Dynamic Discovery Proof-of-Concept

1. Technical Use Cases

The proof-of-concept illustrates a number of technical use cases for service metadata and an infrastructure that supports discovery of service metadata.

1.1. Finding the Gateway and PMode for a Receiving End Entity

One potential use for a discovery infrastructure is to discover the relation from an end entity to its gateway. In this scenario, it is shown how the PEPPOL SMP protocol and format can be used to obtain the association of a receiving end entity to the gateway at which that end entity receives messages. The SMP record is discovered using the BDX Metadata Service Location mechanism which is an elaboration of the PEPPOL SML.

The configuration of the exchange with that gateway is assumed to be defined using the configuration mechanism of the sending and receiving gateway. One configuration mechanism is provided by the Holodeck PMode files and associated security configuration and certificate stores.

The SMP data structure for the receiving end entity provides information on the endpoint address of the gateway endpoint it uses to receive messages. That address can be used, in combination with the process and document identifier, to determine which PMode to use to connect to that end entity. This is illustrated in the test_connect_smp.py script in the proof-of-concept, a screen shot of a run of which is show in the following figure:

Figure 1 Finding a PMode using SMP

The processing in the script follows the following steps:

- At initialization, the script loads the available configuration information at the sending gateway. The configuration is product-specific. The proof-of-concept implementation has functionality to process Holodeck XML files in a specified local directory and build various indexes on the information in these files.
- A metadata service location query is performed for the end entity identifier *govello-1234567890123-4567789012*. This record is a NAPTR-U DNS record, the content of which is shown and discussed in section 2.1. The URI for the metadata resource references a domain called *localhost*.
- The identified metadata resource can be retrieved using an HTTP GET request. It is determined that the resource is an SMP 1.0 *ServiceGroup* structure. (see section 2.2) This structure lists the various documents the party can receive. The script scans this document for a reference for the *Form_A* document type.
- The *Form_A* reference points to a separate resource which defines service metadata for this document type. The resource can again be retrieved using the REST protocol.
- The retrieved resource is an SMP 1.0 *SignedServiceMetadata* resource which lists the use of the form in various business processes.
- The cached PMode configuration is consulted and it is established that a PMode has been configured to exchange between *AT* and *DE* in this process for this document.

This script requires that the sending gateway identifier is provided. This is typically not an issue, there should be no ambiguity at the sending end entity (corner 1) about the gateway (corner 2) it uses to communicate with other entities. Even if an end entity uses multiple gateways, these would in practice serve different services or communities. E.g. a court may use an e-invoicing service provider to pay the bills of its outsourced caterer, and an e-CODEX gateway for e-Justice communication. So in the example, the sending end entity ERV-1 submits its message to a gateway *AT*, which is requested to send the message to an end entity *govello-1234567890123-4567789012* for the process *EPO* and document type *Form_A*. The discovery process finds out that this end entity receives messages at an endpoint that is the endpoint of gateway service provider *DE*. For this process and document type, *AT* and *DE* have a PMode with identifier *AT_DE_EPO_Form_A*.

The key point is that SMP for the end entity does not identify the service provider *DE*. The correlation is based on endpoint address only.

1.2. Generalized Four-Corner Matching

The concept of matching metadata information for end entities and their gateways and the PModes registered at a sending gateway can be generalized and used with other metadata representation formats. One such format is the ebXML Collaboration Protocol Profile (CPP) format. The proof-of-concept script *test_connect_cpp_match.py* shows how a matching process using CPP metadata representations can be performed in a generalized way. The following script shows discovery of the configuration to use for the exchange of a *SubmitOrder* action in a service *OrderingBilling* between end entities *c1* and *c4*, via the sending gateway *c2* and receiving gateway *c3*.

```
C:\opt\bdx\bdmsl>test_connect_cpp_match.py
Connecting c1 to c4 OrderingBilling SubmitOrder None None
Retrieving metadata from http://localhost:8888/c4_end_entity_seller.cpp.xml
Retrieved receiver metadata is a CPP 3.0 CollaborationProtocolProfile
Receiving action bound to channel C4_s1_a1
Actionbinding C4_s1_a1 is bound to a remote channel http://localhost:8888/c3_gat
eway_Seller.cpp.xml#c3_s1_a1
Downloading referenced metadata resource http://localhost:8888/c3_gateway_Seller
.cpp.xml
Actionbinding c3_s1_a1 is bound to a local channel c3_channel
Retrieving metadata from http://localhost:8888/c1_end_entity_buyer.cpp.xml
Sender metadata is also a CPP 3.0 CollaborationProtocolProfile
Sender action binding has ID c1_s1_a1
Actionbinding c1_s1_a1 is bound to a remote channel http://localhost:8888/c2_gat
eway_buyer.cpp.xml#c2_s1_a1
Downloading referenced metadata resource http://localhost:8888/c2_gateway_buyer.
cpp.xml
Actionbinding c2_s1_a1 is bound to a local channel c2_channel
Sending delegated to gateway C2 for SubmitOrder and c4
Found a cached PMode for C2 C3 OrderingBilling SubmitOrder: OrderingBilling_Subm
itOrder_C2_C3
Data retrieved via CPP of receiver and sender OrderingBilling_SubmitOrder_C2_C3
```

Figure 2 Matching CPPs in four-corner contexts

The processing in the script follows the following steps:

- The initialization is as in the script described in 1.1. The example uses PMode data from a file shown in section 2.7.
- A metadata service location query is performed for the receiving end entity identifier *c4*. This record is a NAPTR-U DNS record, the content of which is shown and discussed in section 2.1.
- The identified metadata resource can be retrieved using an HTTP GET request. It is determined that the resource is a CPP 3.0 *CollaborationProtocolProfile* structure. (see section 2.5) This structure lists the various documents the party can send or receive in the context of specific services. In this resource, it is expressed that *c4* has delegated the handling of *SubmitOrder* messages to a third party, *c3*. The metadata resource of c3 and the definition in

that resource are identified as a combination of a URI and a fragment identifier that maps to an action binding with a particular XML identifier.

