



Web Services Security: SAML Interop 1 Scenarios

Working Draft 04, Jan 29, 2004

Document identifier:

wss-saml-interop1-draft-04.doc

Location:

<http://www.oasis-open.org/committees/wss/>

Editor:

Rich Levinson, Netegrity <rlevinson@netegrity.com>

Hal Lockhart, BEA Systems <hlockhart@bea.com>

Contributors:

Prateek Mishra, Netegrity <pmishra@netegrity.com>

Ron Monzillo, Sun Microsystems <ronald.monzillo@sun.com>

Abstract:

This document documents the four scenarios to be used for the WSS-SAML Interoperability Event.

Status:

Committee members should send comments on this specification to the wss@lists.oasis-open.org list. Others should subscribe to and send comments to the wss-comment@lists.oasis-open.org list. To subscribe, send an email message to wss-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

24 Table of Contents

25	Introduction	5
26	1.1 Terminology	5
27	2 Test Application.....	6
28	3 Scenario #1 – Sender-Vouches: Unsigned.....	7
29	3.1 Agreements	7
30	3.1.1 USERNAME-LIST.....	7
31	3.1.2 ISSUERNAME-LIST	7
32	3.2 Parameters.....	7
33	3.3 General Message Flow	7
34	3.3.1 Message exchange overview	7
35	3.4 First Message - Request.....	8
36	3.4.1 Message Elements and Attributes	8
37	3.4.2 Message Creation.....	9
38	3.4.3 Message Processing.....	10
39	3.4.4 Example (Non-normative).....	10
40	3.5 Second Message - Response.....	11
41	3.5.1 Message Elements and Attributes	11
42	3.5.2 Message Creation	11
43	3.5.3 Message Processing.....	11
44	3.5.4 Example (Non-normative).....	12
45	3.6 Other processing.....	12
46	3.6.1 Requester.....	12
47	3.6.2 Responder.....	12
48	3.7 Expected Security Properties.....	12
49	4 Scenario #2 – Sender-Vouches: Unsigned: SSL.....	13
50	4.1 Agreements	13
51	4.1.1 ISSUERNAME-LIST	13
52	4.1.2 Signature Trust Root.....	13
53	4.1.3 REQUESTER-CERT-VALUE.....	13
54	4.2 Parameters.....	13
55	4.3 General Message Flow	13
56	4.3.1 Message exchange overview	14
57	4.4 First Message - Request.....	15
58	4.4.1 Message Elements and Attributes	15
59	4.4.2 Message Creation	15
60	4.4.3 Message Processing.....	16
61	4.4.4 Example (Non-normative).....	17
62	4.5 Second Message - Response.....	18
63	4.5.1 Message Elements and Attributes	18
64	4.5.2 Message Creation	18

65	4.5.3 Message Processing.....	18
66	4.5.4 Example (Non-normative).....	18
67	4.6 Other processing.....	18
68	4.6.1 Requester.....	19
69	4.6.2 Responder.....	19
70	4.7 Expected Security Properties.....	19
71	5 Scenario #3 – Sender-Vouches: Signed.....	20
72	5.1 Agreements	20
73	5.1.1 ISSUERNAME-LIST	20
74	5.1.2 Signature Trust Root.....	20
75	5.1.3 REQUESTER-CERT-VALUE.....	20
76	5.2 Parameters.....	20
77	5.3 General Message Flow	20
78	5.3.1 Message exchange overview	21
79	5.4 First Message - Request.....	22
80	5.4.1 Message Elements and Attributes	22
81	5.4.2 Message Creation	22
82	5.4.3 Message Processing.....	23
83	5.4.4 Example (Non-normative).....	24
84	5.5 Second Message - Response	26
85	5.5.1 Message Elements and Attributes	26
86	5.5.2 Message Creation	26
87	5.5.3 Message Processing.....	26
88	5.5.4 Example (Non-normative).....	26
89	5.6 Other processing	26
90	5.6.1 Requester.....	26
91	5.6.2 Responder.....	27
92	5.7 Expected Security Properties.....	27
93	6 Scenario #4 – Holder-of-Key.....	28
94	6.1 Agreements	28
95	6.1.1 ISSUERNAME-LIST	28
96	6.1.2 ASSERTIONISSUER-CERT-VALUE.....	28
97	6.1.3 Signature Trust Root.....	28
98	6.2 Parameters.....	28
99	6.3 General Message Flow	28
100	6.3.1 Message exchange overview	29
101	6.4 First Message - Request.....	30
102	6.4.1 Message Elements and Attributes	30
103	6.4.2 Message Creation.....	31
104	6.4.3 Message Processing.....	32
105	6.4.4 Example (Non-normative).....	33
106	6.5 Second Message - Response	35
107	6.5.1 Message Elements and Attributes	35
108	6.5.2 Message Creation	35

109	6.5.3 Message Processing.....	35
110	6.5.4 Example (Non-normative).....	35
111	6.6 Other processing.....	35
112	6.6.1 Requester.....	36
113	6.6.2 Responder.....	36
114	6.7 Expected Security Properties.....	36
115	7 References.....	37
116	7.1 Normative.....	37
117	Appendix A. Ping Application WSDL File	38
118	Appendix B. Revision History	40
119	Appendix C. Notices	41
120		

121 **Introduction**

122 This document describes message exchanges to be tested during the SAML interoperability
123 event of the WSS TC. All use the Request/Response Message Exchange Pattern (MEP) with no
124 intermediaries. All invoke the same simple application.

125 These scenarios are intended to test the interoperability of different implementations performing
126 common operations and to test the soundness of the various specifications and clarity and mutual
127 understanding of their meaning and proper application.

128 THESE SCENARIOS ARE NOT INTENDED TO REPRESENT REASONABLE OR USEFUL
129 PRACTICAL APPLICATIONS OF THE SPECIFICATIONS. THEY HAVE BEEN DESIGNED
130 PURELY FOR THE PURPOSES INDICATED ABOVE AND DO NOT NECESSARILY
131 REPRESENT EFFICIENT OR SECURE MEANS OF PERFORMING THE INDICATED
132 FUNCTIONS. IN PARTICULAR, THESE SCENARIOS ARE KNOWN TO VIOLATE SECURITY
133 BEST PRACTICES IN SOME RESPECTS AND IN GENERAL HAVE NOT BEEN EXTENSIVELY
134 VETTED FOR ATTACKS.

135 **1.1 Terminology**

136 The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*,
137 and *optional* in this document are to be interpreted as described in [[RFC2119](#)].

138 2 Test Application

- 139 All scenarios use the same simple application.
- 140 The Requester sends a Ping element with a value of a string. The value should be the name of
141 the organization that has developed the software and the number of the scenario, e.g. "Acme
142 Corp. – Scenario #1".
- 143 The Responder returns a PingResponse element with a value of the same string.

144 **3 Scenario #1 – Sender-Vouches: Unsigned**

145 The request contains a minimal sender-vouches SAML assertion with no optional elements
146 included. There are no signatures or certificates required. The response does not contain a
147 security header.

148 **3.1 Agreements**

149 This section describes the agreements that must be made, directly or indirectly, between the
150 parties who wish to interoperate.

151 **3.1.1 USERNAME-LIST**

152 This is a list of usernames associated with the test users that participate in the test scenario.

153 **3.1.2 ISSUERNAME-LIST**

154 This is a list of trusted issuers of SAML assertions.

155 **3.2 Parameters**

156 This section describes parameters that are required to correctly create or process messages, but
157 not a matter of mutual agreement.

