Specification

Acronym: PEPPOL

Grant Agreement number: 224974

Title: Pan-European Public Procurement Online

PEPPOL Transport Infrastructure
Secure Trusted Asynchronous Reliable Transport (START)

Version: 1.01

Authors:

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	Author	Organisation	Description
1.0	20100215	Paul Fremantle	NITA/WSO2	First version (pending EC approval)
1.01	20101001	Klaus Vilstrup Pedersen	DIFI	EC Approved

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This deliverable contains original unpublished work except where clearly indicated otherwise.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.

Contributors

Organisations

DIFI (Direktoratet for forvaltning og IKT)[1], Norway, www.difi.no
NITA (IT- og Telestyrelsen)[2], Denmark, www.itst.dk
BRZ (Bundesrechenzentrum), Austria
Consip, Italy

Persons

Bergthór Skúlason, NITA
Gert Sylvest, NITA/Avanade
Hans Guldager Knudsen, NITA/Lenio
Jens Jakob Andersen, NITA
Klaus Vilstrup Pedersen, DIFI
Maria Raffaella Migliorini, CONSIP
Mike Edwards, NITA/IBM
Mikkel Hippe Brun, NITA
Paul Fremantle, NITA/WSO2 (editor)
Philip Helger, BRZ
Thomas Gundel, NITA/IT Crew

[1] English: Agency for Public Management and eGovernment

[2] English: National IT- and Telecom Agency

Table of Content

1 Introduction

1.1 Objective

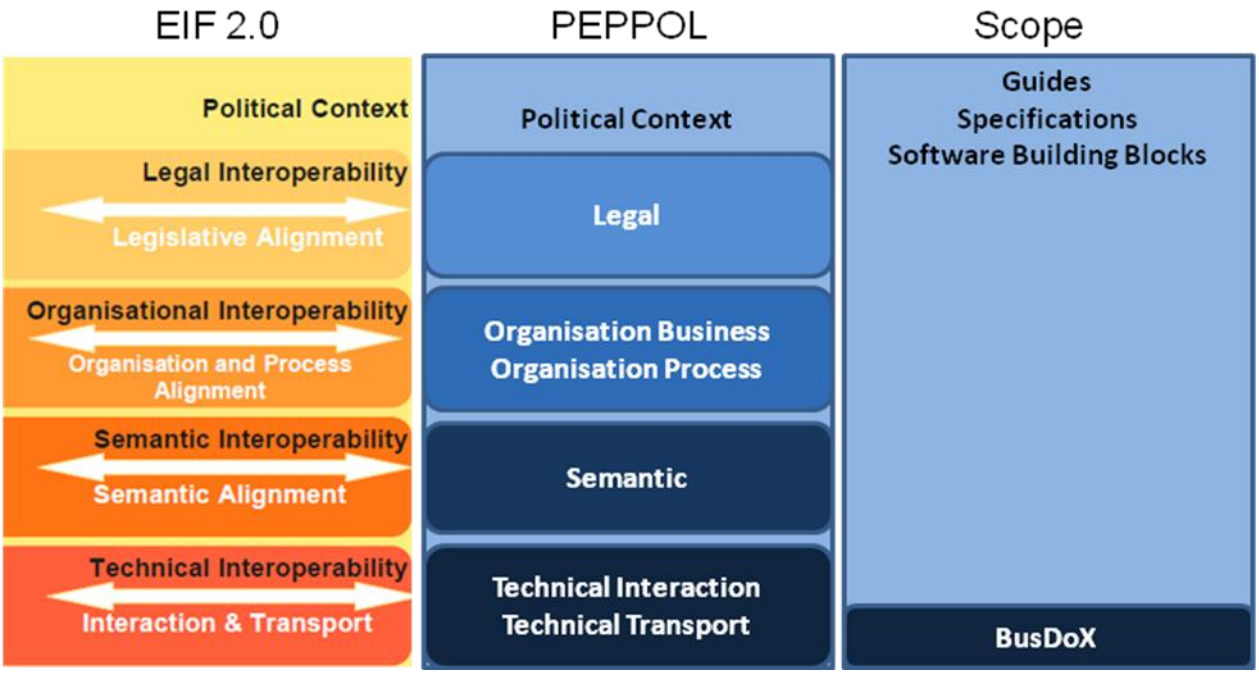
This document describes the SOAP-based profile that is used by Business Document Exchange Network (BUSDOX) Access Points to communicate and the SAML 2.0 assertions that are used in that communication. This profile offers secure and reliable delivery of messages between Access Points. This is currently the only required transport profile in BUSDOX.
BUSDOX Access Points communicate in a peer-to-peer model across the internet to form the BUSDOX infrastructure. Each Access Point derives the endpoint addresses of other BUSDOX Access Points through the BUSDOX Service Metadata Publishing Infrastructure. BUSDOX Access Points may communicate via optional BUSDOX Transport Profiles, but they must always offer a START (Secure Trusted Asynchronous Reliable Transport) endpoint by which any other Access Point may communicate.
In order to instantiate a working network, certain profile information is expected. For example, an instance of BUSDOX is the PEPPOL infrastructure, which includes governance models, certificate rules, identifier formats, and other profiling. This specification therefore excludes such profiling information.
The Secure Trusted Asynchronous Reliable Transport (START) defines a secure, reliable profile using a set of well-known standards:

- SOAP 1.1
- WS-Addressing 1.0
- WS-Security 1.1
- WS-Transfer as a standard approach to accessing the message channels
- WS-ReliableMessaging 1.1
- SAML 2.0

This specification profile describes the usage of these standards to support the requirements of BUSDOX. In particular the usage of these standards is restricted to certain patterns to enable interoperability to be achieved.

