



Secure Trusted Asynchronous Reliable Transport (START)

Version 0.9.5



WP8 2009-09-01



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	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	13
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 <wse: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.....	15
4.9.7	Signature Confirmation	15
4.10	WS-ReliableMessaging 1.1	16
4.10.1	Successful initiation of the Reliable messaging sequences.....	16
4.10.2	Delivery Assurances.....	16
4.10.3	Reliable exchange behaviour.....	16
4.10.4	Ensuring security of the WS-ReliableMessaging sequence	18
5	SAML 2.0 assertion profile.....	18
5.1	Assumptions	18
5.2	SAML Assertion Profile	19
5.3	Authentication Assertion Profile	19
5.3.1	Attribute Encoding.....	20
6	Appendix A.....	21
6.1	XML Schema for message headers.....	21
7	Appendix B: Non-normative SAML token example	21
7.1	“Sender-vouches” subject confirmation	21
7.2	“Holder-of-key” Subject Confirmation	22
8	Appendix C – WS-SecurityPolicy for START	24
	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

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

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

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

21 **2.1 Goals and non-goals**

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

- 24 • Be a single profile that implementers can build and therefore gain access to BUSDOX with no
25 further requirements to implement other transport profiles.
- 26 • Clearly state the transport level requirements in a single document.
- 27 • Define a simple, interoperable communications pattern that APs can use to communicate.
- 28 • Define the message exchange formats and patterns clearly
- 29 • Ensure that messages are reliably delivered between APs, including providing the prerequisites for
30 logging and proof-of-delivery for messages at the transport level
- 31 • Ensure confidentiality of messages using transport-level encryption
- 32 • Ensure integrity and authenticity of received messages by signature validation
- 33 • Support transfer of messages that are opaque to the Access Point
- 34 • Define a set of BUSDOX specific headers within the message envelope so Access Points can
35 forward/transfer messages without requiring access to the business message.
- 36 • Establish a common format for representing authentication events in BUSDOX infrastructure in the
37 form of SAML 2.0 assertions / tokens.

- 38 • Recipients can assume that senders have been authenticated by their token Issuers and obtain
39 further proof via cryptographically signed tokens.

40 The Profile does NOT address:

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

47 **2.2 Terminology**

48 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
49 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC
50 2119.

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

52 **2.2.1 Notational conventions**

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

54 **2.2.2 Normative references**

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

56 [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)
57 [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)

58 [WS-T] "Web Services Transfer (WS-Transfer)", W3C Editor's Draft 22 April 2009,
59 <http://www.w3.org/2002/ws/ra/edcopies/wst.html> (retrieved on 28/4/2009)

60 [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)
61 [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)

62 [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/)
63 [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/>

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

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

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

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

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

71 **2.2.3 Non-normative references**

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

74 **2.3 Namespaces**

75 The following table lists XML namespaces that are used in this specification. The choice of any namespace
76 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/xmldsig#