158 No parameters required.

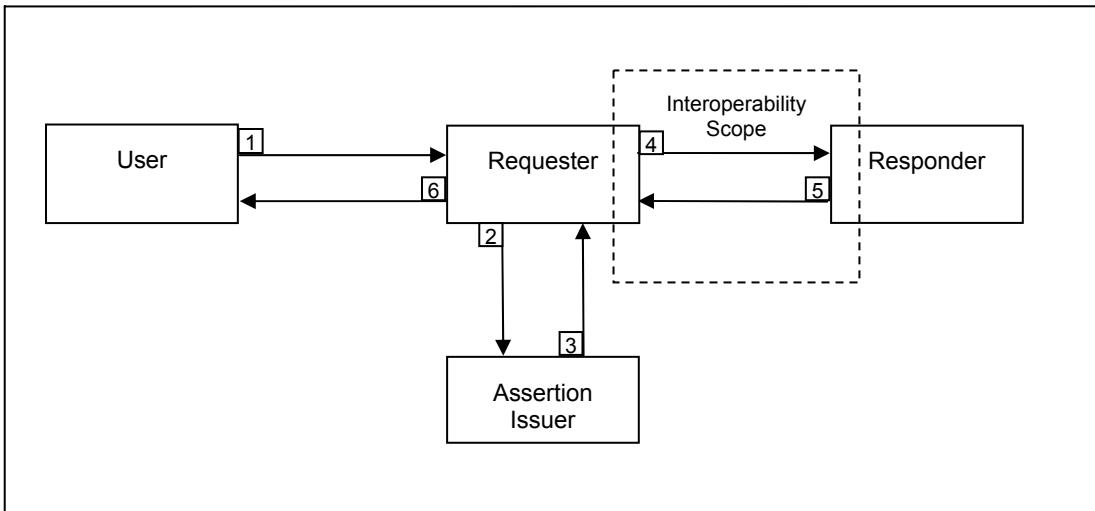
159 **3.3 General Message Flow**

160 This section provides a general overview of the flow of messages.

161 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
162 As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including
163 a null string may be used. The recipient SHOULD ignore the value. The request contains a plain
164 text SAML assertion which contains one valid SubjectStatement of any valid type. The Responder
165 checks the issuer name in the SAML assertion and determines if it is valid against the local
166 directory of trusted issuers. If no errors are detected it returns the response without any security
167 mechanisms.

168 **3.3.1 Message exchange overview**

169 This section contains a high level diagram of the scenario including the actors and the basis of
170 trust. Interoperability for all scenarios is between the Requester and Responder. For each
171 scenario, a hypothetical set of actions that take place prior to the Requester sending the request
172 will be described in order to give some context for the assembly of the request and to show where
173 the basis of trust lies for the Responder. However, the interoperability aspect of each scenario
174 consists solely of the Request that the Requester sends to the Responder and the Response that
175 the Responder returns to the Requester.



177

178

179 In the **Sender Vouches: Unsigned** scenario there is **no technical basis for trust**, because the
 180 messages are sent in the clear with no content or channel protection. This scenario is intended
 181 only to demonstrate message structure interoperability and is not intended for production use. In
 182 typical scenarios the Requester and Assertion Issuer may be the same party however this is not a
 183 requirement and does not impact the interoperability characteristics of this scenario.

184 In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a
 185 potential before and after context for the operability test which only includes steps 4 and 5.

- 186 1. User sends a SOAP request to the Requester
- 187 2. Requester requests a SAML assertion for this User from the Assertion Issuer.
- 188 3. Assertion Issuer returns a Sender-Vouches SAML assertion for the User to the
 Requester.
- 189 4. Requester inserts the Assertion to a wsse:Security header in the SOAP message
 as described in the “First Message” section below and sends the Request to the
 Responder.
- 190 5. Responder processes Request as described in “Message Processing” section
 below and returns Response to Requester as described in “Second Message”
 section below.
- 191 6. Requester returns Response to User.

197

198 3.4 First Message - Request

199 3.4.1 Message Elements and Attributes

200 Items not listed in the following table MAY be present, but MUST NOT be marked with the
 201 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
 202 Items marked as optional MAY be generated and MUST be processed if generated. Items MUST
 203 appear in the order specified, except as noted.

204

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Issuer	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
Body	Mandatory

205

206 **3.4.2 Message Creation**

207 **3.4.2.1 Timestamp**

208 The Created element within the Timestamp SHOULD contain the current local time at the sender
209 expressed in the UTC time zone.

210 **3.4.2.2 Security**

211 The Security element MUST contain the mustUnderstand="1" attribute.

212 **3.4.2.3 Assertion**

213 The Assertion element MUST contain an Issuer attribute, whose value MUST match an Issuer
214 value in the ISSUERNAME-LIST.

215 **3.4.2.4 SubjectStatement**

216 The Assertion element MUST contain one SubjectStatement, which may be any of the standard
217 SAML SubjectStatement extensions.

218 **3.4.2.4.1 ConfirmationMethod**

219 The Subject element MUST also contain a SubjectConfirmation element which MUST contain a
220 ConfirmationMethod with a value of "urn:oasis:names:tc:SAML:cm:sender-vouches".

221 **3.4.2.5 Body**

222 The Body is not signed or encrypted in any way.

223 **3.4.3 Message Processing**

224 This section describes the processing performed by the Responder. If an error is detected, the
225 Responder MUST cease processing the message. If an error is detected, a SOAP Fault MAY be
226 returned and, if so, it MUST have a value of InvalidSecurityToken.

227 **3.4.3.1 Security**

228 **3.4.3.2 Timestamp**

229 The Timestamp element MUST be ignored.

230 **3.4.3.3 Assertion**

231 The Assertion element MUST be validated. The value of the Issuer attribute MUST match an
232 entry in the ISSUERNAME-LIST.

233 **3.4.3.4 SubjectStatement**

234 The Responder MUST check that at least one of the standard SAML SubjectStatement elements
235 is included in the Assertion. Any elements or attributes in the statement other than those explicitly
236 described here SHOULD be ignored.

237 **3.4.3.4.1 ConfirmationMethod**

238 The value of the ConfirmationMethod element MUST be “urn:oasis:names:tc:SAML:cm:sender-
239 vouches”.

240 **3.4.3.5 Body**

241 The Body is passed to the application without modification.

242 **3.4.4 Example (Non-normative)**

243 Here is an example request.

```
244 <?xml version="1.0" encoding="utf-8" ?>
245 <soap:Envelope
246   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
247   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
248   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
249   xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
250   xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
251   <soap:Header>
252     <wsse:Security soap:mustUnderstand="1">
253       <wsu:Timestamp>
254         <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
255       </wsu:Timestamp>
```

```

256     <saml:Assertion
257         xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
258         MajorVersion="1" MinorVersion="0"
259         AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
260         Issuer="www.example.com"
261         IssueInstant="2002-06-19T16:58:33.173Z">
262     <saml:AuthenticationStatement
263         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
264         AuthenticationInstant="2002-06-19T16:57:30.000Z">
265         <saml:Subject>
266             <saml:NameIdentifier
267                 NameQualifier="www.example.com"
268                 Format="">
269                 uid=joe,ou=people,ou=saml-demo,o=example.com
270             </saml:NameIdentifier>
271             <saml:SubjectConfirmation>
272                 <saml:ConfirmationMethod>
273                     urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
274                 </saml:ConfirmationMethod>
275             </saml:SubjectConfirmation>
276         </saml:Subject>
277     </saml:AuthenticationStatement>
278     </saml:Assertion>
279     </wsse:Security>
280 </soap:Header>
281 <soap:Body>
282     <Ping xmlns="http://xmlsoap.org/Ping">
283         <text>EchoString</text>
284     </Ping>
285 </soap:Body>
286 </soap:Envelope>

```

287 **3.5 Second Message - Response**

288 **3.5.1 Message Elements and Attributes**

289 Items not listed in the following table MUST NOT be created or processed. Items marked
 290 mandatory MUST be generated and processed. Items marked optional MAY be generated and
 291 MUST be processed if present. Items MUST appear in the order specified, except as noted.

