



Secure Trusted Asynchronous Reliable Transport (START)

Version 1.0.0



WP8 2009-12-22



Contents

| | | |
|-------|---|----|
| 1.1 | Document history | 4 |
| 1.2 | Editor..... | 4 |
| 1.3 | Contributors (alphabetically) | 4 |
| 2 | Introduction | 5 |
| 2.1 | Goals and non-goals..... | 5 |
| 2.2 | Terminology | 6 |
| 2.2.1 | Notational conventions..... | 6 |
| 2.2.2 | Normative references | 6 |
| 2.2.3 | Non-normative references..... | 7 |
| 2.3 | Namespaces | 7 |
| 3 | Overview | 8 |
| 4 | Specification Profile Details | 10 |
| 4.1 | Use of WS-Transfer | 10 |
| 4.2 | BUSDOX defined headers | 10 |
| 4.3 | Message Exchange | 10 |
| 4.3.1 | Prerequisites for communication | 10 |
| 4.3.2 | Destination Message Channel | 10 |
| 4.3.3 | Delivery of BUSDOX messages..... | 10 |
| 4.3.4 | Faults..... | 11 |
| 4.4 | SOAP 1.1..... | 12 |
| 4.5 | Use of MTOM..... | 13 |
| 4.6 | Use of HTTP | 13 |
| 4.7 | WS-Addressing 1.0..... | 13 |
| 4.8 | WS-Transfer | 13 |
| 4.9 | Security | 13 |
| 4.9.1 | The <wsse:Security> Header Block | 13 |
| 4.9.2 | Message Authentication and Integrity | 14 |
| 4.9.3 | Including a SAML assertion in the Security header | 14 |
| 4.9.4 | Responses | 15 |

| | | |
|----------------------|---|----|
| 4.9.5 | Validation | 15 |
| 4.9.6 | Use of HTTPS | 16 |
| 4.9.7 | Signature Confirmation | 16 |
| 4.10 | WS-ReliableMessaging 1.1 | 16 |
| 4.10.1 | Successful initiation of the Reliable messaging sequences. | 16 |
| 4.10.2 | Delivery Assurances | 17 |
| 4.10.3 | Reliable exchange behaviour | 17 |
| 4.10.4 | Ensuring security of the WS-ReliableMessaging sequence | 18 |
| 5 | Infrastructure Messages | 18 |
| 5.1 | Ping Message | 18 |
| 5.1.1 | Ping Message Header Values..... | 19 |
| 5.1.2 | Business Message | 19 |
| 5.1.3 | Receiving AP Response | 20 |
| 5.1.4 | Authentication | 20 |
| 6 | SAML 2.0 assertion profile | 20 |
| 6.1 | Assumptions..... | 20 |
| 6.2 | SAML Assertion Profile..... | 21 |
| 6.2.1 | Authentication Assertion Profile..... | 21 |
| 6.2.2 | SAML Attribute Encoding..... | 22 |
| 7 | Appendix A..... | 23 |
| 7.1 | XML Schema for message headers | 23 |
| 7.2 | XML Schema for “Ping” message..... | 23 |
| 8 | Appendix B: Non-normative SAML token example | 23 |
| 8.1 | “Sender-vouches” subject confirmation..... | 23 |
| 8.2 | “Holder-of-key” Subject Confirmation..... | 25 |
| Document information | | |

1.1 Document history

| Date | Version | Initials | Changes |
|------------|---------|----------|---|
| 2009-01-03 | 0.1.0 | GS | Created template |
| 2009-02-09 | 0.2 | PZF | First steps |
| 2009-02-17 | 0.3 | PZF | Updated based on feedback from TG and GS |
| 2009-02-17 | 0.5 | GS | Merged SAML profile and Secure Trusted Asynchronous Reliable Transport, minor editorial updates |
| 2009-03-29 | 0.6 | PZF | Updates based on 0.5 Feedback |
| 2009-04-01 | 0.7 | GS | Minor editorial updates |
| 2009-04-28 | 0.8r1 | PZF | Updates based on 0.7 feedback discussions and spreadsheet |
| 2009-08-30 | 0.9 | PZF | Updates based on the Copenhagen F2F and feedback spreadsheet |
| 2009-10-22 | 0.95 | GS | Minor editorial updates |
| 2009-11-17 | 1.0 | PZF | Final updates for 1.0 |
| 2009-11-27 | 1.0 | GS | Minor updates, removed security policy section |
| 2009-12-22 | 1.0 | PZF | Updates based on Philip |

1.2 Editor

Paul Fremantle, WSO2

1.3 Contributors (alphabetically)

Jens Jakob Andersen, NITA

Mikkel Hippe Brun, NITA

Mike Edwards, IBM

Paul Fremantle, WSO2

Thomas Gundel, IT Crew

Philip Helger, Bundesrechenzentrum

Hans Guldager Knudsen, Lenio

Maria Raffaella Migliorini, CONSIP

Bergþór Skúlason, NITA

Gert Sylvest, Avanade

1 **2 Introduction**

2 This document describes the SOAP-based profile that is used by Business Document Exchange Network
3 (BUSDOX) Access Points to communicate and the SAML 2.0 assertions that are used in that communication.
4 This profile offers secure and reliable delivery of messages between Access Points. This is currently the only
5 required transport profile in BUSDOX.