- A recursive lookup is done, this time for the metadata resource *c3*. This time, the action binding is local (meaning *c3* does not delegate the action to another entity), meaning the recursion ends.
- A metadata service location query is also performed for the sending end entity identifier *c1*. This is possible as CPP is not limited to receiving capabilities but also defines sending capabilities. The delegation of *c1* to *c2* does not have to be specified in the script, but can be looked up using the same metadata registry functionality as is used for receiving end entities.
- The metadata for *c2* can be downloaded and does not require further queries, as was the case for *c3*.
- As was the case in the preceding use case, sufficient parameters are available now to determine that a matching PMode exists for *c2* and *c3*.

1.3. Dynamically Inferring a PMode

The previous two scenarios assumed that the gateways were already configured (using a PMode or equivalent mechanism) to exchange messages. The discovery aspect was focussed on the end entities and on the configuration matching. As an extension to these, it is conceivable that the gateways are not yet configured and that, as a result of discovery, it is determined that a new PMode must be configured. Note that PModes references security configuration information, such as certificates. So when dynamically updating messaging configurations, it may be necessary to update certificate stores as well. Any new certificates may be associated with either the sender or the recipient.

The SMP format does not fully support this reconfiguration as it only support a small number of configuration parameters. In particular, it does not support associating receiving gateways with specific certificates. Moreover, it does not support configuration of sending gateways at all. Therefore, the proof-of-concept does not demonstrate PMode (re)configuration using SMP.

For CPP, the proof-of-concept includes an incomplete script that shows the processing that could be done when no previously configured PMode is available.

```
C:\opt\bdx\bdmsl>test_connect_cpp_nomatch.py
Connecting c11 to c14 OrderingBilling SubmitOrder None None
Retrieving metadata from http://localhost:8888/c14_end_entity_seller.cpp.xml
Retrieved receiver metadata is a CPP 3.0 CollaborationProtocolProfile
Receiving action bound to channel C14_s1_a1
Actionbinding C14_s1_a1 is bound to a remote channel http://localhost:8888/c13_g
ateway_Seller.cpp.xml#c13_s1_a1
Downloading referenced metadata resource http://localhost:8888/c13_gateway_Selle
r.cpp.xml
Actionbinding c13_s1_a1 is bound to a local channel c13_channel
Retrieving metadata from http://localhost:8888/c11_end_entity_buyer.cpp.xml
Sender metadata is also a CPP 3.0 CollaborationProtocolProfile
Sender action binding has ID c11_s1_a1
Actionbinding c11_s1_a1 is bound to a remote channel http://localhost:8888/c12_g
ateway_buyer.cpp.xml#c12_s1_a1
Downloading referenced metadata resource http://localhost:8888/c12_gateway_buyer
.cpp.xml
Actionbinding c12_s1_a1 is bound to a local channel c12_channel
Sending delegated to gateway C12 for SubmitOrder and c14
Matching action bindings c12_channel and c13_channel for C12, OrderingBilling, S
ubmitOrder:
Security mapping
Check that receiver TLS server cert Seller_ServerCert is trusted and configured
Check that sender TLS client cert Buyer_ClientCert is trusted and configured
Sender and receiver have compatible message protocol versions: 3.0, endpoint htt
p://c13.example.com/msh
Data retrieved via CPPs of receiver and sender, dynamically created Pmode  C12_C
13_OrderingBilling_SubmitOrder
```

Figure 3 Dynamically Inferring a PMode

The full set of parameters that can be configured in a PMode is quite large, and this functionality is not implement. The script only suggests some potential processing. For example, it determines the TLS client and server certificates that are specified in the CPP files and suggests that the security configuration be checked to see if these certificates are trusted. SMP is tied to PEPPOL's dedicated CA model, and does not encode certificate information.

The (work in progress) CPP 3.0 schema has been enhanced to cover all parameters for ebMS 3.0. The derivation of a PMode is similar to the derivation of a CPA as an intersection of two CPPs. Such intersection has been defined and implemented for the earlier v2.0 CPP/CPA. The sharing of a constructed PMode is not discussed further in this document.

1.4. Checking Matching Business Capabilities

A requirement for business interaction is the ability to determine whether a match in business capabilities exists between two parties [DDREQS]. A buyer party may want to check whether its suppliers are supporting one of more of the business processes it engages in (in complementary role as seller). This means that if party A is able to engage in an *OrderingBilling* service, B should also be. Moreover, for every action that A can perform, either as sender or as receiver, B should be able to perform these roles as well, as receiver or buyer. Depending on specific needs, the reverse (A matching all actions for B) may or may not be relevant. A metadata registry allows these matches to be automated in a simple way, if the metadata for A and B covers both sending and receiving capabilities.

```
C:\opt\bdx\bdmsl>test_business_match.py
Performing a business match for 'c1' and 'c4'
Retrieving metadata from http://localhost:8888/c1_end_entity_buyer.cpp.xml
Retrieving metadata from http://localhost:8888/c4_end_entity_seller.cpp.xml
c1 and c4 both use service OrderingBilling
c1 and c4 have matching send action SubmitOrder
c1 and c4 have matching receive action AcceptOrder
c1 has a receive action RejectOrder for which c4 does not have a send action
No (complete) match for service OrderingBilling
c4 and c1 both use service OrderingBilling
c4 and c1 have matching receive action SubmitOrder
c4 and c1 have matching send action AcceptOrder
All actions of c4 for OrderingBilling are matched by c1
```