292

Name	Mandatory?
Body	Mandatory

293

294 **3.5.2 Message Creation**

295 The response message MUST NOT contain a <wsse:Security> header. Any other header
 296 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

297 **3.5.3 Message Processing**

298 The Body is passed to the application without modification.

299 **3.5.4 Example (Non-normative)**

300 Here is an example response.

```
301 <?xml version="1.0" encoding="utf-8" ?>
302 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" 
303   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
304   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
305   <soap:Body>
306     <PingResponse xmlns="http://xmlsoap.org/Ping">
307       <text>EchoString</text>
308     </PingResponse>
309   </soap:Body>
310 </soap:Envelope>
```

311 **3.6 Other processing**

312 This section describes processing that occurs outside of generating or processing a message.

313 **3.6.1 Requester**

314 No additional processing is required.

315 **3.6.2 Responder**

316 No additional processing is required.

317 **3.7 Expected Security Properties**

318 Use of the service is restricted to Requesters that can obtain and include a SAML assertion with
319 an Issuer name that is on the Responder's ISSUERNAMELIST.

320 **4 Scenario #2 – Sender-Vouches: Unsigned: SSL**

321 The request contains a sender-vouches SAML assertion. There are no signatures required. This
322 scenario MUST be tested over SSL and certificates are required to support SSL at the transport
323 layer. The response does not contain a security header.

324 **4.1 Agreements**

325 This section describes the agreements that must be made, directly or indirectly, between the
326 parties who wish to interoperate.

327 **4.1.1 ISSUERNAME-LIST**

328 This is a list of trusted issuers of SAML assertions.

329 **4.1.2 Signature Trust Root**

330 This refers generally to agreeing on at least one trusted key and any other certificates and
331 sources of revocation information sufficient to validate certificates sent for the purpose of verifying
332 the identity of the Requester.

333 **4.1.3 REQUESTER-CERT-VALUE**

334 This is an opaque identifier indicating the X.509 certificate to be used by the Requester. The
335 Responder MUST have the necessary trusted certificates in the Signature Trust Root to validate
336 the Requester certificate.

337 **4.2 Parameters**

338 This section describes parameters that are required to correctly create or process messages, but
339 not a matter of mutual agreement.

340 No parameters required.

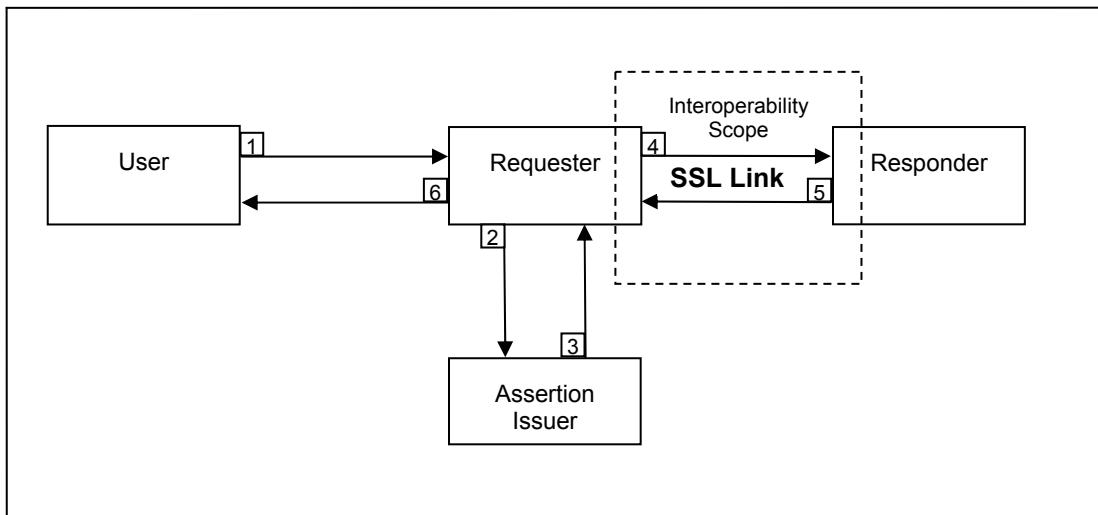
341 **4.3 General Message Flow**

342 This section provides a general overview of the flow of messages.

343 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
344 As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including
345 a null string may be used. The recipient SHOULD ignore the value. The request contains a plain
346 text SAML assertion which contains one valid SubjectStatement of any valid type. The Responder
347 checks the issuer name in the SAML assertion and determines if it is valid against the local
348 directory of trusted issuers. If no errors are detected it returns the response without any security
349 mechanisms.

350 4.3.1 Message exchange overview

351 This section contains a high level diagram of the scenario including the actors and the basis of
352 trust. Interoperability for all scenarios is between the Requester and Responder. For each
353 scenario, a hypothetical set of actions that take place prior to the Requester sending the request
354 will be described in order to give some context for the assembly of the request and to show where
355 the basis of trust lies for the Responder. However, the interoperability aspect of each scenario
356 consists solely of the Request that the Requester sends to the Responder and the Response that
357 the Responder returns to the Requester.



358

359 In the **Sender-Vouches: SSL** scenario the basis of trust is the Requester's client certificate used
360 to establish the SSL link. The Responder relies on the Requester who vouches for the contents of
361 the User message and the SAML Assertion. In typical scenarios the Requester and Assertion
362 Issuer may be the same party however this is not a requirement and does not impact the
363 interoperability characteristics of this scenario.

364 In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a
365 potential before and after context for the operability test which only includes steps 4 and 5.

366

- 367 1. User sends a SOAP request to the Requester
- 368 2. Requester requests a SAML assertion for this User from the Assertion Issuer.
- 369 3. Assertion Issuer returns a Sender-Vouches SAML assertion for the User to the
370 Requester.
- 371 4. Requester inserts the Assertion to a wsse:Security header in the SOAP message
372 as described in the "First Message" section below and sends the Request to the
373 Responder.
- 374 5. Responder processes Request as described in "Message Processing" section
375 below and returns Response to Requester as described in "Second Message"
376 section below.
- 377 6. Requester returns Response to User.

378

379 **4.4 First Message - Request**

380 **4.4.1 Message Elements and Attributes**

381 Items not listed in the following table MAY be present, but MUST NOT be marked with the
382 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
383 Items marked as optional MAY be generated and MUST be processed if generated. Items MUST
384 appear in the order specified, except as noted.

