



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

Web Services Security: Interop 2 Scenarios

Working Draft 05, 19 Sep 2003

Document identifier:

wss-interop2-draft-05.doc

Location:

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

Editor:

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

Contributors:

Chris Kaler, Microsoft <ckaler@microsoft.com>
Hal Lockhart, BEA Systems <hlockhar@bea.com>
Peter Dapkus, BEA Systems <pdapkus@bea.com>
Anthony Nadalin, IBM <drsecure@us.ibm.com>
Frederick Hirsch, nokia <Frederick.Hirsch@nokia.com>
Grant Goodale, Reactivity <grant@reactivity.com>
Symon Chang, CommerceOne <symon.chang@commerceone.com>

Abstract:

This document documents the four scenarios to be used in the second WSS 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.

Table of Contents

29	Introduction	5
30	1.1 Terminology.....	5
31	2 Test Application	6
32	3 Scenario #4 Session Key.....	7
33	3.1 Agreements	7
34	3.1.1 SESSION-KEY-VALUE	7
35	3.1.2 CERT-VALUE	7
36	3.1.3 Signature Trust Root.....	7
37	3.2 Parameters.....	7
38	3.3 General Message Flow	7
39	3.4 First Message - Request.....	8
40	3.4.1 Message Elements and Attributes	8
41	3.4.2 Message Creation.....	8
42	3.4.3 Message Processing.....	9
43	3.4.4 Example (Non-normative)	10
44	3.5 Second Message - Response	11
45	3.5.1 Message Elements and Attributes	11
46	3.5.2 Message Creation.....	12
47	3.5.3 Message Processing.....	13
48	3.5.4 Example (Non-normative)	13
49	3.6 Other processing.....	14
50	3.6.1 Requester	14
51	3.6.2 Responder	14
52	3.7 Expected Security Properties.....	14
53	4 Scenario #5 – Overlapping Signatures	15
54	4.1 Agreements	15
55	4.1.1 CERT-VALUE	15
56	4.1.2 Signature Trust Root.....	15
57	4.2 Parameters.....	15
58	4.3 General Message Flow	15
59	4.4 First Message - Request.....	15
60	4.4.1 Message Elements and Attributes	15
61	4.4.2 Message Creation.....	16
62	4.4.3 Message Processing.....	17
63	4.4.4 Example (Non-normative)	18
64	4.5 Second Message - Response	19
65	4.5.1 Message Elements and Attributes	19
66	4.5.2 Message Creation.....	19
67	4.5.3 Message Processing.....	19
68	4.5.4 Example (Non-normative)	19
69	4.6 Other processing	20
70	4.6.1 Requester	20

71	4.6.2 Responder	20
72	4.7 Expected Security Properties.....	20
73	5 Scenario #6 – Encrypt and Sign	21
74	5.1 Agreements.....	21
75	5.1.1 CERT-VALUE	21
76	5.1.2 Signature Trust Root.....	21
77	5.2 Parameters.....	21
78	5.3 General Message Flow	21
79	5.4 First Message - Request.....	21
80	5.4.1 Message Elements and Attributes	21
81	5.4.2 Message Creation.....	22
82	5.4.3 Message Processing.....	24
83	5.4.4 Example (Non-normative).....	24
84	5.5 Second Message - Response.....	25
85	5.5.1 Message Elements and Attributes	25
86	5.5.2 Message Creation.....	26
87	5.5.3 Message Processing.....	27
88	5.5.4 Example (Non-normative).....	28
89	5.6 Other processing.....	29
90	5.6.1 Requester	29
91	5.6.2 Responder	29
92	5.7 Expected Security Properties.....	29
93	6 Scenario #7 – Signed Token.....	30
94	6.1 Agreements.....	30
95	6.1.1 CERT-VALUE	30
96	6.1.2 Signature Trust Root.....	30
97	6.2 Parameters.....	30
98	6.3 General Message Flow	30
99	6.4 First Message - Request.....	31
100	6.4.1 Message Elements and Attributes	31
101	6.4.2 Message Creation.....	32
102	6.4.3 Message Processing.....	33
103	6.4.4 Example (Non-normative).....	33
104	6.5 Second Message - Response.....	35
105	6.5.1 Message Elements and Attributes	35
106	6.5.2 Message Creation.....	36
107	6.5.3 Message Processing.....	37
108	6.5.4 Example (Non-normative).....	37
109	6.6 Other processing.....	38
110	6.6.1 Requester	38
111	6.6.2 Responder	39
112	6.7 Expected Security Properties.....	39
113	7 References.....	40
114	7.1 Normative	40

115 Appendix A. Ping Application WSDL File 41
116 Appendix B. Revision History 43
117 Appendix C. Notices 44
118

119 **Introduction**

120 This document describes the four message exchanges to be tested during the second
121 interoperability event of the WSS TC. All four use the Request/Response Message Exchange
122 Pattern (MEP) with no intermediaries. All four invoke the same simple application. To avoid
123 confusion, they are called Scenario #4 through Scenario #7.