6 BUSDOX Access Points communicate in a peer-to-peer model across the internet to form the BUSDOX
7 infrastructure. Each Access Point derives the endpoint addresses of other BUSDOX Access Points through
8 the BUSDOX Service Metadata Publishing Infrastructure. BUSDOX Access Points may communicate via
9 optional BUSDOX Transport Profiles, but they must always offer a START (Secure Trusted Asynchronous
10 Reliable Transport) endpoint by which any other Access Point may communicate.

11 In order to instantiate a working network, certain profile information is expected. For example, an instance
12 of BUSDOX is the PEPPOL infrastructure, which includes governance models, certificate rules, identifier
13 formats, and other profiling. This specification therefore excludes such profiling information.

14 The Secure Trusted Asynchronous Reliable Transport (START) defines a secure, reliable profile using a set of
15 well-known standards:

- 16 • SOAP 1.1
- 17 • WS-Addressing 1.0
- 18 • WS-Security 1.1
- 19 • WS-Transfer as a standard approach to accessing the message channels
- 20 • WS-ReliableMessaging 1.1
- 21 • SAML 2.0

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

25 **2.1 Goals and non-goals**

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

- 28 • Be a single profile that implementers can build and therefore gain access to BUSDOX with no
29 further requirements to implement other transport profiles.
- 30 • Clearly state the transport level requirements in a single document.
- 31 • Define a simple, interoperable communications pattern that APs can use to communicate.
- 32 • Define the message exchange formats and patterns clearly
- 33 • Ensure that messages are reliably delivered between APs, including providing the prerequisites for
34 logging and proof-of-delivery for messages at the transport level
- 35 • Ensure confidentiality of messages using transport-level encryption
- 36 • Ensure integrity and authenticity of received messages by signature validation
- 37 • Support transfer of messages that are opaque to the Access Point

- 38 • Define a set of BUSDOX specific headers within the message envelope so Access Points can
39 forward/transfer messages without requiring access to the business message.
- 40 • Establish a common format for representing authentication events in BUSDOX infrastructure in the
41 form of SAML 2.0 assertions / tokens.
- 42 • Recipients can assume that senders have been authenticated by their token Issuers and obtain
43 further proof via cryptographically signed tokens.

44 The Profile does NOT address:

- 45 • The verification of certificates, format of participant identifiers, and other details required to create
46 a full instantiation of BUSDOX.
- 47 • Communication with BUSDOX Service Metadata services or Security Token Services.
- 48 • Use of SAML 2.0 tokens for other purposes is not in scope for this note; these may include using
49 tokens to express sender-side certificate validation results or tokens issued by BUSDOX Security
50 Token Services for authenticating the Token Issuers themselves.
- 51 • The authentication process itself is not in scope for this profile; senders may use whatever
52 credentials accepted by the Token Issuers.

53 **2.2 Terminology**

54 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
55 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC
56 2119.

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

58 **2.2.1 Notational conventions**

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

60 **2.2.2 Normative references**

61 [BDEN-CDEF] Business Document Exchange Network - Common Definitions, CommonDefinitions.pdf

62 [WSS-1.1] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", [http://www.oasis-](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
63 [open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)

64 [WSS-STP] "Web Services Security: SAML Token Profile 1.1" [http://docs.oasis-open.org/wss/v1.1/wss-](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-errata-os-SAMLSecurityTokenProfile.pdf)
65 [v1.1-errata-os-SAMLSecurityTokenProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-errata-os-SAMLSecurityTokenProfile.pdf)

66 [WS-T] "Web Services Transfer (WS-Transfer)", W3C Working Draft 24th September 2009,
67 <http://www.w3.org/TR/2009/WD-ws-transfer-20090924/>

68 [WSRM-1.1] "Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1", [http://docs.oasis-](http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.html)
69 [open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.html](http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.html)

70 [WSA-1.0] "Web Services Addressing 1.0 - Core" ([http://www.w3.org/TR/2005/CR-ws-addr-core-](http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/)
71 [20050817/](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/>

72 [SOAP-1.1] "SOAP Version 1.1 ", <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

73 [RFC-2119] "Key words for use in RFCs to Indicate Requirement Levels",
74 <http://www.ietf.org/rfc/rfc2119.txt>

75 [SAML-2.0 assertion] "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)
76 V2.0", <http://docs.oasis-open.org/security/saml/v2.0/>

77 [MTOM] "SOAP Message Transmission Optimization Mechanism", <http://www.w3.org/TR/soap12-mtom/>

78 [XML-DSIG] XML Signature Syntax and Processing (Second Edition), <http://www.w3.org/TR/xmlsig-core/>

79 2.2.3 Non-normative references

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

82 2.3 Namespaces

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

| Namespace Prefix | Namespace |
|------------------|---|
| 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 |
| wsm | http://docs.oasis-open.org/ws-rx/wsm/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/xmlsig# |