Figure 4 Matching Business Capabilities

The screen dump shows a script that matches the business capabilities of partner *c1* and *c4.* It suggests that *c1* expects order rejection to be exchanged using the action *RejectOrder* but that *c4* has no such action as a sending capability. So when *c1* considers sending an order to *c4* it can already validate whether or not *c4* is able to send the expected responses. In fact, a complete match can be done for all business processes (profiles, in CEN BII terminology) without a need to know the expected message choreography.

A metadata format like CPP that expresses sending capabilities also allows business matches to be computed for parties that can only send documents, such as Supplier/Creditors engaging in the CEN BII profile 04, Basic Invoice Only profile [BII04]. A metadata discovery system based on a metadata format that only supports receiving capabilities like SMP would not have any record for such partners. They would be completely invisible.

1.5. Checking Community Membership

A metadata format that expresses sending capabilities also allows business matches to be computed for parties. A metadata discovery system based on a metadata format that only supports receiving capabilities would not have any record for parties that can only send documents, such as Supplier/Creditors engaging in the CEN BII profile 04, Basic Invoice Only profile [BII04]. Such a discovery system can not be used to see if a party is a member of a community. It can only be used to determine if a party is a member of a community if that party has a receiving capability.

2. Sample Data

2.1. BDXR Metadata Service Location NAPTR-U

Discovery of metadata service documents based on party identifiers in the PoC is based on the draft OASIS BDXR metadata service location [BDMSL], which is a functional successor to and generalization of the PEPPOL service metadata locator protocol [SML]. It can be used to retrieve data for a known party identifier. The BDXR specification is based on the IETF URI-based naming authority pointer resource record concept [RFC4848]. Using a domain name search key, the DNS record provides a URI for a resource.

```
govello-1234567890123-456789012.test.community.eu.
IN NAPTR 100 10 "U" "meta:smp" "!.*!http://localhost:8888/govello-1234567890123-
456789012.servicegroup.smp.xml!" .

c1.test.community.eu
IN NAPTR 100 10 "U" "meta:ebcpp3"
!.*!http://localhost:8888/c1_end_entity_buyer.cpp.xml!" .
```

A NAPTR-U record can be of a particular type, and the type can be used as a filter. In the first entry, the type is *meta:smp*, suggesting a service metadata resource in SMP format. The second entry also has examples of metadata in other formats, such as CPP 3.0.

The example also demonstrates the use of the concept of service provider domains, as described in section 2.2.1 of [BDMSL] and as previously used in [SML] with the addition of support for multiple environments (here *test* and *production*) in a community domain. In this model, the domain that is being published is the concatenation of the party identifier with an environment suffix (here *.test*, meaning that the resource for testing rather than the resource for production is being requested) and a service provider domain (here *.community.eu*). In this example it is assumed that this domain does not use normalization algorithms such as MD5 hashing in SML, but this is done for simplification only.

2.2. SMP ServiceGroup

The resource identifier obtained using a BDXR metadata query of type *meta:smp* could be an SMP *ServiceGroup* [SMP]. A *ServiceGroup* is an association of an identified party and a collection of service metadata references.

SMP is document-oriented rather than service-oriented, and the assumption is that an end entity knows the type of document it wants to send to another end entity, and uses the information in the service group to find a resource with document type-specific routing information.

```
<?xml version="1.0" encoding="UTF-8"?>
<smp:ServiceGroup xmlns:smp="http://busdox.org/serviceMetadata/publishing/1.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ids="http://busdox.org/transport/identifiers/1.0/">
    <ids:ParticipantIdentifier scheme="egvp">govello-1234567890123-
456789012</ids:ParticipantIdentifier>
    <smp:ServiceMetadataReferenceCollection>
        <smp:ServiceMetadataReference
            href="http://localhost:8888/govello-1234567890123-
456789012.Form_A.signedservicemetadata.smp.xml" />
    </smp:ServiceMetadataReferenceCollection>
</smp:ServiceGroup>
```

SMP uses PEPPOL conventions for the encoding of document type identifiers. The proof-of-concept simplifies this and assumes a client uses a simple regular expression match on the values of the *href*

attributes against the document type requested. In PEPPOL, document identifiers typically relate to commonly known document types, such as orders and invoices, which can be understood outside the context of a particular business process. In e-CODEX, document identifiers such as "Form A" only make sense in the context of a particular process.

The XML resources pointed to by service metadata references would be *SignedServiceMetadata* structures.

2.3. SMP SignedServiceMetadata

An SMP *SignedServiceMetadata* structure is a hierarchical structure for a particular identifier and document type identifier. The structure only covers the document type the party can receive. For a party that can only send documents (e.g. the CEN BII Invoice-Only profile), there is no queryable resource record.