385

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Issuer	Mandatory
Conditions	Mandatory
NotBefore	Mandatory
NotOnOrAfter	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
Body	Mandatory

386

387 **4.4.2 Message Creation**

388 **4.4.2.1 Security**

389 The Security element MUST contain the mustUnderstand="1" attribute.

390 **4.4.2.2 Timestamp**

391 The Created element within the Timestamp SHOULD contain the current local time at the sender
392 expressed in the UTC time zone.

393 **4.4.2.3 Assertion**

394 The Assertion element MUST contain an Issuer attribute, whose value MUST match an Issuer
395 value in the ISSUERNAME-LIST.

396 **4.4.2.3.1 Conditions**

397 The Conditions element MUST be present and contain valid values for the NotBefore and
398 NotOnOrAfter attributes.

399 **4.4.2.3.2 SubjectStatement**

400 The Assertion element MUST contain one SubjectStatement, which may be any of the standard
401 SAML SubjectStatement extensions.

402 **4.4.2.3.3 ConfirmationMethod**

403 The Subject element MUST also contain a SubjectConfirmation element which MUST contain a
404 ConfirmationMethod with a value of “urn:oasis:names:tc:SAML:cm:sender-vouches”.

405 **4.4.2.4 Body**

406 The Body is not signed or encrypted in any way.

407 **4.4.3 Message Processing**

408 This section describes the processing performed by the Responder. If an error is detected, the
409 Responder MUST cease processing the message. If an error is detected, a SOAP Fault MAY be
410 returned and, if so, it MUST have a value of InvalidSecurityToken.

411 **4.4.3.1 Security**

412 **4.4.3.2 Timestamp**

413 The Timestamp element MUST be ignored.

414 **4.4.3.3 Assertion**

415 The Assertion element MUST be validated. The value of the Issuer attribute MUST match an
416 entry in the ISSUERNAME-LIST.

417 **4.4.3.3.1 Conditions**

418 As part of validating the Assertion element, the NotBefore and NotOnOrAfter attributes MUST be
419 consistent with the current UTC time.

420 **4.4.3.3.2 SubjectStatement**

421 The Responder MUST check that at least one of the standard SAML SubjectStatement elements
422 is included in the Assertion. Any elements or attributes in the statement other than those explicitly
423 described here SHOULD be ignored.

424 **4.4.3.3.3 ConfirmationMethod**

425 The value of the ConfirmationMethod element MUST be "urn:oasis:names:tc:SAML:cm:sender-
426 vouches".

427 **4.4.3.4 Body**

428 The Body is passed to the application without modification.

429 **4.4.4 Example (Non-normative)**

430 Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurit-secext-01.xsd"
    xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurit-utility-1.0.xsd">
    <soap:Header>
        <wsse:Security soap:mustUnderstand="1">
            <wsu:Timestamp>
                <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
            </wsu:Timestamp>
            <saml:Assertion
                xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
                MajorVersion="1" MinorVersion="0"
                AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
                Issuer="www.example.com"
                IssueInstant="2002-06-19T16:58:33.173Z">
                <saml:Conditions
                    NotBefore="2002-06-19T16:53:33.173Z"
                    NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
                <saml:AuthenticationStatement
                    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
                    AuthenticationInstant="2002-06-19T16:57:30.000Z">
                    <saml:Subject>
                        <saml:NameIdentifier
                            NameQualifier="www.example.com"
                            Format="">
                            uid=joe,ou=people,ou=saml-demo,o=example.com
                        </saml:NameIdentifier>
                        <saml:SubjectConfirmation>
                            <saml:ConfirmationMethod>
                                urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
                            </saml:ConfirmationMethod>
                        </saml:SubjectConfirmation>
                    </saml:Subject>
                </saml:AuthenticationStatement>
```

```

470         </saml:Assertion>
471     </wsse:Security>
472   </soap:Header>
473   <soap:Body>
474     <Ping xmlns="http://xmlsoap.org/Ping">
475       <text>EchoString</text>
476     </Ping>
477   </soap:Body>
478 </soap:Envelope>

```

479 **4.5 Second Message - Response**

480 **4.5.1 Message Elements and Attributes**

481 Items not listed in the following table MUST NOT be created or processed. Items marked
 482 mandatory MUST be generated and processed. Items marked optional MAY be generated and
 483 MUST be processed if present. Items MUST appear in the order specified, except as noted.

484

Name	Mandatory?
Body	Mandatory

485

486 **4.5.2 Message Creation**

487 The response message MUST NOT contain a <wsse:Security> header. Any other header
 488 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

489 **4.5.3 Message Processing**

490 The Body is passed to the application without modification.

491 **4.5.4 Example (Non-normative)**

492 Here is an example response.