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.



1.3 Goals and non-goals

The goal of this profile is to support a high level of assurance and proof-of-delivery across the BUSDOX Infrastructure. The profile is designed to:

- Be a single profile that implementers can build and therefore gain access to BUSDOX with no further requirements to implement other transport profiles.
- Clearly state the transport level requirements in a single document.
- Define a simple, interoperable communications pattern that APs can use to communicate.
- Define the message exchange formats and patterns clearly
- Ensure that messages are reliably delivered between APs, including providing the prerequisites for logging and proof-of-delivery for messages at the transport level
- Ensure confidentiality of messages using transport-level encryption
- Ensure integrity and authenticity of received messages by signature validation
- Support transfer of messages that are opaque to the Access Point
- Define a set of BUSDOX specific headers within the message envelope so Access Points can forward/transfer messages without requiring access to the business message.
- Establish a common format for representing authentication events in BUSDOX infrastructure in the form of SAML 2.0 assertions / tokens.
- Recipients can assume that senders have been authenticated by their token Issuers and obtain further proof via cryptographically signed tokens.

The Profile does NOT address:

- The verification of certificates, format of participant identifiers, and other details required to create a full instantiation of BUSDOX.
- Communication with BUSDOX Service Metadata services or Security Token Services.

- Use of SAML 2.0 tokens for other purposes is not in scope for this note; these may include using tokens to express sender-side certificate validation results or tokens issued by BUSDOX Security Token Services for authenticating the Token Issuers themselves.
- The authentication process itself is not in scope for this profile; senders may use whatever credentials accepted by the Token Issuers.

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.
For common terms used in these specifications, please see [BDEN-CDEF].

Notational conventions
For notational conventions, see [BDEN-CDEF].

Normative references
[BDEN-CDEF] Business Document Exchange Network - Common Definitions, CommonDefinitions.pdf
[WSS-1.1] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
[WSS-STP] "Web Services Security: SAML Token Profile 1.1" http://docs.oasis-open.org/wss/v1.1/wss-v1.1-errata-os-SAMLTokenProfile.pdf
[WS-T] "Web Services Transfer (WS-Transfer)", W3C Working Draft 24th September 2009, http://www.w3.org/TR/2009/WD-ws-transfer-20090924/
[WSRM-1.1] "Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1", http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01-e1.html
[WSA-1.0] "Web Services Addressing 1.0 - Core" (http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/) and "Web Services Addressing 1.0 - SOAP Binding", http://www.w3.org/TR/ws-addr-soap/
[SOAP-1.1] "SOAP Version 1.1 ", http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[RFC-2119] "Key words for use in RFCs to Indicate Requirement Levels", http://www.ietf.org/rfc/rfc2119.txt
[SAML-2.0 assertion] ”Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0”, http://docs.oasis-open.org/security/saml/v2.0/
[MTOM] "SOAP Message Transmission Optimization Mechanism", http://www.w3.org/TR/soap12-mtom/
[XML-DSIG] XML Signature Syntax and Processing (Second Edition), http://www.w3.org/TR/xmldsig-core/