77

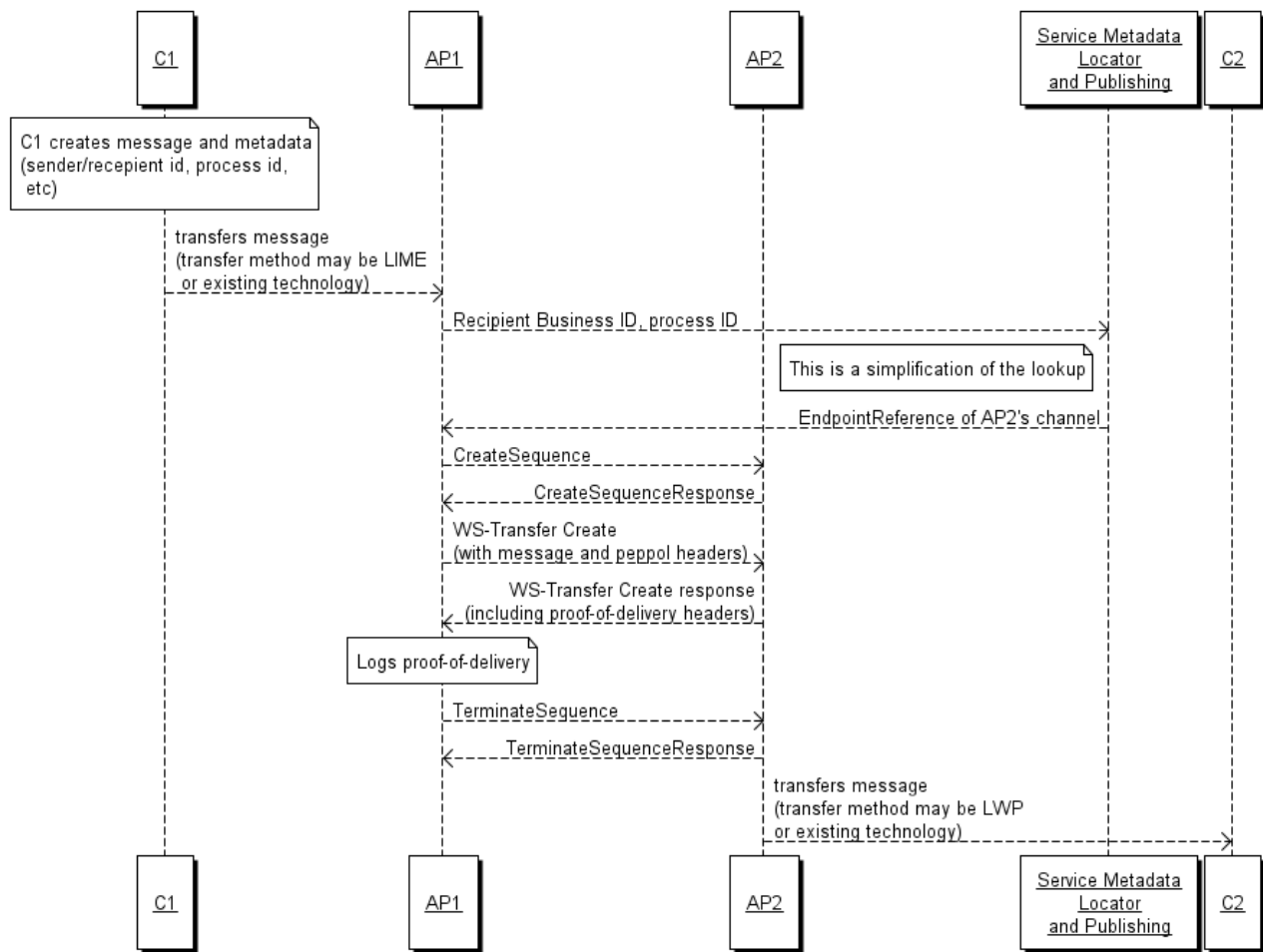
78

79 **3 Overview**

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

84 A typical flow might be:

- 85 • Message is created by Company C1, using either existing systems or a BUSDOX Lightweight Message
86 Exchange Profile client.
- 87 • The message is transferred to Access Point AP1, including information required such as:
 - 88 ○ Recipient Identifier and identifier type
 - 89 ○ Sender Identifier and identifier type
 - 90 ○ Document type identifier, identifier type, subtype discriminator and discriminator type
- 91 • AP1 uses the BUSDOX Metadata Lookup and publishing specifications and services to identify the
92 endpoints and keys for the Access Point handling the recipients messages (AP2).
- 93 • AP1 gets or creates any tokens it needs to include in the message transfer, i.e. X509 certificates and
94 SAML 2.0 token.
- 95 • AP1 adds the correct SOAP headers containing recipient, sender and document type information.
- 96 • AP1 signs the message.
- 97 • AP1 uses WS-ReliableMessaging and TSL to transfer the message to AP2 as part of a reliable exchange
- 98 • AP2 responds with a signed proof-of-delivery message to AP1
- 99 • Finally AP1 logs a signed proof-of-delivery of the message



100

101

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

105

106 **4 Profile Details**

107 The following requirements apply to the BUSDOX START Profile.

108 **4.1 Use of WS-Transfer**

109 The WS-Transfer [WS-T] specification is used here to support sending messages. This is done partly because
110 WS-Transfer matches the semantics of a service that is able to handle XML messages of any type, and partly
111 to increase the commonality between this profile and the BUSDOX Lightweight Message Exchange Profile
112 (LIME). The profile is currently based on an interim version of the WS-Transfer specification because it is
113 WS-I compliant. It is the intention of the editors to move to a stable version of the WS-Transfer
114 specification before this Profile is final.