Documents are linked to a list of service endpoints for specific transport protocols, identified by an attribute value. The following example references a transport profile *ebms3-as4*.

```
<?xml version="1.0" encoding="UTF-8"?>
<smp:SignedServiceMetadata
xmlns:smp="http://busdox.org/serviceMetadata/publishing/1.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ids="http://busdox.org/transport/identifiers/1.0/" >
    <smp:ServiceMetadata>
        <smp:ServiceInformation>
            <ids:ParticipantIdentifier scheme="egvp">govello-1234567890123-
456789012</ids:ParticipantIdentifier>
            <ids:DocumentIdentifier>Form_A</ids:DocumentIdentifier>
            <smp:ProcessList>
                <smp:Process>
                    <ids:ProcessIdentifier>EPO</ids:ProcessIdentifier>
                    <smp:ServiceEndpointList>
                        <smp:Endpoint transportProfile="ebms3-as4">
                            <EndpointReference
xmlns="http://www.w3.org/2005/08/addressing">

<Address>http://someserveringermany.de/gateway</Address>
                            </EndpointReference>

<smp:RequireBusinessLevelSignature>false</smp:RequireBusinessLevelSignature>

<smp:MinimumAuthenticationLevel>2</smp:MinimumAuthenticationLevel>
                            <smp:ServiceActivationDate>2009-05-
01T09:00:00</smp:ServiceActivationDate>
                            <smp:ServiceExpirationDate>2016-05-
01T09:00:00</smp:ServiceExpirationDate>

<smp:Certificate>TlRMTVNTUAABAAAAt7IY4gk....</smp:Certificate>
                            <smp:ServiceDescription>ordering
service</smp:ServiceDescription>

<smp:TechnicalContactUrl>https://example.com</smp:TechnicalContactUrl>

<smp:TechnicalInformationUrl>http://example.com/info</smp:TechnicalInformationUrl>
                        </smp:Endpoint>
                    </smp:ServiceEndpointList>
                </smp:Process>
            </smp:ProcessList>
        </smp:ServiceInformation>
    </smp:ServiceMetadata>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm=""></ds:CanonicalizationMethod>
            <ds:SignatureMethod Algorithm=""></ds:SignatureMethod>
            <ds:Reference>
                <ds:DigestMethod Algorithm=""></ds:DigestMethod>
```

```
                <ds:DigestValue></ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue></ds:SignatureValue>
    </ds:Signature>
</smp:SignedServiceMetadata>
```

2.4. CPP Gateway CollaborationProtocolProfile

An ebXML Collaboration Protocol Profile defines the roles an indentified party can perform in particular business collaborations, and the binding of the exchanges in that process/role context to particular channels. A CPP defines both sending and receiving business capabilities and supports alternative channel bindings, e.g. for ebMS 2.0, 3.0, AS2 or Web Services.

CPP has mechanism to encode the use of particular certificates used by a party and express their use for purposes such as TLS client or server authentication, WS-Security signing or encryption. It is also possible to define application certificates, to support encryption that is handled by an application or end entity rather than the targeted message handler.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpp:CollaborationProtocolProfile xmlns:cpp="http://docs.oasis-
open.org/ebxmlcppa/cppa-3.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xlink="http://www.w3.org/1999/xlink"
    xsi:schemaLocation="http://docs.oasis-open.org/ebxmlcppa/cppa-3.0
https://www.oasis-open.org/committees/download.php/49199/cppa-3.0%20May%202013.xsd"
    cppid="c2_cpp" version="3.0">
    <cpp:PartyInfo partyName="Buyer Gateway">
        <cpp:PartyId>C12</cpp:PartyId>
        <cpp:PartyRef xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href=""/>
        <cpp:CollaborationRole>
            <cpp:ProcessSpecification version="2.0" name="BII06-Procurement"
xlink:type="simple"

xlink:href="http://www.cen.eu/cwa/bii/specs/Profiles/ebbp/BIIprofile06-
Procurementv1.xml"/>
            <cpp:Role name="Buyer" xlink:type="simple"

xlink:href="http://www.cen.eu/cwa/bii/specs/Profiles/ebbp/BIIprofile06-
Procurementv1.xml#Buyer"/>
            <cpp:ServiceBinding>
                <cpp:Service>OrderingBilling</cpp:Service>
                <cpp:ActionBinding id="c12_s1_a1" action="SubmitOrder"
sendOrReceive="send">
                    <cpp:ChannelId>c12_channel</cpp:ChannelId>
                </cpp:ActionBinding>
                <cpp:ActionBinding id="c12_s1_a2" action="AcceptOrder"
sendOrReceive="receive">
                    <cpp:ChannelId>c12_channel</cpp:ChannelId>
                </cpp:ActionBinding>
                <cpp:ActionBinding id="c12_s1_a3" action="RejectOrder"
sendOrReceive="receive">
                    <cpp:ChannelId>c12_channel</cpp:ChannelId>
                </cpp:ActionBinding>
            </cpp:ServiceBinding>
        </cpp:CollaborationRole>

        <cpp:Certificate certId="Buyer_SigningCert">
            <ds:KeyInfo>
                <ds:KeyName>Buyer_SigningCert</ds:KeyName>
            </ds:KeyInfo>
        </cpp:Certificate>
```

```
        <cpp:Certificate certId="Buyer_ServerCert">
            <ds:KeyInfo>
                <ds:KeyName>Buyer_ServerCert_Key</ds:KeyName>
                <ds:X509Data>
                    <ds:X509SubjectName>C2 Gateway</ds:X509SubjectName>
                    <ds:X509Certificate>aGFsbG8gYWxsZW1hYWw=</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </cpp:Certificate>