85

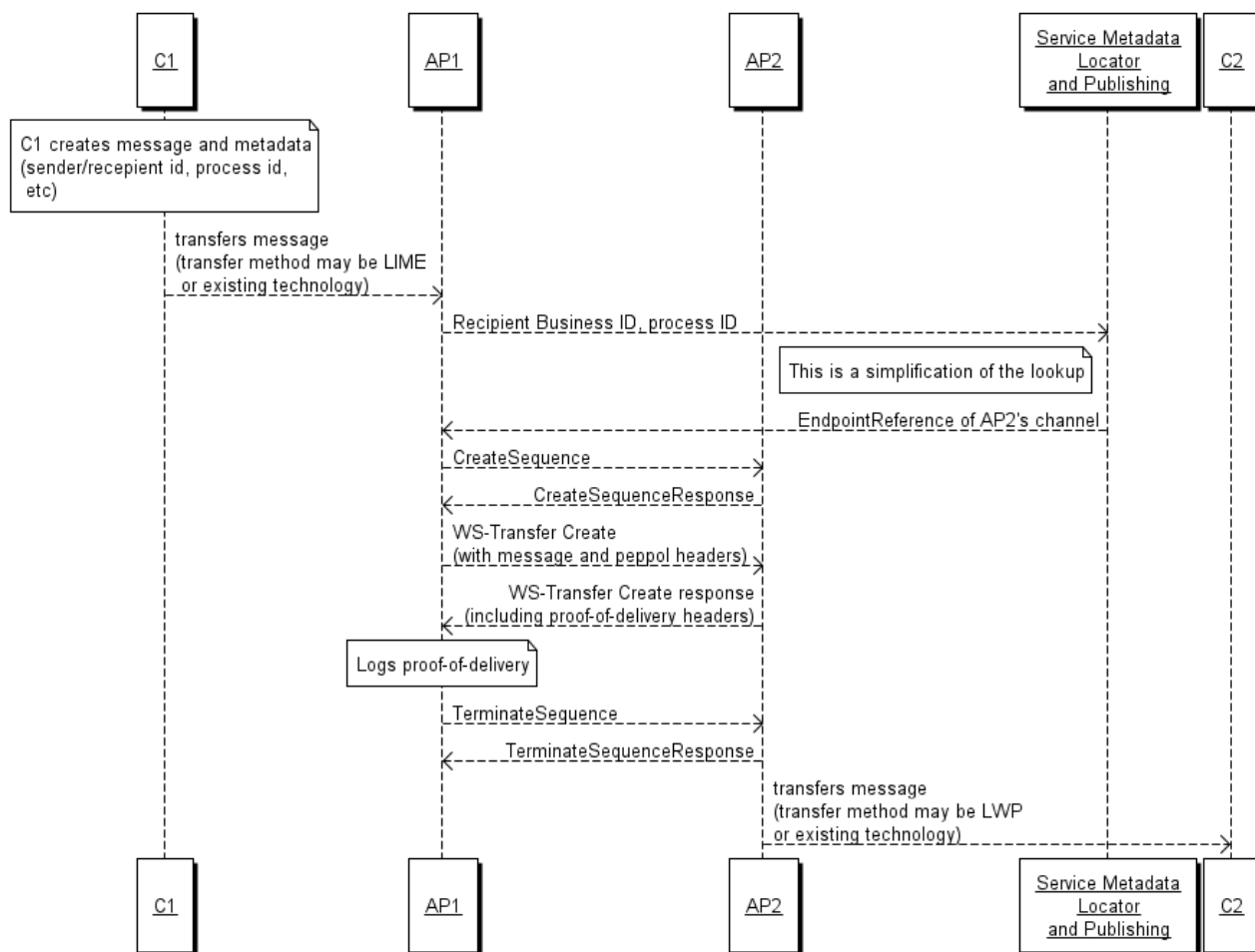
86

87 **3 Overview**

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

92 A typical flow might be:

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



109

110

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

114

115 **4 Specification Profile Details**

116 The following requirements apply to the BUSDOX START Profile.

117 **4.1 Use of WS-Transfer**

118 The WS-Transfer¹ [WS-T] specification is used here to support sending messages. This is done for two
119 reasons: WS-Transfer matches the semantics of a service that is able to handle XML messages of any type,
120 and also to increase the commonality between this profile and the BUSDOX Lightweight Message Exchange
121 Profile (LIME).

122 **4.2 BUSDOX defined headers**

123 Please see [BDEN-CDEF].

124 **4.3 Message Exchange**

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

127 **4.3.1 Prerequisites for communication**

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

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

131 **4.3.2 Destination Message Channel**

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

136 **4.3.3 Delivery of BUSDOX messages**

137 The SrcAP MUST use WSRM 1.1 to ensure reliable delivery of messages.

138 The SrcAP MUST send the message to be transmitted as the body of the WS-Transfer CREATE operation.

¹ 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.

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

140 • ids:RecipientIdentifier

141 • ids:ChannelIdentifier

142 • ids:SenderIdIdentifier

143 • ids:DocumentIdentifier

144 • ids:ProcessIdentifier

145 • ids:MessageIdentifier

146 Other headers MAY be included.

147 **4.3.4 Faults**

148 The DestAP can return faults in five circumstances:

149 • Firstly, it may have a “full channel”. This indicates that the SrcAP should retry at a later time. This is
150 enabled to allow the server to deal with the case where an onward system is not receiving
151 messages.