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

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

134 **1.1 Terminology**

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

137 **2 Test Application**

138 All three scenarios use the same, simple application.

139 The Requester sends a Ping element with a value of a string. The value should be the name of
140 the organization that has developed the software and the number of the scenario, e.g. "Acme
141 Corp. – Scenario #6".

142 The Responder returns a PingResponse element with a value of the same string.

143 **3 Scenario #4 Session Key**

144 The Request Body contains data that has been signed and encrypted. The certificate used to
145 verify the signature is provided in the header. The symmetric key used to perform the encryption
146 is provided out-of-band. The Response Body is also signed and encrypted. The same symmetric
147 key is used to perform the encryption. The certificate used to verify the signature is provided out-
148 of-band.

149 **3.1 Agreements**

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

152 **3.1.1 SESSION-KEY-VALUE**

153 This is an opaque identifier indicating a random symmetric key that has been previously agreed
154 by unspecified means.

155 **3.1.2 CERT-VALUE**

156 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
157 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
158 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of
159 digitalSignature.

160 **3.1.3 Signature Trust Root**

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

164 **3.2 Parameters**

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

167 No parameters are required.

168 **3.3 General Message Flow**

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

170 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
171 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
172 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
173 which is signed and then encrypted. The certificate for signing is included in the message. The
174 encryption is performed using a previously agreed session key.

175 The Responder decrypts the body and then verifies the signature. If no errors are detected it
176 returns the response signing and encrypting the message body. The response is also signed and
177 encrypted. The signing key is provided externally. The encryption is done using the same
178 previously agreed session key.

179 **3.4 First Message - Request**

180 **3.4.1 Message Elements and Attributes**

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

185

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
ReferenceList	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
Cipherdata	Mandatory

186

187 **3.4.2 Message Creation**

188 **3.4.2.1 Security**

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

190 **3.4.2.2 ReferenceList**

191 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
192 refers to the encrypted body of the message.

193 **3.4.2.3 BinarySecurityToken**

194 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
195 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
196 suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage extension. If
197 it does contain a KeyUsage extension, it SHOULD include the value of digitalSignature. The
198 Requester must have access to the private key corresponding to the public key in the certificate.

199 **3.4.2.4 Signature**

200 The signature is over the entire SOAP body.

201 **3.4.2.4.1 SignedInfo**

202 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
203 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
204 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
205 MUST be SHA1.

206 **3.4.2.4.2 SignatureValue**

207 The SignatureValue MUST be calculated as specified by the specification, using the private key
208 corresponding to the public key specified in the certificate in the BinarySecurityToken.

209 **3.4.2.4.3 KeyInfo**

210 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
211 indicates the BinarySecurityToken containing the certificate which will be used for signature
212 verification.

213 **3.4.2.5 Timestamp**

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

216 **3.4.2.6 Body**

217 The body element MUST be first signed and then its contents encrypted.

218 **3.4.2.7 EncryptedData**

219 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
220 EncryptedKey.

221 The Type MUST have the value of #Content.

222 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
223 – CBC.

224 The KeyInfo MUST contain a KeyName which is the SESSION-KEY-VALUE.

225 The CypherData MUST contain the encrypted form of the Body, encrypted under the random key
226 identified by SESSION-KEY-VALUE, using the specified algorithm.

227 **3.4.3 Message Processing**

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

231 3.4.3.1 Security

232 3.4.3.2 ReferenceList

233 The ReferenceList indicates the data to be decrypted.

234 3.4.3.3 Timestamp

235 The Timestamp element MUST be ignored.

236 3.4.3.4 Body

237 The contents of the body MUST first be decrypted and then the signature verified. If no errors are
238 detected, the body MUST be passed to the application.

239 3.4.3.5 EncryptedData

240 The message body contents contained in the EncryptedData, referenced by the ReferenceList
241 MUST be decrypted using the key identified by SESSION-KEY-VALUE, using the specified
242 algorithm.

243 3.4.3.6 BinarySecurityToken

244 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
245 authorized entity. The public key in the certificate MUST be retained for verification of the
246 signature.

247 3.4.3.7 Signature

248 The body after decryption, MUST be verified against the signature using the specified algorithms
249 and transforms and the retained public key.