```

493 <?xml version="1.0" encoding="utf-8" ?>
494 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" 
495   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
496   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
497   <soap:Body>
498     <PingResponse xmlns="http://xmlsoap.org/Ping">
499       <text>EchoString</text>
500     </PingResponse>
501   </soap:Body>
502 </soap:Envelope>

```

503 **4.6 Other processing**

504 This section describes processing that occurs outside of generating or processing a message.

505 **4.6.1 Requester**

506 No additional processing is required.

507 **4.6.2 Responder**

508 No additional processing is required.

509 **4.7 Expected Security Properties**

510 Use of the service is restricted to Requesters that can obtain and include a SAML assertion with
511 an Issuer name that is on the Responder's ISSUERNAMELIST. In addition, the Requester must
512 use a client certificate on an SSL link that the Responder can validate.

513 **5 Scenario #3 – Sender-Vouches: Signed**

514 The request contains a sender-vouches SAML assertion. The Assertion and the Body elements
515 are signed. A reference to the certificate used to verify the signature is provided in the header.
516 The response does not contain a security header.

517 **5.1 Agreements**

518 This section describes the agreements that must be made, directly or indirectly between parties
519 who wish to interoperate.

520 **5.1.1 ISSUERNAME-LIST**

521 This is a list of trusted issuers of SAML assertions. (This is optional in the signed scenario since
522 the basis for trust is the Requester certificate that is used for signing.)

523 **5.1.2 Signature Trust Root**

524 This refers generally to agreeing on at least one trusted key and any other certificates and
525 sources of revocation information sufficient to validate certificates sent for the purpose of
526 signature verification.

527 **5.1.3 REQUESTER-CERT-VALUE**

528 This is an opaque identifier indicating the X.509 certificate to be used by the Requester. The
529 Responder MUST have the necessary trusted certificates in the Signature Trust Root to find and
530 validate the Requester certificate.

531 **5.2 Parameters**

532 This section describes parameters that are required to correctly create or process messages, but
533 not a matter of mutual agreement.

534 No parameters required.

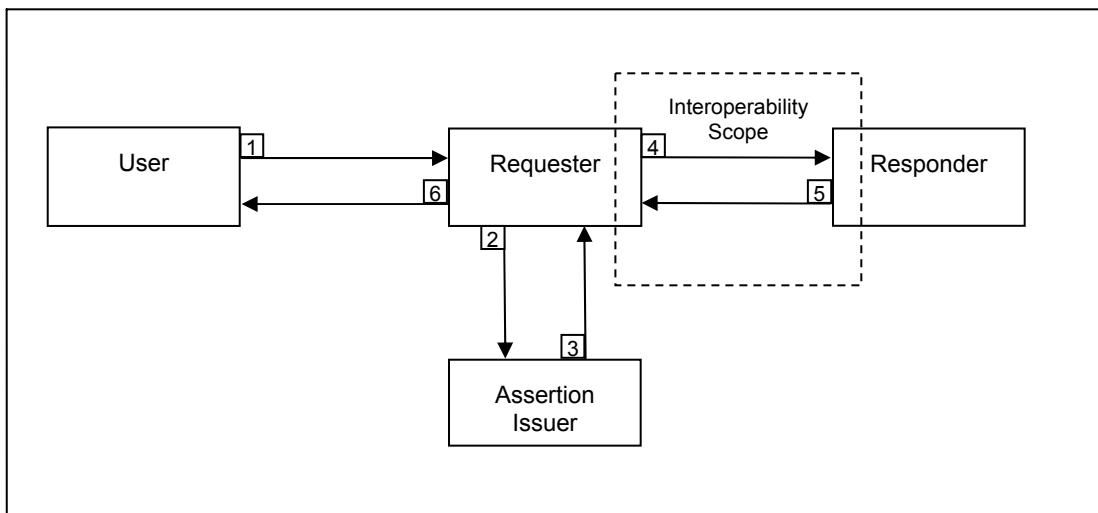
535 **5.3 General Message Flow**

536 This section provides a general overview of the flow of messages.

537 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
538 As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including
539 a null string may be used. The recipient SHOULD ignore the value. The request contains a plain
540 text unsigned SAML assertion. The Responder checks the issuer name in the SAML assertion,
541 validates the Requester certificate referenced in the signature, and verifies the signature. If no
542 errors are detected, the message is delivered to the application and the application returns the
543 response without any security mechanisms.

544 **5.3.1 Message exchange overview**

545 This section contains a high level diagram of the scenario including the actors and the basis of
546 trust. Interoperability for all scenarios is between the Requester and Responder. For each
547 scenario, a hypothetical set of actions that take place prior to the Requester sending the request
548 will be described in order to give some context for the assembly of the request and to show where
549 the basis of trust lies for the Responder. However, the interoperability aspect of each scenario
550 consists solely of the Request that the Requester sends to the Responder and the Response that
551 the Responder returns to the Requester.



552
553 In the **Sender-Vouches: Signed** scenario the basis of trust is the Requester's certificate. The
554 Requester private key is used to sign both the SAML Assertion and the message Body. The
555 Responder relies on the Requester who vouches for the contents of the User message and the
556 SAML Assertion. In typical scenarios the Requester and Assertion Issuer may be the same party
557 however this is not a requirement and does not impact the interoperability characteristics of this
558 scenario.

559 In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a
560 potential before and after context for the interoperability test which only includes steps 4 and 5.

- 561
- 562 1. User sends a SOAP request to the Requester
 - 563 2. Requester requests a SAML assertion for this user from the Assertion Issuer.
 - 564 3. Assertion Issuer returns a Sender-Vouches SAML assertion to the Requester.
 - 565 4. Requester inserts the Assertion to a wsse:Security header in the SOAP message
566 and signs the Assertion and message Body as described in the "First Message"
567 section below and sends the Request to the Responder.
 - 568 5. Responder processes Request as described in "Message Processing" section
569 below and returns Response to Requester as described in "Second Message"
570 section below.
 - 571 6. Requester returns Response to User.

572 **5.4 First Message - Request**

573 **5.4.1 Message Elements and Attributes**

574 Items not listed in the following table MAY be present, but MUST NOT be marked with the
575 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
576 Items marked optional MAY be generated and MUST be processed if present. Items MUST
577 appear in the order specified, except as noted.

578

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Conditions	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory

579

580 **5.4.2 Message Creation**

581 **5.4.2.1 Security**

582 The Security element MUST contain the mustUnderstand="1" attribute.

583 **5.4.2.2 Timestamp**

584 The Created element within the Timestamp SHOULD contain the current local time at the sender.

585 **5.4.2.3 Signature**

586 The signature is over the entire SOAP Body plus over the SAML Assertion.

587 **5.4.2.3.1 SignedInfo**

588 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
589 CanonicalizationMethod MUST be Exclusive Canonicalization. There MUST be 2 Reference
590 elements, one of which MUST specify a relative URI that refers to the SOAP Body element, and
591 the other which MUST specify a relative URI that refers to the Assertion element.

592 **5.4.2.3.2 SignatureValue**

593 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
594 private key corresponding to the public key specified in the certificate identified by the KeyInfo
595 element.

596 **5.4.2.3.3 KeyInfo**

597 The KeyInfo MUST contain an X509SubjectName that identifies the Requester certificate which
598 will be used for signature verification.

599 **5.4.2.4 Assertion**

600 The Assertion MUST be signed and be labeled with an ID (AssertionID) that is referenced in a
601 SignedInfo Reference element of the Signature in the Security element.

602 The Conditions element MUST be present and contain valid NotBefore and NotOnOrAfter
603 attributes

604 A SubjectStatement element MUST be present and MUST contain a SubjectConfirmation
605 element which MUST contain a ConfirmationMethod element with a value of
606 “urn:oasis:tc:SAML:cm:sender-vouches”.

607 **5.4.2.5 Body**

608 The Body element MUST be signed.

609 **5.4.3 Message Processing**

610 This section describes the processing performed by the Responder. If an error is detected, the
611 Responder MUST cease processing the message and issue a Fault with a value of
612 InvalidSecurityToken.

613 **5.4.3.1 Security**

614 **5.4.3.2 Timestamp**

615 The Timestamp element MUST be ignored.

616 **5.4.3.3 Signature**

617 The Requester certificate MUST be obtained based on the KeyInfo/SecurityTokenReference and
618 be validated against the Signature Trust Root. The Body and Assertion MUST be verified against
619 the signature using the specified algorithms and transforms and the retained public key.

620 **5.4.3.4 Assertion**

621 The Assertion element MUST be validated. The value of the Issuer attribute MAY match the
622 Subject of the certificate in the X509SubjectName. If it does not, then the value of the Issuer
623 attribute MUST match an entry in the ISSUERNAME-LIST.

624 **5.4.3.5 Body**

625 The body is passed to the application without modification.

626 **5.4.4 Example (Non-normative)**

627 Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<S12:Envelope
  xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
  secext-01.xsd"
  xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
  1.0.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
  <S12:Header>
    <wsse:Security>
      <wsu:Timestamp>
        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
      </wsu:Timestamp>
      <saml:Assertion
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1">
        <saml:Conditions
          NotBefore="2002-06-19T16:53:33.173Z"
          NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
        <saml:AttributeStatement>
          <saml:Subject>
            <saml:NamelistIdentifier
              NameQualifier="www.example.com"
              Format="">
              uid=joeuser,ou=system,ou=saml-demo,o=xyz.com
            </saml:NamelistIdentifier>
          <saml:SubjectConfirmation>
```

```

661 <saml:ConfirmationMethod>
662   urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
663 </saml:ConfirmationMethod>
664 </saml:SubjectConfirmation>
665 </saml:Subject>
666 <saml:Attribute>
667 ...
668 </saml:Attribute>
669 ...
670 </saml:AttributeStatement>
671 </saml:Assertion>
672 <wsse:SecurityTokenReference wsu:id="STR1">
673   <wsse:KeyIdentifier wsu:id="..." ...
674     wsse:ValueType="http://www.docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-
675     token-profile-1.0#SAMLAssertionID">
676     _a75adf55-01d7-40cc-929f-dbd8372ebdfc
677   </wsse:KeyIdentifier>
678 </wsse:SecurityTokenReference>
679 <wsse:BinarySecurityToken
680   wsu:id="attesterCert"
681   wsu:Value="wsse:X509V3"
682   wsu:EncodingType="wsse:Base64Binary">
683   MIIEZzCCA9CgAwIBAgIQEmtJZc0...
684 </wsse:BinarySecurityToken>
685 <ds:Signature>
686 <ds:SignedInfo>
687   <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
688   <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
689   <ds:Reference URI="#STR1">
690     <Transforms>
691       <ds:Transform Algorithm="http://schemas.xmlsoap.org/ws/2003/06/STR-Transform" />
692       <wsse:TransformationParameters>
693         <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
694       </wsse:TransformationParameters>
695     </Transforms>
696     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
697     <ds:DigestValue>...</ds:DigestValue>
698   </ds:Reference>
699   <ds:Reference URI="#MsgBody">
700     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
701     <ds:DigestValue>...</ds:DigestValue>
702   </ds:Reference>
703   </ds:SignedInfo>
704   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
705   <ds:KeyInfo>
706     <wsse:SecurityTokenReference wsu:id="STR2">
707       <wsse:Reference wsu:id="..." ...
708         wsse:URI="attesterCert"/>
709     </wsse:SecurityTokenReference>
710   </ds:KeyInfo>
711   </ds:Signature>
712 </wsse:Security>
713 </S12:Header>
714 <S12:Body wsu:id="MsgBody">
```

```
715 <ReportRequest>
716   <TickerSymbol>SUNW</TickerSymbol>
717 </ReportRequest>
718 </S12:Body>
719 </S12:Envelope>
```