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

1.5 Namespaces

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

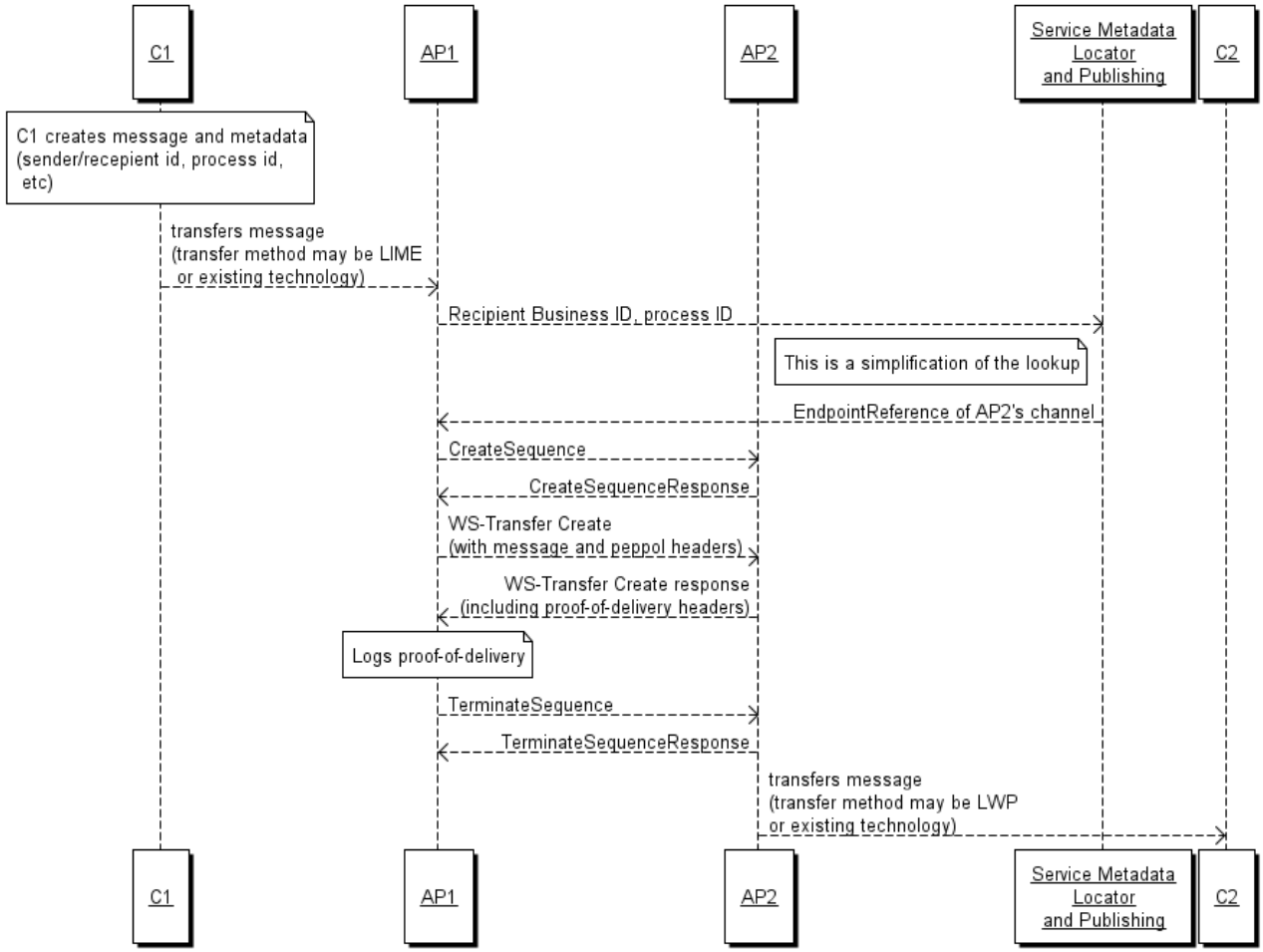
Namespace	Namespace

Prefix	
wsa	http://www.w3.org/2005/08/addressing
s	http://schemas.xmlsoap.org/soap/envelope/
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsrm	http://docs.oasis-open.org/ws-rx/wsrm/200702
start	http://busdox.org/transport/start/1.0/
ids	http://busdox.org/transport/identifiers/1.0/
saml	urn:oasis:names:tc:SAML:2.0:assertion
ds	http://www.w3.org/2000/09/xmldsig#

2 Overview

The BUSDOX Secure Trusted Asynchronous Reliable Transport (START) provides a secure reliable approach for messages to be delivered from one BUSDOX Access Point (AP) to another. From the perspective of an AP the business messages are (or may be) opaque, so the message transfer includes headers that support the routing of messages.
A typical flow might be:

- Message is created by Company C1, using either existing systems or a BUSDOX Lightweight Message Exchange Profile client.
- The message is transferred to Access Point AP1, including information required such as:
    - Recipient Identifier and identifier type
    - Sender Identifier and identifier type
    - Document identifier and process identifier
- AP1 uses the BUSDOX Metadata Lookup and publishing specifications and services to identify the endpoints and keys for the Access Point handling the recipients messages (AP2).
- AP1 gets or creates any tokens it needs to include in the message transfer, i.e. X509 certificates and SAML 2.0 token.
- AP1 adds the correct SOAP headers containing recipient, sender and document type information.
- AP1 signs the message.
- AP1 uses WS-ReliableMessaging and TLS to transfer the message to AP2 as part of a reliable exchange
- AP2 responds with a signed proof-of-delivery message to AP1 (using the WS-Security [WSS-1.1] Signature Confirmation method).
- Finally AP1 logs a signed proof-of-delivery of the message



In this model, AP1 is acting as a Source Access Point (SrcAP) and AP2 is acting as a Destination Access Point (DestAP). The expectation is that most Access Points will act as both SrcAP and DestAP, however this is not required by the specifications.

3 Specification Profile Details

The following requirements apply to the BUSDOX START Profile.

3.1 Use of WS-Transfer

The WS-Transfer[3] [WS-T] specification is used here to support sending messages. This is done for two reasons: WS-Transfer matches the semantics of a service that is able to handle XML messages of any type, and also to increase the commonality between this profile and the BUSDOX Lightweight Message Exchange Profile (LIME).

3.2 BUSDOX defined headers

Please see [BDEN-CDEF].

3.3 Message Exchange

This profile uses the WS-Transfer [WS-T] standard to transfer any XML document from one Access Point to another. The reliability of this exchange is guaranteed by the use of WS-ReliableMessaging 1.1.

Prerequisites for communication

Before an Access Point can deliver a message to another Access Point, the SrcAP MUST have the following information, which it MAY find in the BUSDOX Service Metadata Publishing document:

- The WS-Addressing EndpointReference for the *Destination Message Channel (DestMC)* of the DestAP.

Destination Message Channel

In order for a SrcAP to send a message to a DestAP, the DestAP MUST expose a Destination Message Channel (DestMC). The DestMC is a WS-Transfer endpoint that supports the CREATE operation of the WS-Transfer specification. The Endpoint Reference of this channel is available in a BUSDOX Service Metadata Publishing document.

Delivery of BUSDOX messages

The SrcAP MUST use WSRM 1.1 to ensure reliable delivery of messages.
The SrcAP MUST send the message to be transmitted as the body of the WS-Transfer CREATE operation.

[3] It is expected that future versions of START or errata will update to the Final version of WS-Transfer as and when this becomes available.

The Create message request MUST include the BUSDOX defined headers in the SOAP Header:

- ids:RecipientIdentifier
- ids:ChannelIdentifier
- ids:SenderIdentifier
- ids:DocumentIdentifier
- ids:ProcessIdentifier
- ids:MessageIdentifier
- Other headers MAY be included.