115 **4.2 BUSDOX defined headers**

116 Please see [BDEN-CDEF].

117 **4.3 Message Exchange**

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

120 **4.3.1 Prerequisites for communication**

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

- 123 • The WS-Addressing EndpointReference for the *Destination Message Channel (DestMC)* of the
124 destination AP.
- 125 • The certificate used by the destination AP to sign response messages.

126 **4.3.2 Destination Message Channel**

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

131 **4.3.3 Delivery of BUSDOX messages**

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

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

134

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

136 • ids:RecipientIdentifier

137 • ids:ChannelIdentifier

138 • ids:SenderIdentifier

139 • ids:DocumentIdentifier

140 • ids:ProcessIdentifier

141 • ids:MessageIdentifier

142 Other headers MAY be included.

143 **4.3.4 Faults**

144 The DestAP can return faults in five circumstances:

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

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

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

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

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

156 The faults used are as follows:

157 **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

158

159

160 **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

161 **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

162 **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

163

164 **Server Error**

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

165 **4.4 SOAP 1.1**

166 APs MUST use SOAP 1.1 for all message exchange.

167 Messages MUST use the document/literal style.

168 **4.5 Use of MTOM**

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

173 **4.6 Use of HTTP**

174 Please see Common Definitions section 3.8

175 **4.7 WS-Addressing 1.0**

176 Please see Common Definitions section 3.10

177 **4.8 WS-Transfer**

178 The AP Message Channel interface is based on WS-Transfer [WS-T]. The services offered by the AP MUST be
179 a SOAP/HTTP binding using document/literal style of the interfaces defined in the WS-Transfer WSDL. (It is
180 the intention of the editors to move to a WS-I Basic Profile compliant version of WS-T before this
181 specification is final). Please note that WP8 specification group is tracking the progress of WS-T and is
182 hopeful that there will be a recommendation from the WS-T specification group in time for the 1.0 BUSDOX
183 specs.

184 The Destination Message Channel MUST offer the WS-Transfer CREATE operation to all other Access Points.
185 Other operations may be protected from general access.

186 **4.9 Security**

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

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

191 **4.9.1 The <wsse:Security> Header Block**

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

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

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

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

201 To prevent message replay, receivers SHOULD maintain a message cache, and check received
202 MessageIdentifier values against the cache. How long time a message should be kept in the cache at the
203 WSP is governed by deployment policy.

204 **4.9.2 Message Authentication and Integrity**

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

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

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

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

217 **4.9.3 Including a SAML assertion in the Security header**

218 The sender Access Point MUST include a SAML 2.0 assertion in the security header which contains the
219 identity of the sender and details of the sender authentication (see SAML profile below). Before including
220 the SAML assertion in the header it MUST be encrypted via XML encryption using a symmetric encryption
221 key conveyed under the recipient AP's digital certificate. After encryption a wsu:Id attribute (containing a
222 message-wide unique id) MUST be added to the resulting <xenc:EncryptedData> element, and this element
223 MUST be covered by the message signature (mentioned above) via a direct reference (i.e. not STR
224 Transform) from the signature to this Id (i.e. the message signature's <ds:SignedInfo> contains a
225 <ds:Reference> element which refers the wsu:Id of the <xenc:EncryptedData> element as well as the
226 resulting digest of the element) .

227 **4.9.3.1 Types of SAML assertions**

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

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

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

245 **4.9.4 Responses**

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

249 **4.9.5 Validation**

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

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

256 When validating a signed response message, the sender Access Point SHOULD check that the certificate in
257 the response matches the metadata received from the Service Metadata Publisher. This is done by
258 comparing the subject common name¹ in the certificate to the value stated in the metadata. This check
259 ensures that only the legitimate Access Point stated in the service metadata will be able to produce correct
260 responses.

261 **4.9.6 Use of HTTPS**

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

264 **4.9.7 Signature Confirmation**

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

268 **4.9.7.1 Additional Processing Rules for holder-of-key Assertions**

269 When the authentication assertion has a subject confirmation method being “holder-of-key” it means that
270 the sending AP must prove possession of the key mentioned in the assertion’s <SubjectConfirmationData>

¹ 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.

