>
              </Extension>
            </Endpoint>
          </ServiceEndpointList>
          <Extension>
            <ex:Test xmlns:ex="http://test.eu">Test</ex:Test>
          </Extension>
        </Process>
      </ProcessList>
      <Extension>
        <ex:Test xmlns:ex="http://test.eu">Test</ex:Test>
      </Extension>
    </ServiceInformation>
  </ServiceMetadata>
  <!-- Message signature, details omitted for brevity -->
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"/>
</SignedServiceMetadata>
```

4.4.2 Redirect, non-normative example

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
This sample assumes that the user contacts a service metadata publisher that
resides at "http://serviceMetadata.eu/",
but is redirected to a service metadata publisher that resides at
"http://serviceMetadata2.eu/".
-->
<SignedServiceMetadata xmlns="http://busdox.org/serviceMetadata/publishing/1.0/">
  <ServiceMetadata xmlns="http://busdox.org/serviceMetadata/publishing/1.0/">
    <Redirect xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
href="http://serviceMetadata2.eu/busdox-
actoridupis%3A%3A0010%3A5798000000001/services/busdox-
docidqns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AInvoice-
2%3A%3AInvoice%23%23UBL-2.0">
      <CertificateUID>PID:9208-2001-3-279815395</CertificateUID>
      <Extension>
        <ex:Test xmlns:ex="http://test.eu">Test</ex:Test>
      </Extension>
    </Redirect>
  </ServiceMetadata>
  <!-- Message signature, details omitted for brevity -->
```

```
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"/>
</SignedServiceMetadata>
```

5 Service Metadata Publishing REST binding

This section describes the REST binding of the SMP interface.

5.1 The use of HTTP

A service implementing the REST binding MUST set the HTTP `Content-Type` header, and give it a value of `text/xml` or `application/xml`. A service implementing the REST profile MUST NOT use TLS (Transport Layer Security) or SSL (Secure Sockets Layer). An instance of the BUSDOX infrastructure MAY set restrictions on what ports are allowed.

An implementation of the SMP might choose to manage resources through the HTTP POST, PUT and DELETE verbs. It is however up to each implementation to choose how to manage records, and use of HTTP POST, PUT and DELETE is not mandated or regulated by this specification.

HTTP GET operations MUST return the following HTTP status codes:

HTTP Status Code	Meaning
200	Must be returned if the resource is retrieved correctly.
404	Code 404 must be returned if a specific resource could not be found. This could for example be the result of a request containing a participant identifier that does not exist.
500	Code 500 must be returned if the service experiences an internal processing error.

The service MAY support other HTTP status codes as well.

The service SHOULD NOT use HTTP redirection in the manner indicated by the HTTP 3xx codes. Clients are not required to support active redirection.

5.2 The use of XML and encoding

XML document returned by HTTP GET MUST be UTF-8 encoded. They MUST contain a document type declaration starting with `<?xml` which includes the `encoding` attribute set to `UTF-8`. Please observe that the content of the encoding attribute is not case sensitive. Version 1.0 of XML is used.

5.3 Resources and identifiers

The REST interface comprises 2 types of resources.

Resource	URI	Method	XML resource root element	HTTP Status	Description of returned content
ServiceGroup	/{identifier scheme}::{id}	GET	<ServiceGroup>	200; 500; 404	Holds the participant identifier of the recipient, and a list of references to individual ServiceMetadata resources that are associated with that participant identifier.
SignedServiceMet	/{identifier	GET	<SignedSe	200;	Holds all of the metadata

adata	scheme}::{id}/services/{docType} See section below for {docType} format	rviceMetadata>	500; 404	about a Service, or a redirection URL to another Service Metadata Publisher holding this information.

Fig. 4: Table of resources and identifiers

A service implementing the REST binding MUST support these resource types. It MUST provide access to these using the URI scheme of table in Fig. 3.

5.3.1 On the use of percent encoding

When any types of BUSDOX identifiers are used in URLs, each section between slashes MUST be percent encoded according to [RFC3986] individually, i.e. section by section.

For example, this implies that for an URL in the form of `/{identifier scheme}::{id}/services/{docType}` the slash literals MUST NOT be URL encoded.

5.3.2 Using identifiers in the REST Resource URLs

This section describes specifically how participant and document identifiers are used to reference *ServiceGroup* and *SignedServiceMetadata* REST resources. For a general definition on how to represent participant and document identifiers in URLs, see [PFUOI4].