250 3.4.4 Example (Non-normative)

251 Here is an example request.

```
252 <?xml version="1.0" encoding="utf-8" ?>
253 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
254 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
255 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
256 <soap:Header>
257 <wsse:Security soap:mustUnderstand="1"
258 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
259 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
260 <xenc:DataReference URI="#enc" />
261 </xenc:ReferenceList>
262 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
263 EncodingType="wsse:Base64Binary"
264 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
265 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
266 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
267 <SignedInfo>
268 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
269 />
270 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
271 <Reference URI="#body">
272 <Transforms>
273 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
274 </Transforms>
275 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
276 <DigestValue>QTV...dw=</DigestValue>
277 </Reference>
278 </SignedInfo>
279 <SignatureValue>H+x0...gUw=</SignatureValue>
```

280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306

```

<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#myCert" />
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
<wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="body"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
    cbc" />
    <xenc:KeyInfo>
      <xenc:KeyName>SessionKey</xenc:KeyName>
    </xenc:KeyInfo>
    <xenc:CipherData>
      <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</soap:Body>
</soap:Envelope>

```

307 3.5 Second Message - Response

308 3.5.1 Message Elements and Attributes

309 Items not listed in the following table MUST NOT be created or processed. Items marked
310 mandatory MUST be generated and processed. Items marked optional MAY be generated and
311 MUST be processed if present. Items MUST appear in the order specified, except as noted.
312

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
ReferenceList	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory

EncryptionMethod	Mandatory
KeyInfo	Mandatory
Cipherdata	Mandatory

313

314 **3.5.2 Message Creation**

315 **3.5.2.1 Security**

316 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
317 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

318 **3.5.2.2 ReferenceList**

319 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
320 refers to the encrypted body of the message.

321 **3.5.2.3 Signature**

322 The signature is over the entire SOAP body.

323 **3.5.2.3.1 SignedInfo**

324 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
325 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
326 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
327 MUST be SHA1.

328 **3.5.2.3.2 SignatureValue**

329 The SignatureValue MUST be calculated as specified by the specification, using the private key
330 corresponding to the public key specified by the CERT-VALUE.

331 **3.5.2.3.3 KeyInfo**

332 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
333 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
334 MUST have the value of CERT-VALUE.

335 **3.5.2.4 Timestamp**

336 The Created element within the Timestamp SHOULD contain the current local time at the sender
337 expressed in the UTC timezone.

338 **3.5.2.5 Body**

339 The body element MUST be first signed and then its contents encrypted.

340 **3.5.2.6 EncryptedData**

341 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
342 EncryptedKey.

343 The Type MUST have the value of #Content.

344 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
345 – CBC.

346 The KeyInfo MUST contain a KeyName which is the SESSION-KEY-VALUE.

347 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
348 using the specified algorithm.

349 **3.5.3 Message Processing**

350 This section describes the processing performed by the Responder. If an error is detected, the
351 Responder MUST cease processing the message and report the fault locally with a value of
352 FailedAuthentication.

353 **3.5.3.1 Security**

354 **3.5.3.2 ReferenceList**

355 The ReferenceList indicates the data to be decrypted

356 **3.5.3.3 Timestamp**

357 The Timestamp element MUST be ignored.

358 **3.5.3.4 Body**

359 The contents of the body MUST first be decrypted and then the signature verified.

360 **3.5.3.5 EncryptedData**

361 The message body contents contained in the EncryptedData, referenced by the ReferenceList
362 MUST be decrypted using the key identified by SESSION-KEY-VALUE, using the specified
363 algorithm

364 **3.5.3.6 Signature**

365 The body after decryption, MUST be verified against the signature using the specified algorithms
366 and transforms and the indicated public key.

367 **3.5.4 Example (Non-normative)**

368 Here is an example response.

```
369 <?xml version="1.0" encoding="utf-8" ?>
370 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
371 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
372 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
373 <soap:Header>
374 <wsse:Security soap:mustUnderstand="1"
375 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
376 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
377 <xenc:DataReference URI="#enc" />
378 </xenc:ReferenceList>
379 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
380 <SignedInfo>
381 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
382 />
383 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
384 <Reference URI="#body">
385 <Transforms>
386 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
387 </Transforms>
388 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
```

```
389     <DigestValue>KxW...5B=</DigestValue>
390   </Reference>
391 </SignedInfo>
392 <SignatureValue>8Hkd...al7=</SignatureValue>
393 <KeyInfo>
394   <wsse:SecurityTokenReference>
395     <wsse:KeyIdentifier
396 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
397   </wsse:SecurityTokenReference>
398 </KeyInfo>
399 </Signature>
400 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
401   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
402   </wsu:Timestamp>
403 </wsse:Security>
404 </soap:Header>
405 <soap:Body wsu:Id="body"
406 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
407   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
408     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
409     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
410 cbc" />
411     <xenc:KeyInfo>
412       <xenc:KeyName>SessionKey</xenc:KeyName>
413     </xenc:KeyInfo>
414     <xenc:CipherData>
415       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
416     </xenc:CipherData>
417   </xenc:EncryptedData>
418 </soap:Body>
419 </soap:Envelope>
```

420

421 **3.6 Other processing**

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

423 **3.6.1 Requester**

424 No additional processing is required.

425 **3.6.2 Responder**

426 No additional processing is required.

427 **3.7 Expected Security Properties**

428 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
429 of the request is protected against modification and interception. The response is Authenticated
430 and protected against modification and interception. Protection against interception in both
431 directions depends on the assumption that the session key has been previously agreed in a
432 secure fashion and that it cannot be guessed.