Faults

The DestAP can return faults in five circumstances:

- Firstly, it may have a “full channel”. This indicates that the SrcAP should retry at a later time. This is enabled to allow the server to deal with the case where an onward system is not receiving messages.
- Secondly, the endpoint may not be recognized. This may happen in cases where the SrcAP has cached the endpoint, or there is incorrect/out-of-date information available in the metadata. In this case, the SrcAP should refresh the metadata by re-requesting it from the Service Metadata Lookup and Publishing systems.
- Thirdly, in the case where there is a security processing error (SAML assertion or signature not validated).
- Fourthly in the case where the document type is not accepted for this recipient.
- Finally, there are other server errors that may fall outside those identified above.

The faults used are as follows:

Channel Full Fault

[action]	http://busdox.org/2010/02/channel/fault
Code	s:Sender
Subcode	bden:ChannelFull
Reason	The channel is not accepting messages for this destination
Detail	As detailed by the AP

Unknown Endpoint

[action]	http://busdox.org/2010/02/channel/fault
Code	s:Sender
Subcode	bden:UnknownEndpoint
Reason	The endpoint is not known
Detail	As detailed by the AP

Security Error

[action]	http://busdox.org/2010/02/channel/fault
Code	s:Sender
Subcode	bden:SecurityFault
Reason	There is a security error in processing this request
Detail	As detailed by the AP

Document Type Not Accepted

[action]	http://busdox.org/2010/02/channel/fault
Code	s:Sender
Subcode	bden:DocumentTypeNotAccepted
Reason	The recipient does not accept documents of this type.
Detail	As detailed by the AP

Server Error

[action]	http://busdox.org/2010/02/channel/fault
Code	s:Sender
Subcode	bden:ServerError
Reason	ServerError
Detail	As detailed by the AP

3.4 SOAP 1.1

APs MUST use SOAP 1.1 for all message exchange.
Messages MUST use the document/literal style.

3.5 Use of MTOM

The Message Transmission Optimization Mechanism (MTOM) is an extension to SOAP that supports effective transmission of binary data using the XML Optimized Packaging (XOP) standard. Access Points MUST support MTOM when acting as a service endpoint – that is while receiving messages from another Access Point. APs MAY send messages using MTOM.

3.6 Use of HTTP

Please see Common Definitions section 3.8

3.7 WS-Addressing 1.0

Please see Common Definitions section 3.10

3.8 WS-Transfer

The AP Message Channel interface is based on WS-Transfer [WS-T]. The services offered by the AP MUST be a SOAP/HTTP binding using document/literal style of the interfaces defined in the WS-

Transfer WSDL. The current specification is based on a working draft of the WS-Transfer specification which is WS-I Basic Profile Compliant[4].
The **[Body]**/wst:Create@Dialect attribute MUST NOT be used in the START profile.
The BUSDOX message resource that is created on the receiving AP MUST be the same representation as the BUSDOX message sent by the sending AP. The WS-Transfer specification indicates that in this situation the WS-Transfer CreateResponse body SHOULD be empty other than the ResourceCreated element. This specification strengthens this: the CreateResponse body MUST only contain the ResourceCreated element as specified by the WS-T specification. Extension elements MUST NOT be present.
The Destination Message Channel MUST offer the WS-Transfer CREATE operation to all other Access Points. Other operations are not expected to be exposed as part of the START profile. In certain circumstances the WS-Transfer PUT, DELETE and GET operations MAY be exposed at a BUSDOX endpoint, but their usage is not covered by this specification and MUST NOT be required. These operations SHOULD be protected from general access.

3.9 Security

The same security profile is to be used for both the WS-RM 1.1 protocol messages as well as the WS-Transfer 'business' messages.
All SOAP request and response messages exchanged via this profile (including messages that do not carry business document payloads) MUST be secured using the mechanisms described below.

The <wsse:Security> Header Block
This section defines elements and processing rules for SOAP message security by profiling the <wsse:Security> header block defined in [WSS-1.1]. Processing rules defined in [WSS-1.1] and [WSS-STP] MUST be followed unless stated explicitly otherwise below.
A single <wsse:Security> header block MUST be present and MUST have a mustUnderstand attribute with the logical value of true. Further, it MUST include a <wsu:Timestamp> with a <wsu:Created> element.
The value of the <wsu:Created> element SHOULD be within an appropriate offset from local time. In absence of other guidance, a value of 5 minutes MAY be used.
If the <wsu:Timestamp> element includes a <wsu:Expires> element, the receiver MUST ensure that his local time is before that time.
To prevent message replay, receivers SHOULD maintain a message cache, and check received MessageIdentifier values against the cache. The cache timeouts are out of scope for this specification and are to be defined by local deployment or other governance models.

Message Authentication and Integrity
Authentication and integrity of messages is established by means of digital signatures applied to the SOAP message. The sender MUST create and include a single <ds:Signature> element in the <wsse:Security> header block and this signature MUST reference:

- The SOAP <Body> element
- All security tokens embedded directly under the <wsse:Security> element (see next section for special rules regarding SAML assertions).

[4] It is planned that as soon as the WS-Transfer specifications are updated within the W3C working process the BUSDOX START profile will be amended to use the finalized form.

- All SOAP header blocks in the message defined in this profile, including any all BUSDOX-namespaced headers, all WS-Addressing and any WS-ReliableMessaging headers.

The signature MAY reference other elements including header blocks not mentioned in this profile. The certificate MUST be included in a <wsse:BinarySecurityToken> element in the security header. In the message signature, the <ds:KeyInfo> element MUST refer to this token via a <wsse:SecurityTokenReference>.