720 **5.5 Second Message - Response**

721 **5.5.1 Message Elements and Attributes**

722 Items not listed in the following table MUST NOT be created or processed. Items marked
723 mandatory MUST be generated and processed. Items marked optional MAY be generated and
724 MUST be processed if present. Items MUST appear in the order specified, except as noted.

725

Name	Mandatory?
Body	Mandatory

726

727 **5.5.2 Message Creation**

728 The response message must not contain a <wsse:Security> header. Any other header elements
729 MUST NOT be labeled with a mustUnderstand="1" attribute.

730 **5.5.3 Message Processing**

731 The body is passed to the application without modification.

732 **5.5.4 Example (Non-normative)**

733 Here is an example response.

```
734 <?xml version="1.0" encoding="utf-8" ?>
735 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" 
736   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
737   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
738   <soap:Body>
739     <PingResponse xmlns="http://xmlsoap.org/Ping">
740       <text>EchoString</text>
741     </PingResponse>
742   </soap:Body>
743 </soap:Envelope>
```

744 **5.6 Other processing**

745 This section describes processing that occurs outside of generating or processing a message.

746 **5.6.1 Requester**

747 No additional processing is required.

748 **5.6.2 Responder**

749 **5.7 Expected Security Properties**

750 Use of the service is restricted to Requesters that can obtain and include a SAML assertion with
751 an Issuer name that is on the Responder's ISSUERNAMELIST. In addition, the Requester must
752 sign both the assertion and the body using the Requester's private key and certificate, which will
753 protect the message against modification if it is intercepted or replayed.

754 **6 Scenario #4 – Holder-of-Key**

755 The request contains a holder-of-key SAML assertion. The assertion is signed by the assertion
756 issuer with an enveloped signature. The certificate used to verify the issuer signature is contained
757 within the assertion signature. The message body is signed by the Requester. The certificate
758 used to verify the Requester's signature is contained in the assertion SubjectConfirmation. The
759 response does not contain a security header.

760 **6.1 Agreements**

761 This section describes the agreements that must be made, directly or indirectly between parties
762 who wish to interoperate.

763 **6.1.1 ISSUERNAME-LIST**

764 This is a list of trusted issuers of SAML assertions.

765 **6.1.2 ASSERTIONISSUER-CERT-VALUE**

766 This is an opaque identifier indicating the X.509 certificate to be used by the Assertion Issuer.
767 The Responder MUST have the necessary trusted certificates in the Signature Trust Root to
768 validate the Assertion Issuer certificate, which in turn is used to validate the User certificate which
769 is contained in the assertion.

770 **6.1.3 Signature Trust Root**

771 This refers generally to agreeing on at least one trusted key and any other certificates and
772 sources of revocation information sufficient to validate certificates sent for the purpose of
773 signature verification.

774 **6.2 Parameters**

775 This section describes parameters that are required to correctly create or process messages, but
776 not a matter of mutual agreement.

777 No parameters are required.

778 **6.3 General Message Flow**

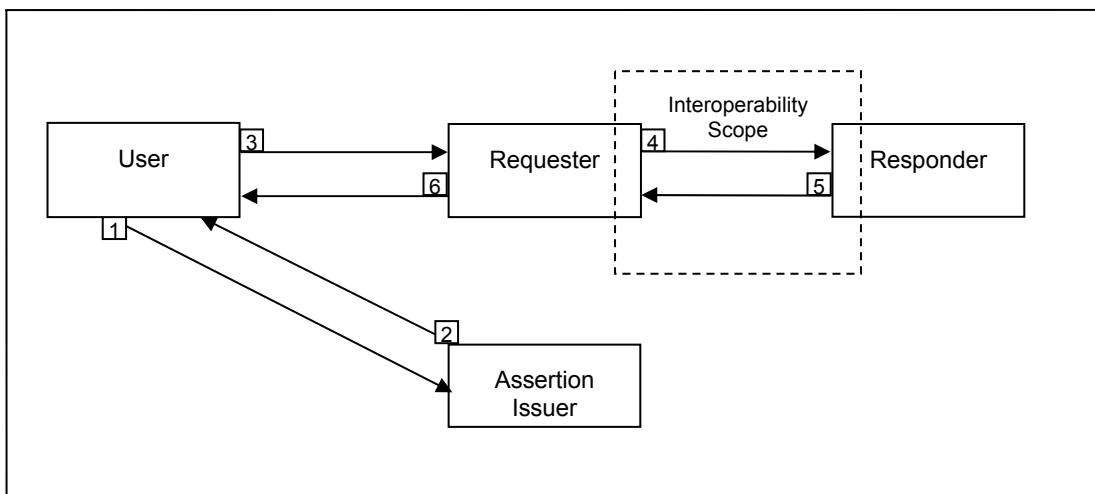
779 This section provides a general overview of the flow of messages.

780 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
781 As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including
782 a null string may be used. The recipient SHOULD ignore the value. The request contains a plain
783 text signed SAML assertion containing the User certificate. The Responder validates the
784 assertion data, verifies the signature on the assertion, validates the issuer certificate contained in
785 that signature, and verifies the signature on the Body using the user certificate/public key which is
786 in the assertion. The order of the previous steps is not significant, although all steps MUST be

787 performed. If no errors are detected, the message is delivered to the application and the
788 application returns the response without any security mechanisms.