        <cpp:Certificate certId="Buyer_ClientCert">
            <ds:KeyInfo> <ds:KeyName>Buyer_ClientCert_Key</ds:KeyName>
</ds:KeyInfo>
        </cpp:Certificate>

        <cpp:DeliveryChannel channelId="c12_channel" docExchangeId="c2_docexchange"
            transportId="c2_transport_client"/>

        <cpp:Transport transportId="c2_transport_client">
            <cpp:TransportSender>
                <cpp:TransportProtocol version="1.1"
method="POST">HTTP</cpp:TransportProtocol>
                <cpp:TransportClientSecurity>
                    <cpp:TransportSecurityProtocol
version="1.2">TLS</cpp:TransportSecurityProtocol>
                    <cpp:ClientCertificateRef certId="Buyer_ClientCert"/>
                </cpp:TransportClientSecurity>
            </cpp:TransportSender>
            <cpp:TransportReceiver>
                <cpp:TransportProtocol version="1.1">HTTP</cpp:TransportProtocol>
                <cpp:Endpoint uri="http://c2.example.com/msh"/>
                <cpp:TransportServerSecurity>
                    <cpp:TransportSecurityProtocol
version="1.2">TLS</cpp:TransportSecurityProtocol>
                    <cpp:ServerCertificateRef certId="Buyer_ServerCert"/>
                </cpp:TransportServerSecurity>
            </cpp:TransportReceiver>
        </cpp:Transport>

        <cpp:DocExchange docExchangeId="c2_docexchange">
            <cpp:ebXMLSenderBinding version="3.0">
                <!-- details omitted -->
                <cpp:SenderNonRepudiation>
                    <cpp:NonRepudiationProtocol>wss-v1.1-spec-errata-os-
X509TokenProfile</cpp:NonRepudiationProtocol>

<cpp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</cpp:HashFunction>
                    <cpp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#rsa-
sha1</cpp:SignatureAlgorithm>
                    <cpp:SigningCertificateRef certId="Buyer_SigningCert"/>
                </cpp:SenderNonRepudiation>
            </cpp:ebXMLSenderBinding>
            <cpp:ebXMLReceiverBinding version="3.0">
                <!-- details omitted -->
            </cpp:ebXMLReceiverBinding>
        </cpp:DocExchange>

    </cpp:PartyInfo>
</cpp:CollaborationProtocolProfile>
```

2.5. CPP End Entity CollaborationProtocolProfile

In CPP, a channel binding uses the XML IDREF values to bind the action to a channel defined in the CPP document. To support four-corner exchanges, a proposed enhancement would be to alternatively allow URI-valued channel references into other collaboration protocol profiles. An end

entity could then reference the profile of the service provider gateway that handles messages in a particular context on its behalf. These references would cover both sending and receiving capabilities.

Note that the referenced CPP could in turn reference other CPPs. In theory, five and more-corner models could be supported.

Combination of locally bound actions (which the end entity handles directly, in peer-to-peer mode) and remotely bound actions (delegated to service providers) are easily handled. The proof-of-concept client assumes a single binding per action for simplification.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpp:CollaborationProtocolProfile xmlns:cpp="http://docs.oasis-
open.org/ebxmlcppa/cppa-3.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xsi:schemaLocation="http://docs.oasis-open.org/ebxmlcppa/cppa-3.0
https://www.oasis-open.org/committees/download.php/49199/cppa-3.0%20May%202013.xsd"
cppid="c1_cpp" version="3.0">
    <cpp:PartyInfo partyName="Buyer" >
        <cpp:PartyId>c11</cpp:PartyId>
        <cpp:PartyRef xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href=""/>
        <cpp:CollaborationRole>
            <cpp:ProcessSpecification version="2.0"
                name="BII06-Procurement"
                xlink:type="simple"

xlink:href="http://www.cen.eu/cwa/bii/specs/Profiles/ebbp/BIIprofile06-
Procurementv1.xml"
                uuid="70cb8e96-862a-46be-9028-02032ae9ed3c"/>
            <cpp:Role name="Buyer"
                xlink:type="simple"

xlink:href="http://www.cen.eu/cwa/bii/specs/Profiles/ebbp/BIIprofile06-
Procurementv1.xml#Buyer"/>
            <cpp:ServiceBinding>
                <cpp:Service>OrderingBilling</cpp:Service>
                <cpp:ActionBinding id="c11_s1_a1" action="SubmitOrder"
sendOrReceive="send">

<cpp:ChannelRef>http://localhost:8888/c12_gateway_buyer.cpp.xml#c12_s1_a1</cpp:Chan
nelRef>
                </cpp:ActionBinding>
                <cpp:ActionBinding id="c11_s1_a2" action="AcceptOrder"
sendOrReceive="receive">

<cpp:ChannelRef>http://localhost:8888/c12_gateway_buyer.cpp.xml#c12_s1_a2</cpp:Chan
nelRef>
                </cpp:ActionBinding>
                <cpp:ActionBinding id="c11_s1_a3" action="RejectOrder"
sendOrReceive="receive">

<cpp:ChannelRef>http://localhost:8888/c12_gateway_buyer.cpp.xml#c12_s1_a3</cpp:Chan
nelRef>
                </cpp:ActionBinding>
            </cpp:ServiceBinding>
        </cpp:CollaborationRole>
    </cpp:PartyInfo>
</cpp:CollaborationProtocolProfile>
```