Including a SAML assertion in the Security header

The sender Access Point MUST include a SAML 2.0 assertion in the security header which contains the identity of the sender and details of the sender authentication (see SAML profile below). Authentication assertions MUST be signed by the sender by including first a <wsse:SecurityTokenReference> in <wsse:Security> header block, and then referencing this element from the message signature using a <ds:Reference> element. The security token reference MUST include a <wsse:KeyIdentifier> with a ValueType of "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID" and specify the ID of the SAML assertion. The <ds:Reference> element MUST use a transform algorithm set to "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsssoap-message-security-1.0#STR-Transform".

Types of SAML assertions

As stated above, SAML assertions are used in this profile to state the sender identity and details of the sender authentication to the recipient. Two different types of SAML assertions are allowed in this profile depending on who authenticated the sender.

The first type is called "sender-vouches" assertions and these are used in scenarios where the sender Access Point itself has authenticated the sender. Thus, the sender Access Point issues and signs the token thereby vouching for the sender identity to the recipient. The recipient Access Point needs to trust the sender Access Point regarding authentication of the sender.

The other type is called "holder-of-key" assertion, and are used in scenarios where trusted third parties (e.g. Identity Providers / Security Token Services) have authenticated the sender on request from the sender Access Point. Here the SAML assertion is issued and signed by the third party (typically a Security Token Service), which must then be trusted by the recipient. The assertion / token MUST be restricted by the external issuer (STS) for use only by the particular (requesting) Access Point by including a subject confirmation element of "holder-of-key" type, which refers the sender Access Point's digital certificate. This means that the sending Access Point has to prove possession of the associated private key in order for the recipient Access Point to trust the token embedded in the message, for example by signing the SOAP message (WS-Security) using its private key and referencing the assertion from the associated signature element.

Responses

Responses are signed with the X.509 certificate of the receiver AP. The full certificate MUST be included (as a base64 encoded value) in a <wsse:BinarySecurityToken> element in the security header. In the message signature, the <ds:KeyInfo> element MUST refer to this token via a <wsse:SecurityTokenReference>.

Validation

The receiver of either request or response messages MUST validate the message signature and security tokens (X.509 certificates and SAML assertions) including issuer signature, test of validity period and trust in the token issuer. Depending on local policy, the receiver SHOULD check revocation status of any certificates used to sign the message and tokens.

The SrcAP SHOULD validate that the Subject Unique Identifier of the certificate used to sign the response messages matches the Subject Unique Identifier published in the Service Metadata Publishing.
When validating a signed response message, the sender Access Point SHOULD check that the certificate in the response matches the metadata received from the Service Metadata Publisher. This is done by comparing the subject common name[5] in the certificate to the value stated in the metadata. This check ensures that only the legitimate Access Point stated in the service metadata will be able to produce correct responses.

Use of HTTPS
Messages MUST be encrypted using one-way TLS. The SOAP envelope or body elements MUST NOT be encrypted using WS-Security with the exception of encrypting SAML assertions as stated above.

Signature Confirmation
In order to provide a high-level of proof-of-delivery, the WS-Security 1.1 [WSS-1.1] "Signature Confirmation" MUST be used (See section 8.5 of [WSS-1.1]. The SrcAP SHOULD log the SignatureConfirmation element for future reference.

Additional Processing Rules for holder-of-key Assertions
When the authentication assertion has a subject confirmation method being "holder-of-key" it means that the sending AP must prove possession of the key mentioned in the assertion's <SubjectConfirmationData> in order for the recipient AP to rely on the assertion. The proof-of-possession of the key will be achieved via the message signature and provides additional assurance that the sender AP is allowed to use the assertion in a web service invocation.
In this profile, a holder-of-key Assertion in the <SubjectConfirmationData> element MUST include a key that can be used to verify the message signature. Thus, the same key used for message authentication and integrity is used to confirm the right to use the assertion for message authorization purposes.
The message signature (i.e. the <ds:Signature> element) MUST refer to the token with the subject confirmation key within the <ds:KeyInfo> element.
The receiver MUST check that the message is signed by the same key mentioned in the assertion's subject confirmation element before relying on the assertion content.

3.10 WS-ReliableMessaging 1.1

In this profile, both the SrcAP and the DestAP act as both RM Sources (RMS) and RM Destinations (RMD). Web Services Reliable Messaging (WSRM) is used to ensure the delivery of both the requests (business messages) as well as the responses.
A single WSRM sequence MAY be used to send multiple messages where the Sender Identifiers and/or Recipient Identifiers are different. This supports the case where an AP may have a number of messages all destined to be handled by a remote AP, but for different recipients. In this case the AP may re-use the WSRM sequence. In addition, the WSRM sequence MAY be held open in the case that other messages may require delivery before either end times out the sequence. The normal

[5] It is assumed that the Subject Common Name element alone is unique to Access Points across certificate renewals. The Common Name is returned from the Service Metadata Publisher as part of the signed metadata.

WSRM sequence timeout model applies as long as the further restrictions outlined below are also conformed to.

Successful initiation of the Reliable messaging sequences.
The SrcAP MUST successfully create a Sequence as the first communication with the DestAP.
The same endpoint reference for the DestMC MUST be used for the wsrm:CreateSequence message.
The wsrm:CreateSequence message MUST include an Offer Sequence, and the DestAP MUST accept the Offered Sequence.
The DestAP MUST send all responses via the Offered sequence.

Delivery Assurances
The DestAP MUST implement the <wsrmp:ExactlyOnce> delivery assurance on the created sequence.
The SrcAP MUST implement the <wsrmp:ExactlyOnce/> delivery assurance on the offered sequence.
The DestAP SHOULD implement the <wsrmp:InOrder/> delivery assurance on the created sequence.