271 in order for the recipient AP to rely on the assertion. The proof-of-possession of the key will be achieved via
272 the message signature and provides additional assurance that the sender AP is allowed to use the assertion
273 in a web service invocation.

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

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

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

281 **4.10 WS-ReliableMessaging 1.1**

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

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

291 **4.10.1 Successful initiation of the Reliable messaging sequences.**

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

293 The same endpoint reference for the DestMC MUST be used for the wsrmp:CreateSequence message.

294 The wsrmp:CreateSequence message MUST include an Offer Sequence, and the DestAP MUST accept the
295 Offered Sequence.

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

297 **4.10.2 Delivery Assurances**

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

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

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

301 **4.10.3 Reliable exchange behaviour**

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

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

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

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

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

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

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

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

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

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

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

324 The SrcAP's RMS MUST support piggybacked acknowledgements.

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

330

331 **4.10.4 Ensuring security of the WS-ReliableMessaging sequence**

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

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

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

337 ***Sequence Attack***

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

344 **5 SAML 2.0 assertion profile**

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

350 **5.1 Assumptions**

- 351 • Authentication Token Issuers are able to authenticate senders (whom they receive messages from).
- 352 • Mechanisms have been established that allows recipients to trust Token Issuers; specifically, they
353 should be able to validate the Token Issuer’s signatures on tokens, and trust the claims provided in
354 the tokens to be correct.
- 355 • Common definitions of authentication levels are established in BUSDOX; this caters for different
356 authentication mechanisms between Token Issuers and senders. By defining common
357 authentication levels (e.g. 1-4) and criteria for mapping existing authentication methods to these
358 levels, the recipient can make an informed decision regarding the risk of accepting the message
359 without knowing the details of the sender-Token Issuer relation.
- 360 • Common syntax and semantics for identifiers in BUSDOX have been established.
- 361 • A Token Issuer is able to map the identity of sender (as established via the private authentication)
362 to the sender’s identifier in the BUSDOX infrastructure.
- 363 • Token Issuers are capable of issuing (signing) SAML tokens upon authentication of senders; the
364 tokens are probably not requested and returned via SAML protocols or WS-Trust if it is performed
365 as an internal operation in an Access Point. However, when an Access Point delegates token
366 issuance to an external STS, WS-Trust may be used.

- 367 • Servers in the BUSDOX infrastructure have reasonably synchronized clocks; some security checks
368 rely on timestamps and this is problematic if clocks drift too much. A time synchronization policy
369 will be defined (e.g. stratum 2² is required).

370 **5.2 SAML Assertion Profile**

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

- 372 • The subject (sender) identity (unique company identifier such as GLN numbers)
373 • Identity and signature of the token issuer
374 • Time of authentication
375 • Strength of authentication method
376 • Life time of token
377 • (Audience of the token (where can the token be used))
378 • Subject confirmation (how can a presenter of a token demonstrate that he is authorized to use the
379 token)

380 See section 0 for a non-normative SAML token sample.

381 **5.3 Authentication Assertion Profile**

382 Below is given a SAML 2.0 Assertion profile that aims to represent the above information. Flexibility and
383 choices in SAML have intentionally been limited in order to simplify implementation and increase chances
384 of interoperability:

- 385 • The Assertion MUST be a SAML 2.0 Assertion
386 • The <Assertion> element SHOULD have an id attribute containing a cryptographically random value
387 between 128 and 160 bytes in length.
388 • The Issuer element MUST be present and contain the ID of the issuer. The NameID format MUST be
389 urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified.
390 • The Subject element MUST be present and contain the BUSDOX business identifier of the sender.
391 The Format attribute MUST be set to the BusinessIdentifierType URI and the value to the
392 BusinessIdentifierValue defined in Service Metadata Publishing document.
393 • The assertion MUST be digitally signed by the issuer; the signature MUST include the issuer's
394 certificate as a PEM base 64 encoded X509 DER value in the <ds:KeyInfo> element.
395 • Assertions MUST NOT be encrypted at the SAML level; this will be handled by the transport
396 protocols.
397 • An Assertion MUST NOT contain an <AuthzDecisionStatement>.
398 • An Assertion MUST contain exactly one <AuthnStatement> and one <AttributeStatement>.
399 • The <AuthnStatement> element MUST have an "AuthInstant" attribute specifying the time
400 authentication occurred.