433 The Responder must not draw any inferences about what party encrypted the message, it
434 particular it should not be assumed it was the same party who signed it.

435 **4 Scenario #5 – Overlapping Signatures**

436 The Request Body contains data that has been signed twice. First the ticket element is signed.
437 The certificate used to verify this signature is provided out-of-band. Next the entire body is
438 signed. The certificate used to verify this signature is provided in the header. The Response Body
439 is not signed or encrypted.

440 **4.1 Agreements**

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

443 **4.1.1 CERT-VALUE**

444 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
445 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
446 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of
447 digitalSignature.

448 The Responder MUST have access to the Private key corresponding to the Public key in the
449 certificate.

450 **4.1.2 Signature Trust Root**

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

454 **4.2 Parameters**

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

457 No parameters are required.

458 **4.3 General Message Flow**

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

460 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
461 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
462 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
463 which is signed twice. First the ticket element is signed. The certificate used to verify this
464 signature is provided out-of-band. Next the entire body is signed. The certificate for this signature
465 is included in the message. The Responder verifies both signatures. If no errors are detected it
466 returns the response without any signatures.

467 **4.4 First Message - Request**

468 **4.4.1 Message Elements and Attributes**

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

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory

474

475 **4.4.2 Message Creation**

476 **4.4.2.1 Security**

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

478 **4.4.2.2 Signature**

479 This signature is over the first element of the SOAP body.

480 **4.4.2.2.1 SignedInfo**

481 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
 482 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the first element under
 483 the SOAP Body element. The only Transform specified MUST be Exclusive Canonicalization. The
 484 DigestMethod MUST be SHA1.

485 **4.4.2.2 SignatureValue**

486 The SignatureValue MUST be calculated as specified by the specification, using the private key
487 corresponding to the public key specified in the certificate identified by the KeyIdentifier CERT-
488 VALUE.

489 **4.4.2.3 KeyInfo**

490 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
491 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
492 MUST have the value of CERT-VALUE.

493 **4.4.2.3 BinarySecurityToken**

494 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
495 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
496 suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage extension. If
497 it does contain a KeyUsage extension, it SHOULD include the values of digitalSignature. The
498 Requester must have access to the private key corresponding to the public key in the certificate.

499 **4.4.2.4 Signature**

500 This signature is over the entire SOAP body.

501 **4.4.2.4.1 SignedInfo**

502 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
503 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
504 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
505 MUST be SHA1.

506 **4.4.2.4.2 SignatureValue**

507 The SignatureValue MUST be calculated as specified by the specification, using the private key
508 corresponding to the public key specified in the certificate in the BinarySecurityToken.

509 **4.4.2.4.3 KeyInfo**

510 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
511 indicates the BinarySecurityToken containing the certificate which will be used for signature
512 verification.

513 **4.4.2.5 Timestamp**

514 The Created element within the Timestamp SHOULD contain the current local time at the sender
515 expressed in the UTC time zone

516 **4.4.2.6 Body**

517 The body element MUST be signed twice. The body contains two Ping requests. The first
518 signature is over only the ticket element and the second signature is over the entire body.

519 **4.4.3 Message Processing**

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

523 4.4.3.1 Security

524 4.4.3.2 Signature

525 The certificate referred to by the KeyIdentifier MUST be validated. The Subject of the certificate
526 MUST be an authorized entity. The first element in the body MUST be verified against the
527 signature using the specified algorithms and transforms and the indicated public key.

528 4.4.3.3 BinarySecurityToken

529 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
530 authorized entity. The public key in the certificate MUST be retained for verification of the
531 signature.

532 4.4.3.4 Signature

533 The body MUST be verified against the signature using the specified algorithms and transforms
534 and the retained public key.

535 4.4.3.5 Timestamp

536 The Timestamp element MUST be ignored.

537 4.4.3.6 Body

538 After verifying both signatures, if no errors are detected, the body MUST be passed to the
539 application.

540 4.4.4 Example (Non-normative)

541 Here is an example request.

```
542 <?xml version="1.0" encoding="utf-8" ?>
543 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
544   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
545   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
546   <soap:Header>
547     <wsse:Security soap:mustUnderstand="1"
548   xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
549     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
550       <SignedInfo>
551         <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
552       />
553       <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
554       <Reference URI="#body">
555         <Transforms>
556           <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
557         </Transforms>
558         <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
559         <DigestValue>AXK...Fe=</DigestValue>
560       </Reference>
561     </SignedInfo>
562     <SignatureValue>MQwx...agv=</SignatureValue>
563     <KeyInfo>
564       <wsse:SecurityTokenReference>
565         <wsse:KeyIdentifier
566   ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
567       </wsse:SecurityTokenReference>
568     </KeyInfo>
569   </Signature>
570   <wsse:BinarySecurityToken ValueType="wsse:X509v3"
571   EncodingType="wsse:Base64Binary"
572   xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
573     wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
574   </Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
```