Reliable exchange behaviour
The following requirements ensure that the reliable messaging framework effectively delivers messages from SrcAP to DestAP, or leaves the Access Points with a clear status of the transmitted messages.
Both RMSs MUST continue to resend unacknowledged messages until the WSRM sequence is closed or terminated.
Both RMDs MUST accept unacknowledged messages until the WSRM sequence is closed or terminated.
When a message is retransmitted, the wsa:MessageID MUST be the same as the original transmission of the message.
The SrcAP MAY send a single message per sequence, or it MAY send multiple messages per sequence. However, it is RECOMMENDED that sequences are timed out when inactive to prevent resource utilization on a remote system.
If the SrcAP is ending a sequence, and it has received and logged a complete acknowledgement for the sequence, it MUST send a TerminateSequence with the LastMsgNumber. It MUST repeat this until it receives a response indicating that the Sequence is terminated.
The DestAP MUST send a final acknowledgement back in response to this TerminateSequence.

If the SrcAP has not received a complete acknowledgement and for some reason is attempting to end the sequence, it MUST send a CloseSequence including LastMsgNumber, and it MUST keep requesting acknowledgement until it has a Final acknowledgement.

The SrcAP MUST send either a CloseSequence or TerminateSequence message for every sequence.
If the DestAP decides to time out a sequence it MUST close the sequence to allow the SrcAP the opportunity to access the final acknowledgement.
To ensure maximum interoperability, the SrcAP's RMS MUST NOT use wsa:ReferenceParameters in the AcksTo endpoint reference.
The SrcAP's RMS MUST support piggybacked acknowledgements.
The SrcAP MUST sign the body of the WS-Transfer Create message together with the WSRM Sequence header, and the corresponding signature MUST be logged in persistent storage. The

DestAP MUST sign the WSRM acknowledgements in conjunction with the WS-Addressing RelatesTo header, and the SrcAP SHOULD keep a persistent log of the latest acknowledgement and signature. Security tokens are only required in SOAP messages that actually carry business document payload.

Ensuring security of the WS-ReliableMessaging sequence
There are a number of potential attacks against the WSRM 1.1 sequence. The following categorizes them and the protections enforced:

Denial of Service by Creating multiple sequences

In order to prevent a random client creating a DoS attack against an access point, the WSRM Create messages will be signed. Any messages that are not signed can be discarded before processing.

Sequence Attack

If an attacker can find or guess the sequence identifier of a sequence, then in theory they can perform DoS attacks (sending incorrect messages, closing or terminating the sequence incorrectly). This is normally protected against using the WSRM UseSequenceSSL or UseSequenceSTR capabilities. However, in case of a specific instance of the BUSDOX infrastructure, any attacker would need to have a certificate issued in the context of that specific instance. In addition the business message is also validated.

4 Infrastructure Messages

The BUSDOX Profile supports the concept of infrastructure messages that are used to:

- Test systems are working
- Validate basic interoperability between Access Points
- Provide a simple starting point for validating communications with newly joined participants.

In this specification there is only one infrastructure message: Ping. This message is loosely modeled on the Internet Ping message. Any BUSDOX Access Point can send a Ping message to any other BUSDOX Access Point, and the receiving AP SHOULD respond with a PingResponse message. In the case where Ping messages are taking an unacceptable load on an Access Point (for example in a suspected Denial of Service attack), then the receiving AP MAY drop Ping messages. It is expected that a governance system (which is out of scope for this profile) would be used to revoke the certificate of any misbehaving Access Points.

4.1 Ping Message

The BUSDOX START Ping Message is targeted at a receiving Access Point using the transport infrastructure rather than to an end participant Id. Therefore the message cannot have real participant identifiers. Instead the BUSDOX profile defines two well-known participant identifiers that are to be used with infrastructure messages.
The normal behaviour that is expected is that an Access Point would aim to test communications for a given participant's Access Point. The Sender AP should look up any valid SMP entry of the Participant, and then send the Ping Message to the Endpoint Reference listed for that entry. Although the endpoint is located by searching for a given document exchange type, the receiving AP SHOULD respond to the Ping irrespectively. Therefore BUSDOX endpoints SHOULD support at least two types of business message - the real business message specified in the SMP and the Ping message.

In some cases (for example where an Access Point is newly added), there may be no relevant SMP entries. In this situation the Endpoint Reference to which the message is sent is passed out of band.

Ping Message Header Values
The message headers are below.

- ids:RecipientIdentifier

- This must have a fixed value of:

```
<smp:ParticipantIdentifier scheme=”busdox-actorid-transport”>
busdox:recipient
  </smp:ParticipantIdentifier>
```

-
- ids:ChannelIdentifier

- This value is found through the normal lookup mechanisms of SML and SMP
- ids:SenderIdentifier

- This must have a fixed value of:

```
<smp:ParticipantIdentifier scheme=”busdox-actorid-transport”>
        busdox:sender
</smp:ParticipantIdentifier>
```

-
- ids:DocumentIdentifier

```
<DocumentIdentifier scheme=”busdox-docid-qns”>
     busdox:ping
</DocumentIdentifier>
```

- ids:ProcessIdentifier

```
<ProcessIdentifier scheme=”busdox-procid-transport”>
     Busdox:noprocess
</DocumentIdentifier>
```

-
- ids:MessageIdentifier: As defined in [BDEN-CDEF]

Business Message
The Ping business message contents of the WS-T Create Body MUST be:
<start:Ping/>

Receiving AP Response
The receiving Access Point MUST respond with a correct CreateResponse. There is no business-level response to a PING.