152 • Secondly, the endpoint may not be recognized. This may happen in cases where the SrcAP has
153 cached the endpoint, or there is incorrect/out-of-date information available in the metadata. In this
154 case, the SrcAP should refresh the metadata by re-requesting it from the Service Metadata Lookup
155 and Publishing systems.

156 • Thirdly, in the case where there is a security processing error (SAML assertion or signature not
157 validated).

158 • Fourthly in the case where the document type is not accepted for this recipient.

159 • Finally, there are other server errors that may fall outside those identified above.

160 The faults used are as follows:

161 **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 |

162

163 **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 |

164 **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 |

165 **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 |

166 **Server Error**

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

167 **4.4 SOAP 1.1**

168 APs MUST use SOAP 1.1 for all message exchange.

169 Messages MUST use the document/literal style.

170 **4.5 Use of MTOM**

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

175 **4.6 Use of HTTP**

176 Please see Common Definitions section 3.8

177 **4.7 WS-Addressing 1.0**

178 Please see Common Definitions section 3.10

179 **4.8 WS-Transfer**

180 The AP Message Channel interface is based on WS-Transfer [WS-T]. The services offered by the AP MUST be
181 a SOAP/HTTP binding using document/literal style of the interfaces defined in the WS-Transfer WSDL. The
182 current specification is based on a working draft of the WS-Transfer specification which is WS-I Basic Profile
183 Compliant².

184 The **[Body]**/wst:Create@Dialect attribute MUST NOT be used in the START profile.

185 The BUSDOX message resource that is created on the receiving AP MUST be the same representation as the
186 BUSDOX message sent by the sending AP. The WS-Transfer specification indicates that in this situation the
187 WS-Transfer CreateResponse body SHOULD be empty other than the ResourceCreated element. This
188 specification strengthens this: the CreateResponse body MUST only contain the ResourceCreated element
189 as specified by the WS-T specification. Extension elements MUST NOT be present.

190 The Destination Message Channel MUST offer the WS-Transfer CREATE operation to all other Access Points.
191 Other operations are not expected to be exposed as part of the START profile. In certain circumstances the
192 WS-Transfer PUT, DELETE and GET operations MAY be exposed at a BUSDOX endpoint, but their usage is
193 not covered by this specification and MUST NOT be required. These operations SHOULD be protected from
194 general access.

195 **4.9 Security**

196 The same security profile is to be used for both the WS-RM 1.1 protocol messages as well as the WS-
197 Transfer 'business' messages.

198 All SOAP request and response messages exchanged via this profile (including messages that do not carry
199 business document payloads) MUST be secured using the mechanisms described below.

200 **4.9.1 The <wsse:Security> Header Block**

² 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.

201 This section defines elements and processing rules for SOAP message security by profiling the
202 <wsse:Security> header block defined in [WSS-1.1]. Processing rules defined in [WSS-1.1] and [WSS-STP]
203 MUST be followed unless stated explicitly otherwise below.

204 A single <wsse:Security> header block MUST be present and MUST have a mustUnderstand attribute with
205 the logical value of true. Further, it MUST include a <wsu:Timestamp> with a <wsu:Created> element.

206 The value of the <wsu:Created> element SHOULD be within an appropriate offset from local time. In
207 absence of other guidance, a value of 5 minutes MAY be used.

208 If the <wsu:Timestamp> element includes a <wsu:Expires> element, the receiver MUST ensure that his local
209 time is before that time.

210 To prevent message replay, receivers SHOULD maintain a message cache, and check received
211 MessageIdentifier values against the cache. The cache timeouts are out of scope for this specification and
212 are to be defined by local deployment or other governance models.

213 **4.9.2 Message Authentication and Integrity**

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

- 217 • The SOAP <Body> element
- 218 • All security tokens embedded directly under the <wsse:Security> element (see next section for
219 special rules regarding SAML assertions).
- 220 • All SOAP header blocks in the message defined in this profile, including any all BUSDOX-
221 namespaced headers, all WS-Addressing and any WS-ReliableMessaging headers.

222 The signature MAY reference other elements including header blocks not mentioned in this profile.

223 The certificate MUST be included in a <wsse:BinarySecurityToken> element in the security header. In the
224 message signature, the <ds:KeyInfo> element MUST refer to this token via a
225 <wsse:SecurityTokenReference>.

226 **4.9.3 Including a SAML assertion in the Security header**

227 The sender Access Point MUST include a SAML 2.0 assertion in the security header which contains the
228 identity of the sender and details of the sender authentication (see SAML profile below).

229 Authentication assertions MUST be signed by the sender by including first a
230 <wsse:SecurityTokenReference> in <wsse:Security> header block, and then referencing this element from
231 the message signature using a <ds:Reference> element. The security token reference MUST include a
232 <wsse:KeyIdentifier> with a ValueType of “http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
233 1.1#SAMLID” and specify the ID of the SAML assertion. The <ds:Reference> element MUST use a transform
234 algorithm set to “http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsssoap-message-security-
235 1.0#STR-Transform”.