2.6. Holodeck PMode Data

The proof-of-concept includes two Holodeck PMode XML files, which are processed and indexed. The first of these is the PMode file for the *EPO* business process.

```
<PModes>

    <Producer name="EE">
      <PartyId>EE</PartyId>
      <Role>GW</Role>
    </Producer>

    <Producer name="IT">
      <PartyId>IT</PartyId>
      <Role>GW</Role>
    </Producer>

    <Producer name="AT">
      <PartyId>AT</PartyId>
      <Role>GW</Role>
    </Producer>


<!-- Definition of the Business Process for EPO for Austria -->

    <UserService name="EPO_Form_A">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>Form_A</Action>
      </CollaborationInfo>
    </UserService>

    <UserService name="EPO_Form_B">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>Form_B</Action>
      </CollaborationInfo>
    </UserService>

    <UserService name="EPO_SubmissionAcceptance">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>SubmissionAcceptanceRejection</Action>
      </CollaborationInfo>
    </UserService>

     <UserService name="EPO_RelayMD">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>RelayREMMDAcceptanceRejection</Action>
      </CollaborationInfo>
    </UserService>

     <UserService name="EPO_Delivery">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
```

```
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>DeliveryNonDeliveryToRecipient</Action>
      </CollaborationInfo>
    </UserService>

     <UserService name="EPO_Retrieval">
      <ToPartyInfo>
        <PartyId>DE</PartyId>
        <Role>GW</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>EPO</Service>
        <Action>RetrievalNonRetrievalToRecipient</Action>
      </CollaborationInfo>
    </UserService>

<!--Definition of the bindings for EPO -->
<!-- From IT to DE -->
    <Binding name="IT_DE_EPO_Form_A">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_A" producer="IT"
security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="IT_DE_EPO_Form_B">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_B" producer="IT"
security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="IT_DE_EPO_SubmissionAcceptance">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_SubmissionAcceptance"
producer="IT" security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="IT_DE_EPO_RelayMD">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_RelayMD" producer="IT"
security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="IT_DE_EPO_Delivery">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Delivery" producer="IT"
security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
```

```
    </Binding>

    <Binding name="IT_DE_EPO_Retrieval">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Retrieval" producer="IT"
security="sign-encrypt-body-header_IT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

<!-- From EE to DE -->

    <Binding name="EE_DE_EPO_Form_A">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_A" producer="EE"
security="sign-encrypt-body-header_EE_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="EE_DE_EPO_Form_B">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_B" producer="EE"
security="sign-encrypt-body-header_EE_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="EE_DE_EPO_SubmissionAcceptance">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_SubmissionAcceptance"
producer="EE" security="sign-encrypt-body-header_EE_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="EE_DE_EPO_RelayMD">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_RelayMD" producer="EE"
security="sign-encrypt-body-header_EE_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="EE_DE_EPO_Delivery">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Delivery" producer="EE"
security="sign-encrypt-body-header_EE_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="EE_DE_EPO_Retrieval">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Retrieval" producer="EE"
security="sign-encrypt-body-header_EE_DE">
```

```
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

<!-- From AT to DE -->
    <Binding name="AT_DE_EPO_Form_A">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_A" producer="AT"
security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="AT_DE_EPO_Form_B">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Form_B" producer="AT"
security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="AT_DE_EPO_SubmissionAcceptance">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_SubmissionAcceptance"
producer="AT" security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="AT_DE_EPO_RelayMD">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_RelayMD" producer="AT"
security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="AT_DE_EPO_Delivery">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Delivery" producer="AT"
security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

    <Binding name="AT_DE_EPO_Retrieval">
      <MEP name="One-Way/Push">
            <Leg number="1" mpc="epo" userService="EPO_Retrieval" producer="AT"
security="sign-encrypt-body-header_AT_DE">
          <Endpoint address="http://someserveringermany.de/gateway"
soapVersion="1.1" />
        </Leg>
      </MEP>
    </Binding>

<!--Defintion of the PModes -->
```

```
      <PMode name="EE_DE_EPO_Form_A" binding="EE_DE_EPO_Form_A" />
      <PMode name="EE_DE_EPO_Form_B" binding="EE_DE_EPO_Form_B" />
      <PMode name="EE_DE_EPO_SubmissionAcceptance"
binding="EE_DE_EPO_SubmissionAcceptance" />
      <PMode name="EE_DE_EPO_RelayMD" binding="EE_DE_EPO_RelayMD" />
      <PMode name="EE_DE_EPO_Delivery" binding="EE_DE_EPO_Delivery" />
      <PMode name="EE_DE_EPO_Retrieval" binding="EE_DE_EPO_Retrieval" />

      <PMode name="AT_DE_EPO_Form_A" binding="AT_DE_EPO_Form_A" />
      <PMode name="AT_DE_EPO_Form_B" binding="AT_DE_EPO_Form_B" />
      <PMode name="AT_DE_EPO_SubmissionAcceptance"
binding="AT_DE_EPO_SubmissionAcceptance" />
      <PMode name="AT_DE_EPO_RelayMD" binding="AT_DE_EPO_RelayMD" />
      <PMode name="AT_DE_EPO_Delivery" binding="AT_DE_EPO_Delivery" />
      <PMode name="AT_DE_EPO_Retrieval" binding="AT_DE_EPO_Retrieval" />

      <PMode name="IT_DE_EPO_Form_A" binding="IT_DE_EPO_Form_A" />
      <PMode name="IT_DE_EPO_Form_B" binding="IT_DE_EPO_Form_B" />
      <PMode name="IT_DE_EPO_SubmissionAcceptance"
binding="IT_DE_EPO_SubmissionAcceptance" />
      <PMode name="IT_DE_EPO_RelayMD" binding="IT_DE_EPO_RelayMD" />
      <PMode name="IT_DE_EPO_Delivery" binding="IT_DE_EPO_Delivery" />
      <PMode name="IT_DE_EPO_Retrieval" binding="IT_DE_EPO_Retrieval" />

</PModes>
```