789 6.3.1 Message exchange overview

790 This section contains a high level diagram of the scenario including the actors and the basis of
791 trust. Interoperability for all scenarios is between the Requester and Responder. For each
792 scenario, a hypothetical set of actions that take place prior to the Requester sending the request
793 will be described in order to give some context for the assembly of the request and to show where
794 the basis of trust lies for the Responder. However, the interoperability aspect of each scenario
795 consists solely of the Request that the Requester sends to the Responder and the Response that
796 the Responder returns to the Requester.



797
798 In the **Holder-of-Key: Signed** scenario the basis of trust is the Assertion Issuer's certificate. The
799 Assertion Issuer's private key is used to sign the SAML Assertion for the User. The Responder
800 relies on the Assertion Issuer to have issued the assertion to an authorized User. In typical
801 scenarios the User and Requester may be the same party however this is not a requirement and
802 does not impact the interoperability characteristics of this scenario.

- 803
- 804 1. User sends a request for a SAML assertion to the Assertion Issuer.
 - 805 2. The Assertion Issuer authenticates the User and returns a SAML assertion
 - 806 containing the User certificate and signed with the Assertion Issuer private key.
 - 807 3. User inserts the Assertion to a wsse:Security header in the SOAP message and
 - 808 signs the message Body with the User private key, and sends the message to
 - 809 the Requester.
 - 810 4. Requester receives message from User which has form as described in the "First
 - 811 Message" section below and sends the Request to the Responder.
 - 812 5. Responder processes Request as described in "Message Processing" section
 - 813 below and returns Response to Requester as described in "Second Message"
 - 814 section below.
 - 815 6. Requester returns Response to User.

816 **6.4 First Message - Request**

817 **6.4.1 Message Elements and Attributes**

818 Items not listed in the following table MAY be present, but MUST NOT be marked with the
819 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
820 Items marked optional MAY be generated and MUST be processed if present. Items MUST
821 appear in the order specified, except as noted.

822

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Conditions	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
KeyInfo	Mandatory
Signature (Assertion)	Mandatory
SignedInfo	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Signature (Body)	Mandatory
SignedInfo	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory

823

824 **6.4.2 Message Creation**

825 **6.4.2.1 Security**

826 The Security element MUST contain the mustUnderstand="1" attribute.

827 **6.4.2.2 Timestamp**

828 The Created element within the Timestamp SHOULD contain the current local time at the sender.

829 **6.4.2.3 Assertion**

830 The Assertion MUST contain an ISSUER attribute, whose value MUST match an Issuer value in
831 the ISSUERNAME-LIST.

832 **6.4.2.3.1 Conditions**

833 The Conditions element MUST be present and contain valid values for the NotBefore and
834 NotOnOrAfter attributes.

835 **6.4.2.3.2 SubjectConfirmation**

836 The SubjectConfirmation element MUST contain both a ConfirmationMethod element and a
837 KeyInfo element.

838 **6.4.2.3.2.1 ConfirmationMethod**

839 The ConfirmationMethod element MUST have a value of
840 "urn:oasis:names:tc:SAML:1.0:cm:holder-of-key".

841 **6.4.2.3.2.2 KeyInfo**

842 The KeyInfo element MUST contain the user's public key, which MUST be contained in either an
843 X509Data element or a KeyValue element. (Note: because this KeyInfo is part of the SAML
844 assertion token, it is outside the direct scope of WS-Security and does not contain a
845 SecurityTokenReference.)

846 **6.4.2.3.3 Signature (Assertion)**

847 This Signature element is a child of the Assertion element and the signature is over the Assertion
848 element

849 **6.4.2.3.3.1 SignedInfo**

850 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
851 CanonicalizationMethod SHOULD be Exclusive Canonicalization. If not, the Assertion element
852 MAY need to be removed from the request before the signature is verified. There MUST be 1
853 Reference element, which MUST specify a relative URI of "" indicating the parent of the Signature
854 element. The Reference element MUST contain a Transform element with an Algorithm attribute
855 with the value "http://www.w3.org/2000/09/xmldsig#enveloped-signature".

856 **6.4.2.3.3.2 SignatureValue**

857 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
858 private key corresponding to the public key contained in the KeyInfo element in this Signature
859 element.

860 **6.4.2.3.3.3 KeyInfo**

861 This KeyInfo element MUST contain a Base 64 representation of the Issuer's certificate. The
862 certificate MUST contain the Issuer's public key, which can be used to verify the SignatureValue
863 in this Signature element. (Note: because this KeyInfo is part of the SAML assertion token, it is
864 outside the direct scope of WS-Security and does not contain a SecurityTokenReference.)

865 **6.4.2.4 Signature (Body)**

866 This Signature element is a child of the Security element and the signature is over the entire
867 SOAP body.

868 **6.4.2.4.1 SignedInfo**

869 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
870 CanonicalizationMethod MUST be Exclusive Canonicalization. There MUST be 1 Reference
871 element which MUST specify a relative URI that refers to the SOAP Body element.

872 **6.4.2.4.2 SignatureValue**

873 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
874 private key corresponding to the public key specified in the KeyInfo element that is in the
875 Assertion element.

876 **6.4.2.4.3 KeyInfo**

877 This KeyInfo element must contain a SecurityTokenReference that contains a KeyIdentifier
878 element that has a ValueType attribute with a value of "saml:Assertion". The KeyIdentifier MUST
879 have a value equal to the value of the AssertionID attribute of the Assertion element.

880 **6.4.2.5 Body**

881 The body element MUST be signed.

882 **6.4.3 Message Processing**

883 This section describes the processing performed by the Responder. If an error is detected, the
884 Responder MUST cease processing the message and issue a Fault with a value of
885 FailedAuthentication.

886 **6.4.3.1 Timestamp**

887 The Timestamp element MUST be ignored.

888 **6.4.3.2 Security**

889 **6.4.3.3 Assertion**

890 The Assertion element MUST be validated. The value of the Issuer attribute SHOULD match the
891 Subject of the certificate used to verify the signature contained in the assertion. If it does not, the
892 not, then the value of the Issuer attribute MUST match an entry in the ISSUERNAME-LIST. The
893 user public key or certificate MUST be obtained from the KeyInfo element of the
894 SubjectConfirmation element and retained for verification of the Body signature.

895 **6.4.3.3.1 Signature (Assertion)**

896 The Assertion MUST be verified against the signature it contains, using the specified algorithms
897 and transforms and the public key that can be obtained from the assertion issuer certificate
898 contained in the Signature element. The issuer certificate MUST be validated against the
899 Signature Trust Root.

900 **6.4.3.4 Signature (Body)**

901 The Body MUST be verified against the Signature element that is a child of the Security element
902 using the user public key that was retained from the Assertion.