For the URL referencing a *ServiceGroup* resource, the `{identifier scheme}::{id}` part follows the participant identifier format described in the "ParticipantIdentifier" section of the 'Policy for use of identifiers' document [PFUOI4].

The following URL format is used:

```
/{identifier scheme}::{id}
```

In the reference to the *SignedServiceMetadata* or *Redirect* resources (`/{id}/services/{docType}`), the `{docType}` part consists of `{document type identifier scheme}::{document type identifier}`. For information on the format of `{document type identifier}`, see the DocumentIdentifier section of the 'Policy for use of identifiers' document [PFUOI4].

5.3.3 Non-normative identifier example

We assume an SMP can be accessed at the URL `http://serviceMetadata.eu`.

A business with the participant identifier `0010:5798000000001` would have the following identifier for the *ServiceGroup* resource:

```
http://serviceMetadata.eu/busdox-actorid-upis::0010:5798000000001
```

After percent encoding:

```
http://serviceMetadata.eu/busdox-actorid-upis%3a%3a0010%3a5798000000001
```

In the case of a NES-UBL order, a *SignedServiceMetadata* or *Redirect* resource can then be identified by

- Identifier format type: `busdox-docid-qns`
- Root namespace: `urn:oasis:names:specification:ubl:schema:xsd:Order-2`
- Document element local name: `Order`

- Subtype identifier: `UBL-2.0` (since several versions of the Order schema may use the same namespace + document element name)

The document type identifier will then be:

```
busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:Order-
2::Order##UBL-2.0
```

The document type identifier MUST be percent encoded as described in [RFC3986]. The above, non-normative example is thus encoded to

```
busdox-docid-
qns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AOrder-
2%3A%3AOrder%23%23UBL-2.0
```

The entire URL reference to a *SignedServiceMetadata* or *Redirect* resource thus has the form

```
{URL to server}/{identifier scheme}::{id}/services/{document identifier
type}::{rootNamespace}::{documentElementLocalName}[##{Subtype identifier}]
```

The percent-encoded form of the identifier using the above example will then be

```
http://serviceMetadata.eu/busdox-actorid-
upis%3a%3a0010%3a5798000000001/services/busdox-docid-
qns%3A%3Aurn%3Aoasis%3Anames%3Aspecification%3Aubl%3Aschema%3Axsd%3AOrder-
2%3A%3AOrder%23%23UBL-2.0
```

Note that the forward slashes delimiting the individual parts of the REST resource identifier URL are not percent encoded, since they are part of the URL.

5.3.4 Implementation considerations

When a client is redirected to an SMP using the DNS-based SML scheme described in [BDEN-SML], the HTTP `Host` header will be set to a value originating from the CNAME alias set in the SML (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.23). Implementations should be prepared to accept requests with this “host” header value.

5.4 Referencing the SMP REST binding

For referencing the SMP REST binding, for example from SML records, the following identifier should be used for the version 1.0 of the SMP REST binding:

```
http://busdox.org/serviceMetadata/publishing/1.0/
```

This is identical to the target namespace of the SMP schema.

5.5 Security

At the transport level, the service MUST NOT be secured.

5.5.1 Message signature

The message returned by the service is signed by the Service Metadata Publisher with XML-Signature according to the standard `http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/`.

The signature MUST be an enveloped XML signature represented via a `ds:Signature` element embedded in the `SignedServiceMetadata` element. The `ds:Signature` element MUST be constructed according to the following rules:

- The <Reference> MUST use exactly one Transform being: `http://www.w3.org/2000/09/xmldsig#envelopedsignature`

- The <ds:KeyInfo> element MUST contain an <ds:X509Data> element with an <ds:X509Certificate> sub-element containing the signer's X.509 certificate as PEM (base 64) encoded X509 DER value.
- The canonicalization algorithm MUST be `http://www.w3.org/2001/10/xml-exc-c14n#`
- The SignatureMethod MUST `be http://www.w3.org/2000/09/xmldsig#rsa-sha1`
- The DigestMethod MUST be `http://www.w3.org/2000/09/xmldsig#sha1`

5.5.2 Verifying the signature

When verifying the signature, the consumer has access to the full certificate as a PEM (base 64) encoded X509 DER value within the `dsSignature` element. The consumer may verify the signature by

a) extracting the certificate from the `ds:X509Data` element,
b) verify that it has been issued by the trusted root,
c) perform a validation of the signature, and
d) perform the required certificate validation steps (which might include checking expiration/activation dates and revocation lists).