² 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

- 401
- 402
- 403
- 404
- 405
- 406
- 407
- 408
- 409
- 410
- 411
- 412
- 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.

413 **5.3.1 Attribute Encoding**

- 414
- 415
- 416
- 417
- 418
- 419
- 420
- 421
- 422
- 423
- 424
- 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.

425

426

427 6 Appendix A

428 6.1 XML Schema for message headers

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

430 7 Appendix B: Non-normative SAML token example

431 7.1 "Sender-vouches" subject confirmation

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

```
437
438 <saml:Assertion ID="a12312312312312372975934659137459
439 871324587613845613984756981346598314634513451345"
440   IssueInstant="2001-12-31T12:00:00"
441   Version="2.0"
442   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
443   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
444   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
445   xmlns:xs="http://www.w3.org/2001/XMLSchema">
446
447   <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
448     format:unspecified">
449     http://SomeAccessPoint.busdox.org</saml:Issuer>
450   <ds:Signature>
451     <ds:SignedInfo>
452       <ds:CanonicalizationMethod
453         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
454       <ds:SignatureMethod
455         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
456       <ds:Reference URI="#idvalue31231231231312">
457         <ds:Transforms>
458           <ds:Transform Algorithm=
459             "http://www.w3.org/2000/09/xmldsig#envelopedsignature" />
460         </ds:Transforms>
461         <ds:DigestMethod
462           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
463         <ds:DigestValue>
464           TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
465         </ds:Reference>
466       </ds:SignedInfo>
467     <ds:SignatureValue>
468       x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwRCKrtwPS6vdVNCcY5rHa
469       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen
470       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
471     </ds:SignatureValue>
472     <ds:KeyInfo>
473       <ds:X509Data>
474         <!-- The Access Point's certificate -->
```

```

475         <ds:X509Certificate>
476         MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
477         MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBACTB01hZGlzb24xIDAeBgNVBAoT
478         F1VuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLEyJEaXZpc21...
479         </ds:X509Certificate>
480     </ds:X509Data>
481 </ds:KeyInfo>
482 </ds:Signature>
483
484     <saml:Subject>
485         <!-- Here comes a NameID indicating the business identifier of the
486 sender -->
487         <saml:NameID
488 Format="http://busdox.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifi
489 er/1.0/">
490             0010:5798000000001
491         </saml:NameID>
492
493         <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:sender-
494 vouches"/>
495     </saml:Subject>
496
497     <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
498         <saml:AuthnContext>
499             <saml:AuthnContextClassRef>
500                 urn:oasis:names:tc:SAML:2.0:ac:classes:X509
501             </saml:AuthnContextClassRef>
502         </saml:AuthnContext>
503     </saml:AuthnStatement>
504
505     <saml:AttributeStatement>
506         <!-- Assurance Level Attribute -->
507         <saml:Attribute
508             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
509             Name="urn:eu:busdox:attribute:assurance-level">
510             <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>
511         </saml:Attribute>
512     </saml:AttributeStatement>
513 </saml:Assertion>
514

```

515 7.2 “Holder-of-key” Subject Confirmation

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

```

519 <saml:Assertion ID="a12312312312312372975934659137459
520 871324587613845613984756981346598314634513451345"
521 IssueInstant="2001-12-31T12:00:00"
522 Version="2.0"
523 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
524 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
525 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
526 xmlns:xs="http://www.w3.org/2001/XMLSchema">
527
528     <saml:Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-
529         format:unspecified">