236 **4.9.3.1 Types of SAML assertions**

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

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

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

254 **4.9.4 Responses**

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

258 **4.9.5 Validation**

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

263 The SrcAP SHOULD validate that the Subject Unique Identifier of the certificate used to sign the response
264 messages matches the Subject Unique Identifier published in the Service Metadata Publishing.

265 When validating a signed response message, the sender Access Point SHOULD check that the certificate in
266 the response matches the metadata received from the Service Metadata Publisher. This is done by
267 comparing the subject common name³ in the certificate to the value stated in the metadata. This check
268 ensures that only the legitimate Access Point stated in the service metadata will be able to produce correct
269 responses.

³ 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.

270 **4.9.6 Use of HTTPS**

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

273 **4.9.7 Signature Confirmation**

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

277 **4.9.7.1 Additional Processing Rules for holder-of-key Assertions**

278 When the authentication assertion has a subject confirmation method being "holder-of-key" it means that
279 the sending AP must prove possession of the key mentioned in the assertion's <SubjectConfirmationData>
280 in order for the recipient AP to rely on the assertion. The proof-of-possession of the key will be achieved via
281 the message signature and provides additional assurance that the sender AP is allowed to use the assertion
282 in a web service invocation.

283 In this profile, a holder-of-key Assertion in the <SubjectConfirmationData> element MUST include a key
284 that can be used to verify the message signature. Thus, the same key used for message authentication and
285 integrity is used to confirm the right to use the assertion for message authorization purposes.

286 The message signature (i.e. the <ds:Signature> element) MUST refer to the token with the subject
287 confirmation key within the <ds:KeyInfo> element.

288 The receiver MUST check that the message is signed by the same key mentioned in the assertion's subject
289 confirmation element before relying on the assertion content.

290 **4.10 WS-ReliableMessaging 1.1**

291 In this profile, both the SrcAP and the DestAP act as both RM Sources (RMS) and RM Destinations (RMD).
292 Web Services Reliable Messaging (WSRM) is used to ensure the delivery of both the requests (business
293 messages) as well as the responses.

294 A single WSRM sequence MAY be used to send multiple messages where the Sender Identifiers and/or
295 Recipient Identifiers are different. This supports the case where an AP may have a number of messages all
296 destined to be handled by a remote AP, but for different recipients. In this case the AP may re-use the
297 WSRM sequence. In addition, the WSRM sequence MAY be held open in the case that other messages may
298 require delivery before either end times out the sequence. The normal WSRM sequence timeout model
299 applies as long as the further restrictions outlined below are also conformed to.

300 **4.10.1 Successful initiation of the Reliable messaging sequences.**

301 The SrcAP MUST successfully create a Sequence as the first communication with the DestAP.

302 The same endpoint reference for the DestMC MUST be used for the wsrn:CreateSequence message.

303 The wsrn:CreateSequence message MUST include an Offer Sequence, and the DestAP MUST accept the
304 Offered Sequence.

305 The DestAP MUST send all responses via the Offered sequence.

306 **4.10.2 Delivery Assurances**

307 The DestAP MUST implement the <wsrmp:ExactlyOnce> delivery assurance on the created sequence.

308 The SrcAP MUST implement the <wsrmp:ExactlyOnce/> delivery assurance on the offered sequence.

309 The DestAP SHOULD implement the <wsrmp:InOrder/> delivery assurance on the created sequence.

310 **4.10.3 Reliable exchange behaviour**

311 The following requirements ensure that the reliable messaging framework effectively delivers messages
312 from SrcAP to DestAP, or leaves the Access Points with a clear status of the transmitted messages.

313 Both RMSs MUST continue to resend unacknowledged messages until the WSRM sequence is closed or
314 terminated.

315 Both RMDs MUST accept unacknowledged messages until the WSRM sequence is closed or terminated.

316 When a message is retransmitted, the wsa:MessageID MUST be the same as the original transmission of
317 the message.

318 The SrcAP MAY send a single message per sequence, or it MAY send multiple messages per sequence.
319 However, it is RECOMMENDED that sequences are timed out when inactive to prevent resource utilization
320 on a remote system.

321 If the SrcAP is ending a sequence, and it has received and logged a complete acknowledgement for the
322 sequence, it MUST send a TerminateSequence with the LastMsgNumber. It MUST repeat this until it
323 receives a response indicating that the Sequence is terminated.

324 The DestAP MUST send a final acknowledgement back in response to this TerminateSequence.

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

328 The SrcAP MUST send either a CloseSequence or TerminateSequence message for every sequence.

329 If the DestAP decides to time out a sequence it MUST close the sequence to allow the SrcAP the
330 opportunity to access the final acknowledgement.

331 To ensure maximum interoperability, the SrcAP's RMS MUST NOT use wsa:ReferenceParameters in the
332 AcksTo endpoint reference.

333 The SrcAP's RMS MUST support piggybacked acknowledgements.

334 The SrcAP MUST sign the body of the WS-Transfer Create message together with the WSRM Sequence
335 header, and the corresponding signature MUST be logged in persistent storage. The DestAP MUST sign the
336 WSRM acknowledgements in conjunction with the WS-Addressing RelatesTo header, and the SrcAP
337 SHOULD keep a persistent log of the latest acknowledgement and signature. Security tokens are only
338 required in SOAP messages that actually carry business document payload.