Authentication
The aim of the Ping message is to test as much of the BUSDOX infrastructure as possible. Therefore, although there is no business need for a SAML token to be present, the Ping message SHOULD have a token present. The SAML token, if included in a Ping message, MUST have the Subject identity be the same as the Issuer, that is, the identity of the Access Point. If the Ping message has no attached

SAML Token, the receiving AP MAY discard it and fault, or may respond with an empty CreateResponse.

5 SAML 2.0 assertion profile

This document outlines the requirements for SAML 2.0 assertions in BUSDOX. The assertions are issued upon authentication of Sending Parties and vouches for the sender identity and assurance level to the recipient. In the description below, these actions are performed by a logical role named "(Authentication) Token Issuer". Access Points may play this role themselves, since they have a close relationship with their senders, or they may out-source it to third-party Security Token Services.

5.1 Assumptions

- Authentication Token Issuers are able to authenticate senders (whom they receive messages from).
- Mechanisms have been established that allows recipients to trust Token Issuers; specifically, they should be able to validate the Token Issuer's signatures on tokens, and trust the claims provided in the tokens to be correct.
- Common definitions of authentication levels are established in BUSDOX; this caters for different authentication mechanisms between Token Issuers and senders. By defining common authentication levels (e.g. 1-4) and criteria for mapping existing authentication methods to these levels, the recipient can make an informed decision regarding the risk of accepting the message without knowing the details of the sender-Token Issuer relation.
- Common syntax and semantics for identifiers have been established. As identifiers will vary with application domains, they are left to deployment profiles to specify.
- A Token Issuer is able to map the identity of sender (as established via the private authentication) to the sender's identifier.
- Token Issuers are capable of issuing (signing) SAML tokens upon authentication of senders; the tokens are probably not requested and returned via SAML protocols or WS-Trust if it is performed as an internal operation in an Access Point. However, when an Access Point delegates token issuance to an external STS, WS-Trust may be used.
- Servers in the BUSDOX infrastructure have reasonably synchronized clocks; some security checks rely on timestamps and this is problematic if clocks drift too much. A time synchronization policy will be defined (e.g. stratum 2[6] is required).

5.2 SAML Assertion Profile

The main content of a SAML 2.0 assertion issued by a Token Issuer upon authenticating a sender is:

- The subject (sender) identity identifier
- Identity and signature of the token issuer
- Time of authentication
- Strength of authentication method

[6] NTP (Network Time Protocol) servers are organized in a hierarchy where each level is called Stratum. Stratum-0 are devices such as atomic clocks, Stratum-1 are computers attached to Stratum-0 devices and Stratum-2 are computers that send NTP requests to Stratum-1 computers. For details see http://en.wikipedia.org/wiki/Network_time_protocol

- Life time of token
- Audience of the token (where can the token be used)
- Subject confirmation (how can a presenter of a token demonstrate that he is authorized to use the token)

See section 7 for a non-normative SAML token sample.

Authentication Assertion Profile

Below is given a SAML 2.0 Assertion profile that aims to represent the above information with the exception of identifier formats which are left to deployment profiles to decide. Flexibility and choices in SAML have intentionally been limited in order to simplify implementation and increase chances of interoperability:

- The Assertion MUST be a SAML 2.0 Assertion
- The <Assertion> element SHOULD have an id attribute containing a cryptographically random value between 128 and 160 bytes in length.
- The Issuer element MUST be present and contain appropriate BUSDOX identifier of the issuer.
- The NameID format MUST be urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified.
- The Subject element MUST be present and contain appropriate BUSDOX identifier of the sender.
- The Format attribute MUST be set to the ParticipantIdentifierType URI and the value to the ParticipantIdentifierValue defined in Service Metadata Publishing document.
- The assertion MUST be digitally signed by the issuer; the signature MUST include the issuer's certificate as a PEM base 64 encoded X509 DER value in the <ds:KeyInfo> element.
- Assertions MUST NOT be encrypted at the SAML level; this will be handled by the transport protocols.
- An Assertion MUST NOT contain an <AuthzDecisionStatement>.
- An Assertion MUST contain exactly one <AuthnStatement> and one <AttributeStatement>.
- The <AuthnStatement> element MUST have an "AuthInstant" attribute specifying the time authentication occurred.
- The subject element MUST contain at least one <SubjectConfirmation> sub-element containing a Method of "urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" or "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches". In case of "holder-of-key", the element MUST again have two sub-elements:
    - A NameID with format attribute set to "urn:oasis:names:tc:SAML:2.0:nameid-format:entity" and value indicating the ID of the Token Issuer. This indicates that the assertion is used by the Access Point (included in SOAP message) on behalf of the sender.
    - A <SubjectConfirmationData> element with xsi:type saml2:KeyInfoConfirmationDataType and a sub-element referring to the issuer's certificate. The <SubjectConfirmationData> element MUST have an "NotOnOrAfter" attribute defining when the assertion expires. A recipient MUST reject the assertion after this time.
- Advice elements MAY safely be ignored by implementations.

SAML Attribute Encoding

- All attribute names defined in this profile MUST be prefixed with “urn:eu:busdox:attribute”; recipients are not required to process attributes not defined in this profile.
- The <AttributeStatement> MUST contain an attribute stating the level of authentication. The attribute name must be “urn:eu:busdox:attribute:assurance-level” and the value an integer in the range 1-4. BUSDOX will use the assurance levels defined by NIST in the Electronic Authentication Guide, publication 800-63 (http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf). Level 1 is the lowest assurance level and 4 is the highest.
- The <NameFormat> XML attribute on <Attribute> elements MUST be: urn:oasis:names:tc:SAML:2.0:attrname-format:basic
- The <FriendlyName> XML attribute MAY be used but implementations SHOULD NOT rely on it.
- All attribute values MUST be simple text strings with type "xs:string".
- Encrypted attributes MUST NOT be used.