```

```

530     http://SomeTokenService.busdox.org</saml:Issuer>
531 <ds:Signature>
532   <ds:SignedInfo>
533     <ds:CanonicalizationMethod
534       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
535     <ds:SignatureMethod
536       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
537     <ds:Reference URI="#idvalue31231231231312">
538       <ds:Transforms>
539         <ds:Transform Algorithm=
540           "http://www.w3.org/2000/09/xmldsig#envelopedsignature" />
541       </ds:Transforms>
542       <ds:DigestMethod
543         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
544       <ds:DigestValue>
545         TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
546     </ds:Reference>
547   </ds:SignedInfo>
548   <ds:SignatureValue>
549     x/GyPbzmFEe85pGD3claxG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHa
550     EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaK1ywS7gFgsD01qjyen
551     w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
552   </ds:SignatureValue>
553   <ds:KeyInfo>
554     <ds:X509Data>
555       <!-- The Access Point's certificate -->
556       <ds:X509Certificate>
557         MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
558         MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
559         F1VuaXZlcnNpdHkqb2YgV21zY29uc2luMSswKQYDVQQLEyJEaXZpc21...
560       </ds:X509Certificate>
561     </ds:X509Data>
562   </ds:KeyInfo>
563 </ds:Signature>
564
565 <saml:Subject>
566   <!-- Here comes a NameID indicating the business identifier of the
567 sender -->
568   <saml:NameID
569 Format="http://busdox.org/profiles/serviceMetadata/1.0/UniversalBusinessIdentifi
570 er/1.0/">
571     0010:5798000000001
572   </saml:NameID>
573
574   <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-
575 of-key">
576
577 <!-- Here comes a NameID indicating the ID of the sending Access Point who must
578 confirm with a key -->
579   <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
580 format:unspecified">
581     http://SomeAccessPoint.busdox.org
582   </saml:NameID>
583
584   <!-- Here comes info on the key to confirm with (same as signing key) -->
585   <saml:SubjectConfirmationData
586 xsi:type="saml:KeyInfoConfirmationDataType" NotOnOrAfter="2009-12-31T12:00:00">
587     <ds:KeyInfo>
588       <ds:X509Data>

```

```

589         <ds:X509Certificate>
590             <!-- Here comes the AP's X509 cert -->
591             MIICyjCCAjOgAwIBA.
592         </ds:X509Certificate>
593     </ds:X509Data>
594 </ds:KeyInfo>
595 </saml:SubjectConfirmationData>
596
597 </saml:SubjectConfirmation>
598 </saml:Subject>
599
600 <saml:AuthnStatement AuthnInstant="2009-01-31T12:00:00Z">
601     <saml:AuthnContext>
602         <saml:AuthnContextClassRef>
603             urn:oasis:names:tc:SAML:2.0:ac:classes:X509
604         </saml:AuthnContextClassRef>
605     </saml:AuthnContext>
606 </saml:AuthnStatement>
607
608 <saml:AttributeStatement>
609     <!-- Assurance Level Attribute -->
610     <saml:Attribute
611         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
612         Name="urn:eu:busdox:attribute:assurance-level">
613         <saml:AttributeValue xsi:type="xs:string">3</saml:AttributeValue>
614     </saml:Attribute>
615 </saml:AttributeStatement>
616 </saml:Assertion>
617

```

618 8 Appendix C – WS-SecurityPolicy for START

619 The WS-SecurityPolicy for this profile is as follows:

```

620 <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
621     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
622     utility-1.0.xsd"
623     wsu:Id="BUSDOX">
624     <wsp:ExactlyOne>
625         <wsp>All>
626             <sp:TransportBinding
627                 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
628                 <wsp:Policy>
629                     <sp:TransportToken>
630                         <wsp:Policy>
631                             <sp:HttpsToken RequireClientCertificate="false" />
632                         </wsp:Policy>
633                     </sp:TransportToken>
634                     <sp:AlgorithmSuite>
635                         <wsp:Policy>
636                             <sp:Basic256 />
637                         </wsp:Policy>
638                     </sp:AlgorithmSuite>
639                     <sp:Layout>
640                         <wsp:Policy>
641                             <sp:Lax />
642                         </wsp:Policy>