903 **6.4.3.5 Body**

904 The Body is passed to the application without modification.

905 **6.4.4 Example (Non-normative)**

906 Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<S12:Envelope
  xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
  secext-01.xsd"
  xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
  1.0.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <S12:Header>
    <wsse:Security>
      <wsu:Timestamp>
        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
      </wsu:Timestamp>
      <saml:Assertion
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1"
        xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
```

```

929 <saml:Conditions
930   NotBefore="2002-06-19T16:53:33.173Z"
931   NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
932 <saml:AttributeStatement>
933   <saml:Subject>
934     <saml:NameIdentifier
935       NameQualifier="www.example.com"
936       Format="">
937       uid=joe,ou=people,ou=saml-demo,o=baltimore.com
938     </saml:NameIdentifier>
939   <saml:SubjectConfirmation>
940     <saml:ConfirmationMethod>
941       urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
942     </saml:ConfirmationMethod>
943     <ds:KeyInfo>
944       <ds:KeyValue>...</ds:KeyValue>
945     </ds:KeyInfo>
946   </saml:SubjectConfirmation>
947 </saml:Subject>
948 <saml:Attribute
949   AttributeName="MemberLevel"
950   AttributeNamespace="http://www.oasis-open.org/Catalyst2002/attributes">
951     <saml:AttributeValue>gold</saml:AttributeValue>
952   </saml:Attribute>
953   <saml:Attribute
954     AttributeName="E-mail"
955     AttributeNamespace="http://www.oasis-
956     open.org/Catalyst2002/attributes">
957     <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
958   </saml:Attribute>
959 </saml:AttributeStatement>
960 <ds:Signature>...</ds:Signature>
961 </saml:Assertion>
962 <ds:Signature>
963 <ds:SignedInfo>
964   <ds:CanonicalizationMethod
965     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
966   <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
967   <ds:Reference URI="#MsgBody">
968     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
969     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
970   </ds:Reference>
971 </ds:SignedInfo>
972 <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
973 <ds:KeyInfo>
974   <wsse:SecurityTokenReference wsu:id="STR1">
975     <wsse:Reference wsu:id="...">
976       wsse:ValueType="http://www.docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-
977       token-profile-1.0#SAMLAssertion-1.0"
978       wsse:URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc"/>
979     </wsse:SecurityTokenReference>
980   </ds:KeyInfo>
981 </ds:Signature>
982 </wsse:Security>
```

```
983 </S12:Header>
984 <S12:Body wsu:Id="MsgBody">
985 <ReportRequest>
986 <TickerSymbol>SUNW</TickerSymbol>
987 </ReportRequest>
988 </S12:Body>
989 </S12:Envelope>
```

990 **6.5 Second Message - Response**

991 **6.5.1 Message Elements and Attributes**

992 Items not listed in the following table MUST NOT be created or processed. Items marked
993 mandatory MUST be generated and processed. Items marked optional MAY be generated and
994 MUST be processed if present. Items MUST appear in the order specified, except as noted.

995

Name	Mandatory?
Body	Mandatory

996

997 **6.5.2 Message Creation**

998 The response message must not contain a <wsse:Security> header. Any other header elements
999 MUST NOT be labeled with a mustUnderstand="1" attribute.

1000 **6.5.3 Message Processing**

1001 The body is passed to the application without modification.

1002 **6.5.4 Example (Non-normative)**

1003 Here is an example response.

```
1004 <?xml version="1.0" encoding="utf-8" ?>
1005 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" 
1006   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1007   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1008   <soap:Body>
1009     <PingResponse xmlns="http://xmlsoap.org/Ping">
1010       <text>EchoString</text>
1011     </PingResponse>
1012   </soap:Body>
1013 </soap:Envelope>
```

1014 **6.6 Other processing**

1015 This section describes processing that occurs outside of generating or processing a message.

1016 **6.6.1 Requester**

1017 No additional processing is required.

1018 **6.6.2 Responder**

1019 **6.7 Expected Security Properties**

1020 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
1021 of the request is protected against modification and interception. The authorized party must also
1022 obtain a signed SAML assertion containing the authorized party's certificate and include the
1023 assertion in the request as the basis for trust of the authorized party's certificate.

1024

7 References

1025

7.1 Normative

1026

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1027 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.

1028

Appendix A. Ping Application WSDL File

```

1029 <definitions xmlns:tns="http://xmlsoap.org/Ping"
1030   xmlns="http://schemas.xmlsoap.org/wsdl/"
1031   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1032   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1033   targetNamespace="http://xmlsoap.org/Ping" name="Ping">
1034     <types>
1035       <schema targetNamespace="http://xmlsoap.org/Ping"
1036         xmlns="http://www.w3.org/2001/XMLSchema">
1037         <complexType name="ping">
1038           <sequence>
1039             <element name="text" type="xsd:string"
1040               nillable="true"/>
1041           </sequence>
1042         </complexType>
1043         <complexType name="pingResponse">
1044           <sequence>
1045             <element name="text" type="xsd:string"
1046               nillable="true"/>
1047           </sequence>
1048         </complexType>
1049         <element name="Ping" type="tns:ping"/>
1050         <element name="PingResponse" type="tns:pingResponse"/>
1051       </schema>
1052     </types>
1053     <message name="PingRequest">
1054       <part name="ping" element="tns:Ping"/>
1055     </message>
1056     <message name="PingResponse">
1057       <part name="pingResponse" element="tns:PingResponse"/>
1058     </message>
1059     <portType name="PingPort">
1060       <operation name="Ping">
1061         <input message="tns:PingRequest"/>
1062         <output message="tns:PingResponse"/>
1063       </operation>
1064     </portType>
1065     <binding name="PingBinding" type="tns:PingPort">
1066       <soap:binding style="document"
1067         transport="http://schemas.xmlsoap.org/soap/http"/>
1068       <operation name="Ping">
1069         <soap:operation/>
1070         <input>
1071           <soap:body use="literal"/>
1072         </input>
1073         <output>
1074           <soap:body use="literal"/>
1075         </output>
1076       </operation>
1077     </binding>
1078     <service name="PingService">
1079       <port name="PingPort" binding="tns:PingBinding">
1080         <soap:address
1081           location="http://localhost:8080/pingejb/Ping"/>
1082         </port>
1083       </service>
1084     </definitions>

```

1086

Appendix B. Revision History

1087

Rev	Date	By Whom	What
wss-saml-01	2003-10-15	Rich Levinson	Initial version
wss-saml-02	2003-10-22	Rich Levinson	Added holder-of-key use case plus minor updates to other sections.
Wss-saml-03	2003-11-7	Rich Levinson	Update w comments from Ron Monzillo, add ssl use case, remove userlist checking, use cert refs, add intro framework, clarify issuerlist plus other specific technical adjustments.
Wss-saml-04	2004-1-29	Rich Levinson	Update minor comments. Bring in synch with WS-SAML Profile (V9 27-Jan-04)

1088

1089 Appendix C. Notices

1090 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1091 that might be claimed to pertain to the implementation or use of the technology described in this
1092 document or the extent to which any license under such rights might or might not be available;
1093 neither does it represent that it has made any effort to identify any such rights. Information on
1094 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1095 website. Copies of claims of rights made available for publication and any assurances of licenses
1096 to be made available, or the result of an attempt made to obtain a general license or permission
1097 for the use of such proprietary rights by implementors or users of this specification, can be
1098 obtained from the OASIS Executive Director.

1099 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1100 applications, or other proprietary rights which may cover technology that may be required to
1101 implement this specification. Please address the information to the OASIS Executive Director.

1102 Copyright © OASIS Open 2002. *All Rights Reserved.*

1103 This document and translations of it may be copied and furnished to others, and derivative works
1104 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1105 published and distributed, in whole or in part, without restriction of any kind, provided that the
1106 above copyright notice and this paragraph are included on all such copies and derivative works.
1107 However, this document itself does not be modified in any way, such as by removing the
1108 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1109 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1110 Property Rights document must be followed, or as required to translate it into languages other
1111 than English.

1112 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1113 successors or assigns.

1114 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1115 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1116 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1117 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1118 PARTICULAR PURPOSE.

1119

1120

1121