2.7. Holodeck PMode Data for E-Procurement

The second PMode file is an artificial data set for PEPPOL business process (the CEN BII process 06, Procurement) for two gateways: "C2" and "C3" (corners two and three).

```
<PModes>

  <Producer name="C2">
    <PartyId>C2</PartyId>
    <Role>Buyer</Role>
  </Producer>

  <Producer name="C3">
    <PartyId>C3</PartyId>
    <Role>Seller</Role>
  </Producer>

  <UserService name="OrderingBilling_SubmitOrder_C3">
      <ToPartyInfo>
        <PartyId>C3</PartyId>
        <Role>Seller</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>OrderingBilling</Service>
        <Action>SubmitOrder</Action>
      </CollaborationInfo>
    </UserService>

  <UserService name="OrderingBilling_AcceptOrder_C2">
      <ToPartyInfo>
        <PartyId>C2</PartyId>
        <Role>Buyer</Role>
      </ToPartyInfo>
      <CollaborationInfo>
        <Service>OrderingBilling</Service>
        <Action>AcceptOrder</Action>
      </CollaborationInfo>
    </UserService>

  <UserService name="OrderingBilling_RejectOrder_C2">
```

```
    <ToPartyInfo>
      <PartyId>c2</PartyId>
      <Role>Buyer</Role>
    </ToPartyInfo>
    <CollaborationInfo>
      <Service>OrderingBilling</Service>
      <Action>RejectOrder</Action>
    </CollaborationInfo>
  </UserService>

  <Binding name="OrderingBilling_SubmitOrder_C2_C3">
    <MEP name="One-Way/Push">
      <Leg number="1" mpc="epo"
        userService="OrderingBilling_SubmitOrder_C3" producer="C2" security="sign-
encrypt-body-header_C2_C3">
        <Endpoint address="http://c3.example.com/msh" soapVersion="1.2" />
      </Leg>
    </MEP>
  </Binding>

  <Binding name="OrderingBilling_AcceptOrder_C3_C2">
    <MEP name="One-Way/Push">
      <Leg number="1" mpc="epo"
        userService="OrderingBilling_AcceptOrder_C2" producer="C3" security="sign-
encrypt-body-header_C3_C2">
        <Endpoint address="http://c2.example.com/msh" soapVersion="1.2" />
      </Leg>
    </MEP>
  </Binding>

  <Binding name="OrderingBilling_RejectOrder_C3_C2">
    <MEP name="One-Way/Push">
      <Leg number="1" mpc="epo"
        userService="OrderingBilling_RejectOrder_C2" producer="C3" security="sign-
encrypt-body-header_C3_C2">
        <Endpoint address="http://c2.example.com/msh" soapVersion="1.2" />
      </Leg>
    </MEP>
  </Binding>


<!--Defintion of the PModes -->

  <PMode name="OrderingBilling_SubmitOrder_C2_C3"
binding="OrderingBilling_SubmitOrder_C2_C3" />
  <PMode name="OrderingBilling_AcceptOrder_C3_C2"
binding="OrderingBilling_AcceptOrder_C3_C2" />
  <PMode name="OrderingBilling_RejectOrder_C3_C2"
binding="OrderingBilling_RejectOrder_C3_C2" />

</PModes>
```

3. Proof-of-Concept

The proof of concept is implemented as a series of scripts.

3.1. Installation

The following software packages are needed to run the PoC:

- Python 2.7.
- Twisted Matrix networking framework.
- lxml, a library for processing XML, XPath, XSLT and related.
- dnspython, a DNS client library

The PoC can be downloaded as:

- https://secure.e-codex.eu/gitblit/summary/research%2Fbdx%2Fddpoc.git

The PoC files can be extracted to some folder on your computer and should run from there without further configuration (but I haven't checked completely).

The Python scripts are platform-independent. Windows batch files are provided for the BDMSL server and for the metadata server but would be easily rewritten for other platforms.

3.2. BDX Metadata Location Service

The BDX Metadata Location service specification [BDMSL], like SML [SML], is based on DNS. So NAPTR-U records can be published using any conformant DNS server. The proof-of-concept uses the Twisted Names DNS server. The NAPTR records are encoded in the Twisted Names syntax.

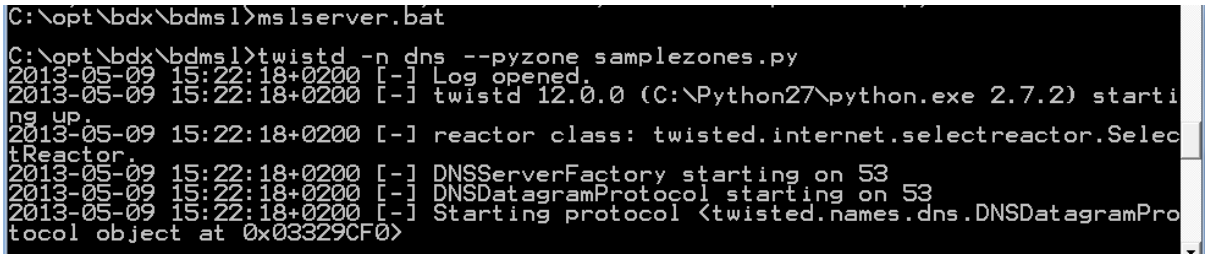

Figure 5 BDX Metadata Location Service

The script *mslserver.bat* runs the Twisted demon with a zone definition file. If you run another DNS server on the computer on which you install the scripts, modify the server to use a different port number.