575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606

```
<SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <Reference URI="#tick">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>QTV...dw=</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>H+x0...gUw=</SignatureValue>
<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#myCert" />
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
<wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="body">
  <Ping xmlns="http://xmlsoap.org/Ping">
    <text>Acme Corp. - Scenario #5</text>
    <ticket wsu:Id="tick">1234567</ticket>
  </Ping>
</soap:Body>
</soap:Envelope>
```

4.5 Second Message - Response

4.5.1 Message Elements and Attributes

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

Name	Mandatory?
Body	Mandatory

4.5.2 Message Creation

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

4.5.3 Message Processing

The body is passed to the application without modification.

4.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
623 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
624 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
625 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
626 <soap:Body>
627 <PingResponse xmlns="http://xmlsoap.org/Ping">
628 <text> Acme Corp. - Scenario #5</text>
629 </PingResponse>
630 </soap:Body>
631 </soap:Envelope>
```

632 **4.6 Other processing**

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

634 **4.6.1 Requester**

635 No additional processing is required.

636 **4.6.2 Responder**

637 No additional processing is required.

638 **4.7 Expected Security Properties**

639 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
640 of the request is protected against modification. The response is not protected in any way.

641 **5 Scenario #6 – Encrypt and Sign**

642 The Request Body contains data that has been encrypted and signed. The certificate associated
643 with the encryption is provided out-of-band. The certificate used to verify the signature is provided
644 in the header. The Response Body is also encrypted and signed, reversing the roles of the key
645 pairs identified by the certificates.

646 **5.1 Agreements**

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

649 **5.1.1 CERT-VALUE**

650 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
651 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
652 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of
653 keyEncipherment, dataEncipherment and digitalSignature.

654 The Responder MUST have access to the Private key corresponding to the Public key in the
655 certificate.

656 **5.1.2 Signature Trust Root**

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

660 **5.2 Parameters**

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

663 No parameters are required.

664 **5.3 General Message Flow**

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

666 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
667 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
668 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
669 which is encrypted and then signed. The certificate for encryption is provided externally. The
670 certificate for signing is included in the message The Responder verifies the signature and then
671 decrypts the body. If no errors are detected it returns the response encrypting and signing the
672 message body. The roles of the key pairs are reversed from that of the request, using the
673 encryption key to sign and the signing key to encrypt.

674 **5.4 First Message - Request**

675 **5.4.1 Message Elements and Attributes**

676 Items not listed in the following table MAY be present, but MUST NOT be marked with the
677 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.

678 Items marked optional MAY be generated and MUST be processed if present. Items MUST
 679 appear in the order specified, except as noted.
 680

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

681

682 **5.4.2 Message Creation**

683 **5.4.2.1 Security**

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

685 **5.4.2.2 BinarySecurityToken**

686 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
 687 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate

688 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
689 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
690 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have
691 access to the private key corresponding to the public key in the certificate.

692 **5.4.2.3 Signature**

693 The signature is over the entire SOAP body.

694 **5.4.2.3.1 SignedInfo**

695 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
696 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
697 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
698 MUST be SHA1.

699 **5.4.2.3.2 SignatureValue**

700 The SignatureValue MUST be calculated as specified by the specification, using the private key
701 corresponding to the public key specified in the certificate in the BinarySecurityToken.

702 **5.4.2.3.3 KeyInfo**

703 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
704 indicates the BinarySecurityToken containing the certificate which will be used for signature
705 verification.

706 **5.4.2.4 EncryptedKey**

707 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

708 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
709 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
710 MUST have the value of CERT-VALUE.

711 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
712 Key specified in the specified X.509 certificate, using the specified algorithm.

713 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
714 refers to the encrypted body of the message.

715 **5.4.2.5 Timestamp**

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

718 **5.4.2.6 Body**

719 The contents of the body element MUST be first encrypted and then the entire element signed.

720 **5.4.2.7 EncryptedData**

721 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
722 EncryptedKey.

723 The Type MUST have the value of #Content.

724 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
725 – CBC.

726 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
727 using the specified algorithm.

728 **5.4.3 Message Processing**

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

732 **5.4.3.1 Security**

733 **5.4.3.2 BinarySecurityToken**

734 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
735 authorized entity. The public key in the certificate MUST be retained for verification of the
736 signature.

737 **5.4.3.3 Signature**

738 The body after decryption, MUST be verified against the signature using the specified algorithms
739 and transforms and the retained public key.

740 **5.4.3.4 EncryptedKey**

741 The random key contained in the CipherData MUST be decrypted using the private key
742 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

743 **5.4.3.5 Timestamp**

744 The Timestamp element MUST be ignored.

745 **5.4.3.6 Body**

746 The signature over the body MUST first be verified decrypted and then its contents decrypted. If
747 no errors are detected, the body MUST be passed to the application.

748 **5.4.3.7 EncryptedData**

749 The message body contents contained in the EncryptedData, referenced by the ReferenceList
750 MUST be decrypted using the random key, using the specified algorithm.

751 **5.4.4 Example (Non-normative)**

752 Here is an example request.