339

340 **4.10.4 Ensuring security of the WS-ReliableMessaging sequence**

341 There are a number of potential attacks against the WSRM 1.1 sequence. The following categorizes them
342 and the protections enforced:

343 ***Denial of Service by Creating multiple sequences***

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

346 ***Sequence Attack***

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

353 **5 Infrastructure Messages**

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

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

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

364 **5.1 Ping Message**

365 The BUSDOX START Ping Message is targeted at a receiving Access Point using the transport infrastructure
366 rather than to an end participant Id. Therefore the message cannot have real participant identifiers. Instead
367 the BUSDOX profile defines two well-known participant identifiers that are to be used with infrastructure
368 messages.

369 The normal behaviour that is expected is that an Access Point would aim to test communications for a given
370 participant's Access Point. The Sender AP should look up any valid SMP entry of the Participant, and then
371 send the Ping Message to the Endpoint Reference listed for that entry. Although the endpoint is located by
372 searching for a given document exchange type, the receiving AP SHOULD respond to the Ping irrespectively.
373 Therefore BUSDOX endpoints SHOULD support at least two types of business message - the real business
374 message specified in the SMP and the Ping message.

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

377 **5.1.1 Ping Message Header Values**

378 The message headers are below.

- 379 • ids:RecipientIdentifier

380 This must have a fixed value of:

```
381 <smp:ParticipantIdentifier scheme="busdox-actorid-transport">  
382 busdox:recipient  
383 </smp:ParticipantIdentifier>
```

384

- 385 • ids:ChannelIdentifier

386 This value is found through the normal lookup mechanisms of SML and SMP

- 387 • ids:SenderIdentifier

388 This must have a fixed value of:

```
389 <smp:ParticipantIdentifier scheme="busdox-actorid-transport">  
390 busdox:sender  
391 </smp:ParticipantIdentifier>
```

392

- 393 • ids:DocumentIdentifier

```
394 <DocumentIdentifier scheme="busdox-docid-qns">  
395 busdox:ping  
396 </DocumentIdentifier>
```

397

- 398 • ids:ProcessIdentifier

```
399 <ProcessIdentifier scheme="busdox-procid-transport">  
400 Busdox:noprocess  
401 </DocumentIdentifier>
```

402

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

404 **5.1.2 Business Message**

405 The Ping business message contents of the WS-T Create Body MUST be:

```
406 <start:Ping/>
```

407 **5.1.3 Receiving AP Response**

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

410 **5.1.4 Authentication**

411 The aim of the Ping message is to test as much of the BUSDOX infrastructure as possible. Therefore,
412 although there is no business need for a SAML token to be present, the Ping message SHOULD have a token
413 present. The SAML token, if included in a Ping message, MUST have the Subject identity be the same as the
414 Issuer, that is, the identity of the Access Point. If the Ping message has no attached SAML Token, the
415 receiving AP MAY discard it and fault, or may respond with an empty CreateResponse.

416 **6 SAML 2.0 assertion profile**

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

422 **6.1 Assumptions**

- 423 • Authentication Token Issuers are able to authenticate senders (whom they receive messages from).
- 424 • Mechanisms have been established that allows recipients to trust Token Issuers; specifically, they
425 should be able to validate the Token Issuer's signatures on tokens, and trust the claims provided in
426 the tokens to be correct.
- 427 • Common definitions of authentication levels are established in BUSDOX; this caters for different
428 authentication mechanisms between Token Issuers and senders. By defining common
429 authentication levels (e.g. 1-4) and criteria for mapping existing authentication methods to these
430 levels, the recipient can make an informed decision regarding the risk of accepting the message
431 without knowing the details of the sender-Token Issuer relation.
- 432 • Common syntax and semantics for identifiers have been established. As identifiers will vary with
433 application domains, they are left to deployment profiles to specify.
- 434 • A Token Issuer is able to map the identity of sender (as established via the private authentication)
435 to the sender's identifier.
- 436 • Token Issuers are capable of issuing (signing) SAML tokens upon authentication of senders; the
437 tokens are probably not requested and returned via SAML protocols or WS-Trust if it is performed
438 as an internal operation in an Access Point. However, when an Access Point delegates token
439 issuance to an external STS, WS-Trust may be used.
- 440 • Servers in the BUSDOX infrastructure have reasonably synchronized clocks; some security checks
441 rely on timestamps and this is problematic if clocks drift too much. A time synchronization policy
442 will be defined (e.g. stratum 2⁴ is required).