3.3. Metadata Publication Service

BDX location records provide URI for metadata resources. These resources are obtained using HTTP GET requests as the publication protocol is REST-based.

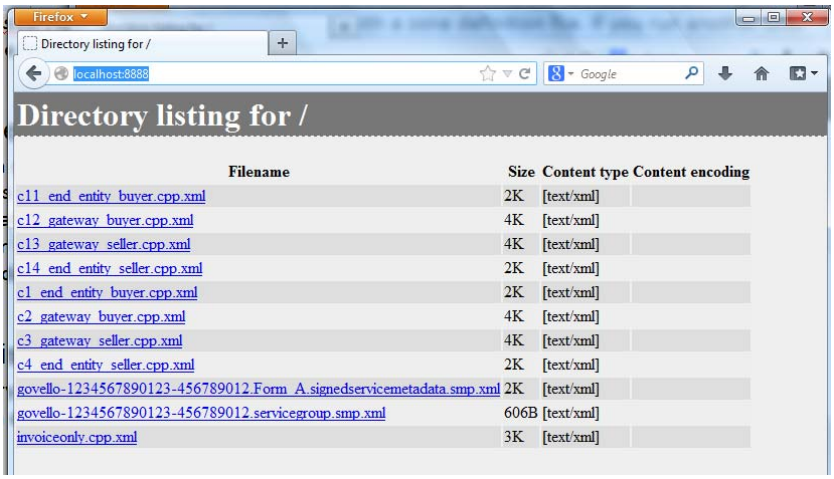

Figure 6 Metadata Server

A production metadata service is likely to store metadata information in a database and to generate XML representations of metadata in SMP, CPP or other formats dynamically. The proof-of-concept includes a minimal metadata service that serves metadata XML files from a directory on port *8888*. It uses the Twisted Web framework and is 9 lines of code.

3.4. Discovery Library

The discovery scripts are written using a number of modules. The modules illustrate implementation options for a discovery library. The software is open source under the EUPL license, http://ec.europa.eu/idabc/eupl.

3.4.1. bdxr.py

This module defines a simple class *MetadataServiceClient* that knows about DNS and the processing of NAPTR-U records.

3.4.2. connect.py

This module defines a *ConnectClient* class that uses the *MetadataServiceClient* and invokes any metadata format-specific processing.

3.4.3. holodeck.py

This module define a *PModeResolver* class that, on initialization, becomes aware of all configured PModes and has ways to search in this set.

3.4.4. smp.py

Processing specific to the SMP XML format.

3.4.5. cpp.py

Processing specific to the CPP XML format.

3.4.6. pmode.py

Defines a placeholder *PMode* class.

4. Discussion and Further Work

The proof-of-concept showcases advanced discovery functionality, which can be summarized as:

- Discovering and matching business capabilities of end entities, relations of end entities to gateways (generalized for multi-corner models).
- Dynamic matching of technical capabilities (minimal)

The main results are:

- To discover end entities, the DNS-based discovery mechanism under development in the OASIS BDXR TC is successfully used.
- To retrieve structured information on entities, the HTTP GET-based mechanism proposed in the BusDox SMP is also used successfully.
- Two candidate capability representation formats (SMP 1.0 and draft CPP v3.0) have been evaluated with respect to these requirements. SMP supports the routing requirement, is tied to a particular PKI security model and assumes transport protocols with few configuration options. The v3.0 CPP (work in progress) is a richer format and supports a wider range of requirements.
- Multiple metadata formats and metadata retrieval protocols can easily coexist. A toolkit can easily add support for additional formats. A client can filter results based on format.

5. References

[BDMSL] D. Moberg, ed. OASIS Business Document Metadata Service Location Version 1.0. BDXR TC Working Draft version 0.2. January 2013. https://www.oasis-open.org/committees/document.php?document_id=47815

[BII04] CEN BII Profile 04. Basic Invoice Only. http://www.cen.eu/cwa/bii/specs/Profiles/ProfileDoc/BII%20profile%2004%20-%20Invoice%20only%20v1.pdf

[CPATK] Joinup CPA Creation Toolkit. http://joinup.ec.europa.eu/software/cpatoolkit/home

[D5.2] e-CODEX D5.2 Annex I Scenario for the Convergence of LSP e-Delivery solutions http://www.e-codex.eu/news-and-media/media/deliverables.html?eID=dam_frontend_push&docID=125

[D5.9] E-CODEX Deliverable D5.9 Concept of Implementation.

[DDREQS] P. van der Eijk. Dynamic Service Discovery Requirements. https://lists.oasis-open.org/archives/bdxr/201301/msg00001.html

[PMTK] EJ van Nigtevecht. E-CODEX Pmode Toolkit version 0.1. https://www.jol.nrw.de/bscw/bscw.cgi/3979785

[RFC4848] L. Deagle. Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS). IETF RFC. April 2007. https://tools.ietf.org/rfc/rfc4848.txt

[SML] PEPPOL BusDox Service Metadata Location Specification ver 1.1.0 https://www.oasis-open.org/committees/document.php?document_id=47488

[SMP] PEPPOL BusDox Service Metadata Publisher Specification ver 1.1.0 https://www.oasis-open.org/committees/document.php?document_id=47489