```
753 <?xml version="1.0" encoding="utf-8" ?>
754 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
755 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
756 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
757 <soap:Header>
758 <wsse:Security soap:mustUnderstand="1"
759 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
760 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
761 EncodingType="wsse:Base64Binary"
762 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
763 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
764 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
765 <SignedInfo>
766 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
767 />
768 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
769 <Reference URI="#body">
770 <Transforms>
771 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
772 </Transforms>
```



```

773     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
774     <DigestValue>QTV...dw=</DigestValue>
775     </Reference>
776 </SignedInfo>
777 <SignatureValue>H+x0...gUw=</SignatureValue>
778 <KeyInfo>
779   <wsse:SecurityTokenReference>
780     <wsse:Reference URI="#myCert" />
781   </wsse:SecurityTokenReference>
782 </KeyInfo>
783 </Signature>
784 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
785   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
786 />
787   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
788     <wsse:SecurityTokenReference>
789       <wsse:KeyIdentifier
790 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
791     </wsse:SecurityTokenReference>
792   </KeyInfo>
793   <xenc:CipherData>
794     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
795   </xenc:CipherData>
796   <xenc:ReferenceList>
797     <xenc:DataReference URI="#enc" />
798   </xenc:ReferenceList>
799 </xenc:EncryptedKey>
800 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
801   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
802 </wsu:Timestamp>
803 </wsse:Security>
804 </soap:Header>
805 <soap:Body wsu:Id="body"
806 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
807   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
808     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
809     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
810 cbc" />
811     <xenc:CipherData>
812       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
813     </xenc:CipherData>
814   </xenc:EncryptedData>
815 </soap:Body>
816 </soap:Envelope>

```

817

818 5.5 Second Message - Response

819 5.5.1 Message Elements and Attributes

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

823

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Signature	Mandatory
SignedInfo	Mandatory

CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

824

825 **5.5.2 Message Creation**

826 **5.5.2.1 Security**

827 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
828 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

829 **5.5.2.2 Signature**

830 The signature is over the entire SOAP body.

831 **5.5.2.2.1 SignedInfo**

832 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
833 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
834 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
835 MUST be SHA1.

836 **5.5.2.2.2 SignatureValue**

837 The SignatureValue MUST be calculated as specified by the specification, using the private key
838 corresponding to the public key specified in the certificate in the BinarySecurityToken.

839 **5.5.2.2.3 KeyInfo**

840 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
841 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
842 MUST have the value of CERT-VALUE.

843 **5.5.2.3 BinarySecurityToken**

844 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
845 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent
846 in the request.

847 **5.5.2.4 EncryptedKey**

848 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

849 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
850 indicates the BinarySecurityToken containing the certificate which will be used for signature
851 verification.

852 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
853 Key specified in the specified X.509 certificate, using the specified algorithm.

854 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
855 refers to the encrypted body of the message.

856 **5.5.2.5 Timestamp**

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

859 **5.5.2.6 Body**

860 The contents of the body element MUST be first encrypted and then the entire element signed.

861 **5.5.2.7 EncryptedData**

862 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
863 EncryptedKey.

864 The Type MUST have the value of #Content.

865 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
866 – CBC.

867 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
868 using the specified algorithm.

869 **5.5.3 Message Processing**

870 This section describes the processing performed by the Responder. If an error is detected, the
871 Responder MUST cease processing the message and report the fault locally with a value of
872 FailedAuthentication.

873 **5.5.3.1 Security**

874 **5.5.3.2 Timestamp**

875 The Timestamp element MUST be ignored.

876 5.5.3.3 Body

877 The contents of the body MUST first be decrypted and then the signature verified.

878 5.5.3.4 EncryptedData

879 The message body contents contained in the EncryptedData, referenced by the ReferenceList
880 MUST be decrypted using the random key, using the specified algorithm.

881 5.5.3.5 Signature

882 The body after decryption, MUST be verified against the signature using the specified algorithms
883 and transforms and the indicated public key.

884 5.5.3.6 BinarySecurityToken

885 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
886 authorized entity. The certificate is used to identify the private key to be used for decryption.

887 5.5.3.7 EncryptedKey

888 The random key contained in the CipherData MUST be decrypted using the private key
889 corresponding to the certificate specified by the Reference, using the specified algorithm.

890 5.5.4 Example (Non-normative)

891 Here is an example response.