⁴ 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

443 **6.2 SAML Assertion Profile**

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

- 445 • The subject (sender) identity identifier
- 446 • Identity and signature of the token issuer
- 447 • Time of authentication
- 448 • Strength of authentication method
- 449 • Life time of token
- 450 • Audience of the token (where can the token be used)
- 451 • Subject confirmation (how can a presenter of a token demonstrate that he is authorized to use the
- 452 token)

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

454 **6.2.1 Authentication Assertion Profile**

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

- 459 • The Assertion MUST be a SAML 2.0 Assertion
- 460 • The <Assertion> element SHOULD have an id attribute containing a cryptographically random value
461 between 128 and 160 bytes in length.
- 462 • The Issuer element MUST be present and contain appropriate BUSDOX identifier of the issuer.
- 463 • The NameID format MUST be urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified.
- 464 • The Subject element MUST be present and contain appropriate BUSDOX identifier of the sender.
- 465 • The Format attribute MUST be set to the ParticipantIdentifierType URI and the value to the
466 ParticipantIdentifierValue defined in Service Metadata Publishing document.
- 467 • The assertion MUST be digitally signed by the issuer; the signature MUST include the issuer's
468 certificate as a PEM base 64 encoded X509 DER value in the <ds:KeyInfo> element.
- 469 • Assertions MUST NOT be encrypted at the SAML level; this will be handled by the transport
470 protocols.
- 471 • An Assertion MUST NOT contain an <AuthzDecisionStatement>.
- 472 • An Assertion MUST contain exactly one <AuthnStatement> and one <AttributeStatement>.
- 473 • The <AuthnStatement> element MUST have an "AuthInstant" attribute specifying the time
474 authentication occurred.
- 475 • The subject element MUST contain at least one <SubjectConfirmation> sub-element containing a
476 Method of "urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" or
477 "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches". In case of "holder-of-key", the element MUST
478 again have two sub-elements:
 - 479 ○ A NameID with format attribute set to "urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
480 and value indicating the ID of the Token Issuer. This indicates that the assertion is used by
481 the Access Point (included in SOAP message) on behalf of the sender.

- 482 o A <SubjectConfirmationData> element with xsi:type saml2:KeyInfoConfirmationDataType
483 and a sub-element referring to the issuer's certificate. The <SubjectConfirmationData>
484 element MUST have an "NotOnOrAfter" attribute defining when the assertion expires. A
485 recipient MUST reject the assertion after this time.
486 • Advice elements MAY safely be ignored by implementations.

487 **6.2.2 SAML Attribute Encoding**

- 488 • All attribute names defined in this profile MUST be prefixed with "urn:eu:busdox:attribute";
489 recipients are not required to process attributes not defined in this profile.
- 490 • The <AttributeStatement> MUST contain an attribute stating the level of authentication. The
491 attribute name must be "urn:eu:busdox:attribute:assurance-level" and the value an integer in the
492 range 1-4. BUSDOX will use the assurance levels defined by NIST in the Electronic Authentication
493 Guide, publication 800-63 ([http://csrc.nist.gov/publications/nistpubs/800-63/SP800-](http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf)
494 [63V1_0_2.pdf](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.
- 495 • The <NameFormat> XML attribute on <Attribute> elements MUST be:
496 urn:oasis:names:tc:SAML:2.0:attrname-format:basic
- 497 • The <FriendlyName> XML attribute MAY be used but implementations SHOULD NOT rely on it.
- 498 • All attribute values MUST be simple text strings with type "xs:string".
- 499 • Encrypted attributes MUST NOT be used.

500

501

502 7 Appendix A

503 7.1 XML Schema for message headers

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

505 7.2 XML Schema for “Ping” message

```
506 <?xml version="1.0" encoding="UTF-8"?>
507 <schema
508   targetNamespace="http://busdox.org/transport/start/1.0/"
509   elementFormDefault="qualified"
510   xmlns="http://www.w3.org/2001/XMLSchema"
511   xmlns:tns="http://busdox.org/transport/start/1.0/"
512   version="1.0.0">
513
514   <element name="Ping" type="tns:PingType" />
515
516   <!-- Length is 0 - only use PING as a marker -->
517   <simpleType name="PingType">
518     <restriction base="string">
519       <length value="0" />
520     </restriction>
521   </simpleType>
522 </schema>
523
524
```