6 Appendix A

6.1 XML Schema for message headers

For an XML Schema for the SOAP header identifiers, see [BDEN-CDEF].

6.2 XML Schema for "Ping" message

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="http://busdox.org/transport/start/1.0/"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://busdox.org/transport/start/1.0/"
  version="1.0.0">

  <element name="Ping" type="tns:PingType" />

  <!-- Length is 0 - only use PING as a marker -->
  <simpleType name="PingType">
    <restriction base="string">
      <length value="0" />
    </restriction>
  </simpleType>

</schema>
```

7 Appendix B: Non-normative SAML token example

7.1 "Sender-vouches" subject confirmation

In the first example, the SAML token has been issued by the sending Access Point upon its authentication of the sender. It contains a subject confirmation element of type "sender-vouches" meaning that the sending Access Point vouches for the identity of the sender specified in the assertion's Subject element. Note also that the assertion contains an assurance level attribute stating the level of assurance in the claimed identity (e.g. authentication strength).

```
<saml:Assertion ID="a12312312312312372975934659137459
871324587613845613984756981346598314634513451345"
  IssueInstant="2001-12-31T12:00:00"
  Version="2.0"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
      format:unspecified">
     http://SomeAcessPoint.busdox.org</saml:Issuer>
  <ds:Signature>
```

```
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="#idvalue31231231231312">
        <ds:Transforms>
          <ds:Transform Algorithm=
            "http://www.w3.org/2000/09/xmldsig#envelopedsignature"/>
        </ds:Transforms>
        <ds:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
                    TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
      </ds:Reference>
        </ds:SignedInfo>
     <ds:SignatureValue>
      x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHa
           EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen
           w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
     </ds:SignatureValue>
     <ds:KeyInfo>
       <ds:X509Data>
           <!-- The Access Point's certificate -->
      <ds:X509Certificate>
      MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
      MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
      F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2l...
           </ds:X509Certificate>
       </ds:X509Data>
     </ds:KeyInfo>
  </ds:Signature>

  <saml:Subject>
        <!-- Here comes a NameID indicating the participant identifier of the
sender -->
        <saml:NameID
Format="http://busdox.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifie
r/1.0/">
           0010:5798000000001
        </saml:NameID>

        <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:sender-
vouches"/>
  </saml:Subject>

  <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
      <saml:AuthnContext>
           <saml:AuthnContextClassRef>
              urn:oasis:names:tc:SAML:2.0:ac:classes:X509
           </saml:AuthnContextClassRef>
      </saml:AuthnContext>
  </saml:AuthnStatement>

  <saml:AttributeStatement>
    <!-- Assurance Level Attribute -->
    <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
        Name="urn:eu:busdox:attribute:assurance-level">
```

```
        <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

7.2 "Holder-of-key" Subject Confirmation

In the next example, the SAML token has been issued by an external Security Token Service (STS) and not by the sender Access Point. This is useful in scenarios where Access Points do not authenticate senders themselves by out-source this to an external Identity Provider.

```
<saml:Assertion ID="a12312312312312372975934659137459
871324587613845613984756981346598314634513451345"
  IssueInstant="2001-12-31T12:00:00"
  Version="2.0"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
      format:unspecified">
     http://SomeTokenService.busdox.org</saml:Issuer>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="#idvalue31231231231312">
        <ds:Transforms>
          <ds:Transform Algorithm=
            "http://www.w3.org/2000/09/xmldsig#envelopedsignature"/>
        </ds:Transforms>
        <ds:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
                    TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
      </ds:Reference>
        </ds:SignedInfo>
     <ds:SignatureValue>
      x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHa
           EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen
           w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
     </ds:SignatureValue>
     <ds:KeyInfo>
       <ds:X509Data>
           <!-- The STS certificate -->
      <ds:X509Certificate>
      MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
      MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
      F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2l...
           </ds:X509Certificate>
       </ds:X509Data>
     </ds:KeyInfo>
  </ds:Signature>

  <saml:Subject>
```

```
        <!-- Here comes a NameID indicating the participant identifier of the
sender -->
        <saml:NameID
Format="http://busdox.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifie
r/1.0/">
           0010:5798000000001
        </saml:NameID>

        <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-
of-key">

<!-- Here comes a NameID indicating the ID of the sending Access Point who must
confirm with a key -->
        <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">
           http://SomeAcessPoint.busdox.org
        </saml:NameID>

       <!-- Here comes info on the key to confirm with (AP certificate> -->
        <saml:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType"
NotOnOrAfter="2009-12-31T12:00:00">
             <ds:KeyInfo>
                  <ds:X509Data>
                    <ds:X509Certificate>
                      <!-- Here comes the AP's X509 cert -->
                      MIICyjCCAjOgAwIBA.
                      </ds:X509Certificate>
                  </ds:X509Data>
              </ds:KeyInfo>
        </saml:SubjectConfirmationData>

     </saml:SubjectConfirmation>
  </saml:Subject>

  <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
      <saml:AuthnContext>
            <saml:AuthnContextClassRef>
               urn:oasis:names:tc:SAML:2.0:ac:classes:X509
            </saml:AuthnContextClassRef>
      </saml:AuthnContext>
  </saml:AuthnStatement>

  <saml:AttributeStatement>
    <!-- Assurance Level Attribute -->
    <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
        Name="urn:eu:busdox:attribute:assurance-level">
        <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```