```
892 <?xml version="1.0" encoding="utf-8" ?>
893 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
894   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
895   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
896   <soap:Header>
897     <wsse:Security soap:mustUnderstand="1"
898     xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
899       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
900         <SignedInfo>
901           <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
902           />
903           <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
904           <Reference URI="#body">
905             <Transforms>
906               <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
907             </Transforms>
908             <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
909             <DigestValue>KxW...5B=</DigestValue>
910           </Reference>
911         </SignedInfo>
912         <SignatureValue>8Hkd...a17=</SignatureValue>
913       <KeyInfo>
914         <wsse:SecurityTokenReference>
915           <wsse:KeyIdentifier
916           ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
917         </wsse:SecurityTokenReference>
918       </KeyInfo>
919     </Signature>
920     <wsse:BinarySecurityToken ValueType="wsse:X509v3"
921     EncodingType="wsse:Base64Binary"
922     xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
923     wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
924     <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmenc#">
925     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-1_5"
926     />
927     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
928     <wsse:SecurityTokenReference>
929     <wsse:Reference URI="#myCert" />
```

```
930     </wsse:SecurityTokenReference>
931 </KeyInfo>
932 <xenc:CipherData>
933   <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
934 </xenc:CipherData>
935 <xenc:ReferenceList>
936   <xenc:DataReference URI="#enc" />
937 </xenc:ReferenceList>
938 </xenc:EncryptedKey>
939 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
940   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
941 </wsu:Timestamp>
942 </wsse:Security>
943 </soap:Header>
944 <soap:Body wsu:Id="body"
945 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
946   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
947     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
948     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
949     cbc" />
950     <xenc:CipherData>
951       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
952     </xenc:CipherData>
953   </xenc:EncryptedData>
954 </soap:Body>
955 </soap:Envelope>
```

956

957 **5.6 Other processing**

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

959 **5.6.1 Requester**

960 No additional processing is required.

961 **5.6.2 Responder**

962 No additional processing is required.

963 **5.7 Expected Security Properties**

964 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
965 of the request is protected against modification and interception. The response is Authenticated
966 and protected against modification and interception. Note that the fact that the signature is over
967 the cyphertext may raise doubts as to whether the signing entity was aware what was signed.

968 The cleartext SignatureValue may also assist a known plaintext attack. The Responder must not
969 draw any inferences about what party encrypted the message, it particular it should not be
970 assumed it was the same party who signed it.

971 **6 Scenario #7 – Signed Token**

972 The Request Body contains data that has been signed and encrypted. The signature also
973 protects an enclosed Security Token by means of the STR Dereference Transform. The
974 certificate used to verify the signature is provided in the header. The certificate associated with
975 the encryption is provided out-of-band. The Response Body is also signed and encrypted,
976 reversing the roles of the key pairs identified by the certificates.

977 **6.1 Agreements**

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

980 **6.1.1 CERT-VALUE**

981 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
982 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
983 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of
984 keyEncipherment, dataEncipherment and digitalSignature.

985 The Responder MUST have access to the Private key corresponding to the Public key in the
986 certificate.

987 **6.1.2 Signature Trust Root**

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

991 **6.2 Parameters**

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

994 No parameters are required.

995 **6.3 General Message Flow**

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

997 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
998 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
999 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
1000 which is signed and then encrypted. The signature also covers the Token used for signing. The
1001 certificate for signing is included in the message. The certificate for encryption is provided
1002 externally. The Responder decrypts the body and then verifies the signature. If no errors are
1003 detected it returns the response signing and encrypting the message body. The roles of the key
1004 pairs are reversed from that of the request, using the signing key to encrypt and the encryption
1005 key to sign. The signature also covers the Token used for signing.

1006 **6.4 First Message - Request**

1007 **6.4.1 Message Elements and Attributes**

1008 Items not listed in the following table MAY be present, but MUST NOT be marked with the
1009 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
1010 Items marked optional MAY be generated and MUST be processed if present. Items MUST
1011 appear in the order specified, except as noted.

1012

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1013

1014 **6.4.2 Message Creation**

1015 **6.4.2.1 Security**

1016 The Security element MUST contain the mustUnderstand="1" attribute.

1017 **6.4.2.2 EncryptedKey**

1018 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1019 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
1020 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
1021 MUST have the value of CERT-VALUE.

1022 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
1023 Key specified in the specified X.509 certificate, using the specified algorithm.

1024 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
1025 refers to the encrypted body of the message.

1026 **6.4.2.3 BinarySecurityToken**

1027 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
1028 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
1029 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
1030 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
1031 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have
1032 access to the private key corresponding to the public key in the certificate.

1033 **6.4.2.4 Signature**

1034 The signature is over the entire SOAP body.

1035 **6.4.2.4.1 SignedInfo**

1036 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
1037 be RSA-SHA1.

1038 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference
1039 contained in the Signature. The STR Dereference Transform with a parameter of the Exclusive
1040 Canonicalization Transform MUST be specified. The DigestMethod MUST be SHA1.

1041 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The
1042 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be
1043 SHA1.

1044 **6.4.2.4.2 SignatureValue**

1045 The SignatureValue MUST be calculated as specified by the specification, using the private key
1046 corresponding to the public key specified in the certificate in the BinarySecurityToken.

1047 **6.4.2.4.3 KeyInfo**

1048 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
1049 indicates the BinarySecurityToken containing the certificate which will be used for signature
1050 verification.

1051 **6.4.2.5 Timestamp**

1052 The Created element within the Timestamp SHOULD contain the current local time at the sender
1053 expressed in the UTC time zone.

1054 **6.4.2.6 Body**

1055 The body element MUST be first signed and then its contents encrypted.

1056 **6.4.2.7 EncryptedData**

1057 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
1058 EncryptedKey.

1059 The Type MUST have the value of #Content.

1060 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
1061 – CBC.

1062 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
1063 using the specified algorithm.

1064 **6.4.3 Message Processing**

1065 This section describes the processing performed by the Responder. If an error is detected, the
1066 Responder MUST cease processing the message and issue a Fault with a value of
1067 FailedAuthentication.

1068 **6.4.3.1 Security**

1069 **6.4.3.2 EncryptedKey**

1070 The random key contained in the CipherData MUST be decrypted using the private key
1071 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

1072 **6.4.3.3 Timestamp**

1073 The Timestamp element MUST be ignored.

1074 **6.4.3.4 Body**

1075 The contents of the body MUST first be decrypted and then the signature verified. If no errors are
1076 detected, the body MUST be passed to the application.

1077 **6.4.3.5 EncryptedData**

1078 The message body contents contained in the EncryptedData, referenced by the ReferenceList
1079 MUST be decrypted using the random key, using the specified algorithm.

1080 **6.4.3.6 BinarySecurityToken**

1081 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
1082 authorized entity. The public key in the certificate MUST be retained for verification of the
1083 signature.

1084 **6.4.3.7 Signature**

1085 The body after decryption, MUST be verified against the signature using the specified algorithms
1086 and transforms and the retained public key.

1087 **6.4.4 Example (Non-normative)**

1088 Here is an example request.

1089 `<?xml version="1.0" encoding="utf-8" ?>`