525 8 Appendix B: Non-normative SAML token example

526 8.1 “Sender-vouches” subject confirmation

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

```
532
533 <saml:Assertion ID="a12312312312312372975934659137459
534 871324587613845613984756981346598314634513451345"
535   IssueInstant="2001-12-31T12:00:00"
536   Version="2.0"
537   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
538   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
539   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
540   xmlns:xs="http://www.w3.org/2001/XMLSchema">
541
542   <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
543     format:unspecified">
544     http://SomeAccessPoint.busdox.org</saml:Issuer>
545   <ds:Signature>
546     <ds:SignedInfo>
```

```

547     <ds:CanonicalizationMethod
548         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
549     <ds:SignatureMethod
550         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
551     <ds:Reference URI="#idvalue31231231231312">
552         <ds:Transforms>
553             <ds:Transform Algorithm=
554                 "http://www.w3.org/2000/09/xmldsig#envelopedsignature" />
555             </ds:Transforms>
556             <ds:DigestMethod
557                 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
558             <ds:DigestValue>
559                 TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
560             </ds:Reference>
561         </ds:SignedInfo>
562     <ds:SignatureValue>
563         x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHa
564         EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaKlywS7gFgsD01qjyen
565         w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
566     </ds:SignatureValue>
567     <ds:KeyInfo>
568         <ds:X509Data>
569             <!-- The Access Point's certificate -->
570             <ds:X509Certificate>
571                 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwakxCzAJBgNVBAYTAlVT
572                 MRlWEAYDVQQIEWlXaXNjb25zaW4xEDA0BgNVBACTB01hZGlzb24xIDAeBgNVBAoT
573                 FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc21...
574             </ds:X509Certificate>
575         </ds:X509Data>
576     </ds:KeyInfo>
577 </ds:Signature>
578
579     <saml:Subject>
580         <!-- Here comes a NameID indicating the participant identifier of the
581 sender -->
582         <saml:NameID
583 Format="http://busdov.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifi
584 er/1.0/">
585             0010:5798000000001
586         </saml:NameID>
587
588         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:sender-
589 vouches" />
590     </saml:Subject>
591
592     <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
593         <saml:AuthnContext>
594             <saml:AuthnContextClassRef>
595                 urn:oasis:names:tc:SAML:2.0:ac:classes:X509
596             </saml:AuthnContextClassRef>
597         </saml:AuthnContext>
598     </saml:AuthnStatement>
599
600     <saml:AttributeStatement>
601         <!-- Assurance Level Attribute -->
602         <saml:Attribute
603             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
604             Name="urn:eu:busdov:attribute:assurance-level">
605             <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>

```

```

606     </saml:Attribute>
607 </saml:AttributeStatement>
608 </saml:Assertion>
609

```

610 8.2 “Holder-of-key” Subject Confirmation

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

```

614 <saml:Assertion ID="a12312312312312372975934659137459
615 871324587613845613984756981346598314634513451345"
616   IssueInstant="2001-12-31T12:00:00"
617   Version="2.0"
618   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
619   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
620   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
621   xmlns:xs="http://www.w3.org/2001/XMLSchema">
622
623   <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
624     format:unspecified">
625     http://SomeTokenService.busdox.org</saml:Issuer>
626   <ds:Signature>
627     <ds:SignedInfo>
628       <ds:CanonicalizationMethod
629         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
630       <ds:SignatureMethod
631         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
632       <ds:Reference URI="#idvalue31231231231312">
633         <ds:Transforms>
634           <ds:Transform Algorithm=
635             "http://www.w3.org/2000/09/xmldsig#envelopedsignature" />
636         </ds:Transforms>
637         <ds:DigestMethod
638           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
639         <ds:DigestValue>
640           TCDVsuG6grhyHbzHqFwFzGrxIPE=</ds:DigestValue>
641         </ds:Reference>
642       </ds:SignedInfo>
643     <ds:SignatureValue>
644       x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHa
645       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen
646       w6vKhaqlled10BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
647     </ds:SignatureValue>
648     <ds:KeyInfo>
649       <ds:X509Data>
650         <!-- The STS certificate -->
651         <ds:X509Certificate>
652           MIICyJCCAjoGAWIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
653           MRlWEAyDVoQIEWlXaXNjb25zaW4xZDA0BGNVBAcTB01hZG1zb24xIDAeBgNVBAoT
654           F1VuaXZlcnNpdHkqb2YgV21zY29uc2luMSswKQYDVQQLEyJEaXZpc21...
655         </ds:X509Certificate>
656       </ds:X509Data>
657     </ds:KeyInfo>
658   </ds:Signature>
659
660 <saml:Subject>

```

```

661         <!-- Here comes a NameID indicating the participant identifier of the
662 sender -->
663         <saml:NameID
664 Format="http://busdox.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifi
665 er/1.0/">
666             0010:5798000000001
667         </saml:NameID>
668
669         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-
670 of-key">
671
672 <!-- Here comes a NameID indicating the ID of the sending Access Point who must
673 confirm with a key -->
674         <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
675 format:unspecified">
676             http://SomeAccessPoint.busdox.org
677         </saml:NameID>
678
679         <!-- Here comes info on the key to confirm with (AP certificate) -->
680         <saml:SubjectConfirmationData
681 xsi:type="saml:KeyInfoConfirmationDataType" NotOnOrAfter="2009-12-31T12:00:00">
682             <ds:KeyInfo>
683                 <ds:X509Data>
684                     <ds:X509Certificate>
685                         <!-- Here comes the AP's X509 cert -->
686                         MIICyjCCAjOgAwIBA.
687                         </ds:X509Certificate>
688                     </ds:X509Data>
689                 </ds:KeyInfo>
690             </saml:SubjectConfirmationData>
691
692         </saml:SubjectConfirmation>
693     </saml:Subject>
694
695     <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
696         <saml:AuthnContext>
697             <saml:AuthnContextClassRef>
698                 urn:oasis:names:tc:SAML:2.0:ac:classes:X509
699             </saml:AuthnContextClassRef>
700         </saml:AuthnContext>
701     </saml:AuthnStatement>
702
703     <saml:AttributeStatement>
704         <!-- Assurance Level Attribute -->
705         <saml:Attribute
706             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
707             Name="urn:eu:busdox:attribute:assurance-level">
708             <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>
709         </saml:Attribute>
710     </saml:AttributeStatement>
711 </saml:Assertion>
712
713

```