5.5.3 Verifying the signature of the destination SMP

For the redirect scheme, the unique identifier of the destination SMP signing certificate is stored at the redirecting SMP. In addition to the regular signature validation performed by the client of the destination SMP resources, the client SHOULD also validate that the identifier of the destination SMP signing certificate corresponds to the unique identifier which the redirecting SMP claims belongs to the destination SMP.

6 Appendix A: Schema for the REST interface

This section defines the XML Schema for all the resources of the REST interface.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ServiceMetadataPublishing"
targetNamespace="http://busdox.org/serviceMetadata/publishing/1.0/"
elementFormDefault="qualified"
xmlns="http://busdox.org/serviceMetadata/publishing/1.0/"
xmlns:ids="http://busdox.org/transport/identifiers/1.0/"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <xs:import schemaLocation="xmldsig-core-schema.xsd"
namespace="http://www.w3.org/2000/09/xmldsig#"/>
  <xs:import schemaLocation="oasis-200401-wss-wssecurity-utility-1.0.xsd"
namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"/>
  <xs:import schemaLocation="ws-addr.xsd"
namespace="http://www.w3.org/2005/08/addressing"/>
  <xs:import schemaLocation="Identifiers-1.0.xsd"
namespace="http://busdox.org/transport/identifiers/1.0/"/>

  <xs:element name="ServiceGroup" type="ServiceGroupType"/>
  <xs:element name="ServiceMetadata" type="ServiceMetadataType"/>
  <xs:element name="SignedServiceMetadata" type="SignedServiceMetadataType"/>

  <xs:complexType name="SignedServiceMetadataType">
    <xs:sequence>
      <xs:element ref="ServiceMetadata"/>
      <xs:element ref="ds:Signature"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceMetadataType">
    <xs:sequence>
      <xs:choice>
        <xs:element name="ServiceInformation" type="ServiceInformationType"/>
        <xs:element name="Redirect" type="RedirectType"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceInformationType">
    <xs:sequence>
      <xs:element ref="ids:ParticipantIdentifier"/>
      <xs:element ref="ids:DocumentIdentifier"/>
      <xs:element name="ProcessList" type="ProcessListType"/>
      <xs:element name="Extension" type="ExtensionType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ProcessListType">
    <xs:sequence>
      <xs:element name="Process" type="ProcessType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

```

```
  <xs:complexType name="ProcessType">
    <xs:sequence>
      <xs:element ref="ids:ProcessIdentifier"/>
      <xs:element name="ServiceEndpointList" type="ServiceEndpointList"/>
      <xs:element name="Extension" type="ExtensionType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceEndpointList">
    <xs:sequence>
      <xs:element name="Endpoint" type="EndpointType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="EndpointType">
    <xs:sequence>
      <xs:element ref="wsa:EndpointReference"/>
      <xs:element name="RequireBusinessLevelSignature" type="xs:boolean"/>
      <xs:element name="MinimumAuthenticationLevel" type="xs:string"
minOccurs="0"/>
      <xs:element name="ServiceActivationDate" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="ServiceExpirationDate" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="Certificate" type="xs:string"/>
      <xs:element name="ServiceDescription" type="xs:string"/>
      <xs:element name="TechnicalContactUrl" type="xs:anyURI"/>
      <xs:element name="TechnicalInformationUrl" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="Extension" type="ExtensionType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="transportProfile" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="ServiceGroupType">
    <xs:sequence>
      <xs:element ref="ids:ParticipantIdentifier"/>
      <xs:element name="ServiceMetadataReferenceCollection"
type="ServiceMetadataReferenceCollectionType"/>
      <xs:element name="Extension" type="ExtensionType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceMetadataReferenceCollectionType">
    <xs:sequence>
      <xs:element name="ServiceMetadataReference"
type="ServiceMetadataReferenceType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceMetadataReferenceType">
    <xs:attribute name="href" type="xs:anyURI"/>
  </xs:complexType>

  <xs:complexType name="RedirectType">
    <xs:sequence>
      <xs:element name="CertificateUID" type="xs:string"/>
      <xs:element name="Extension" type="ExtensionType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="href" type="xs:anyURI"/>
  </xs:complexType>
```

```
  <xs:complexType name="ExtensionType">
    <xs:sequence>
      <xs:any/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```