```

```

643         </sp:Layout>
644         <sp:IncludeTimestamp />
645     </wsp:Policy>
646 </sp:TransportBinding>
647 <sp:EncryptedSupportingTokens
648     xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
649     <wsp:Policy>
650         <sp:IssuedToken
651             sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
652             <Issuer
653                 xmlns="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
654                 <Address xmlns="http://www.w3.org/2005/08/addressing">
655                     https://kirillgdev04/Security_Federation_SecurityTokenService_Indigo/Symmetric.s
656                     vc/Scenario_1_IssuedTokenOverTransport_UsernameOverTransport
657                 </Address>
658             </Issuer>
659             <sp:RequestSecurityTokenTemplate
660                 xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">
661                 <t:TokenType>urn:oasis:names:tc:SAML:2.0:assertion
662                 </t:TokenType>
663                 <t:KeyType>
664                     http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
665                 </t:KeyType>
666                 <t:KeySize>256</t:KeySize>
667             </sp:RequestSecurityTokenTemplate>
668             <wsp:Policy>
669                 <sp:RequireInternalReference />
670             </wsp:Policy>
671         </sp:IssuedToken>
672     </wsp:Policy>
673 </sp:EncryptedSupportingTokens>
674 <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
675     <sp:Policy>
676         <sp:MustSupportRefKeyIdentifier />
677         <sp:MustSupportRefIssuerSerial />
678         <sp:MustSupportRefThumbprint />
679         <sp:RequireSignatureConfirmation />
680     </sp:Policy>
681 </sp:Wss11>
682 <sp:Wss10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
683     <sp:Policy>
684         <sp:MustSupportRefKeyIdentifier />
685         <sp:MustSupportRefIssuerSerial />
686     </sp:Policy>
687 </sp:Wss10>
688 <sp:SignedParts
689     xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
690     <sp:Body />
691 </sp:SignedParts>
692 </wsp:All>
693 </wsp:ExactlyOne>
694 </wsp:Policy>
695 <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
696     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
697     utility-1.0.xsd"

```

```

702     wsu:Id="BUSDOX">
703     <wsp:ExactlyOne>
704     <wsp:All>
705     <sp:TransportBinding
706     xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
707     <wsp:Policy>
708     <sp:TransportToken>
709     <wsp:Policy>
710     <sp:HttpsToken RequireClientCertificate="false" />
711     </wsp:Policy>
712     </sp:TransportToken>
713     <sp:AlgorithmSuite>
714     <wsp:Policy>
715     <sp:Basic256 />
716     </wsp:Policy>
717     </sp:AlgorithmSuite>
718     <sp:Layout>
719     <wsp:Policy>
720     <sp:Lax />
721     </wsp:Policy>
722     </sp:Layout>
723     <sp:IncludeTimestamp />
724     </wsp:Policy>
725     </sp:TransportBinding>
726     <sp:EncryptedSupportingTokens
727     xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
728     <wsp:Policy>
729     <sp:IssuedToken
730     sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeTok
731     en/AlwaysToRecipient">
732     <Issuer
733     xmlns="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
734     <Address xmlns="http://www.w3.org/2005/08/addressing">
735     https://kirillgdev04/Security_Federation_SecurityTokenService_Indigo/Symmetric.s
736     vc/Scenario_1_IssuedTokenOverTransport_UsernameOverTransport
737     </Address>
738     </Issuer>
739     <sp:RequestSecurityTokenTemplate
740     xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">
741     <t:TokenType>urn:oasis:names:tc:SAML:2.0:assertion
742     </t:TokenType>
743     <t:KeyType>
744     http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
745     </t:KeyType>
746     <t:KeySize>256</t:KeySize>
747     </sp:RequestSecurityTokenTemplate>
748     <wsp:Policy>
749     <sp:RequireInternalReference />
750     </wsp:Policy>
751     </sp:IssuedToken>
752     </wsp:Policy>
753     </sp:EncryptedSupportingTokens>
754     <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
755     <sp:Policy>
756     <sp:MustSupportRefKeyIdentifier />
757     <sp:MustSupportRefIssuerSerial />
758     <sp:MustSupportRefThumbprint />

```

```
761         <sp:RequireSignatureConfirmation />
762     </sp:Policy>
763 </sp:Wss11>
764 <sp:Wss10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
765     <sp:Policy>
766         <sp:MustSupportRefKeyIdentifier />
767         <sp:MustSupportRefIssuerSerial />
768     </sp:Policy>
769 </sp:Wss10>
770 <sp:SignedParts
771     xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
772     <sp:Body />
773 </sp:SignedParts>
774 </wsp:All>
775 </wsp:ExactlyOne>
776 </wsp:Policy>
777
778
```