```

1090 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1091 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1092 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1093 <soap:Header>
1094 <wsse:Security soap:mustUnderstand="1"
1095 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1096 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1097 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1098 />
1099 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1100 <wsse:SecurityTokenReference>
1101 <wsse:KeyIdentifier
1102 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
1103 </wsse:SecurityTokenReference>
1104 </KeyInfo>
1105 <xenc:CipherData>
1106 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
1107 </xenc:CipherData>
1108 <xenc:ReferenceList>
1109 <xenc:DataReference URI="#enc" />
1110 </xenc:ReferenceList>
1111 </xenc:EncryptedKey>
1112 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1113 EncodingType="wsse:Base64Binary"
1114 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1115 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1116 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1117 <SignedInfo>
1118 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1119 />
1120 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1121 <Reference URI="#Token">
1122 <Transforms>
1123 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
1124 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1125 c14n#" />
1126 </Transform>
1127 </Transforms>
1128 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1129 <DigestValue>pHrr...xK=</DigestValue>
1130 </Reference>
1131 <Reference URI="#body">
1132 <Transforms>
1133 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1134 </Transforms>
1135 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1136 <DigestValue>QTV...dw=</DigestValue>
1137 </Reference>
1138 </SignedInfo>
1139 <SignatureValue>H+x0...gUw=</SignatureValue>
1140 <KeyInfo>
1141 <wsse:SecurityTokenReference wsu:Id="Token">
1142 <wsse:Reference URI="#myCert" />
1143 </wsse:SecurityTokenReference>
1144 </KeyInfo>
1145 </Signature>
1146 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1147 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1148 </wsu:Timestamp>
1149 </wsse:Security>
1150 </soap:Header>
1151 <soap:Body wsu:Id="body"
1152 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1153 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1154 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1155 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
1156 cbc" />
1157 <xenc:CipherData>
1158 <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
1159 </xenc:CipherData>
1160 </xenc:EncryptedData>

```

1161
1162
1163

```
</soap:Body>  
</soap:Envelope>
```

1164

6.5 Second Message - Response

1165

6.5.1 Message Elements and Attributes

1166
1167
1168
1169

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

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1170

1171 **6.5.2 Message Creation**

1172 **6.5.2.1 Security**

1173 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
1174 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

1175 **6.5.2.2 BinarySecurityToken**

1176 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
1177 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent
1178 in the request.

1179 **6.5.2.3 EncryptedKey**

1180 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1181 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
1182 indicates the BinarySecurityToken containing the certificate which will be used for signature
1183 verification.

1184 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
1185 Key specified in the specified X.509 certificate, using the specified algorithm.

1186 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
1187 refers to the encrypted body of the message.

1188 **6.5.2.4 Signature**

1189 The signature is over the entire SOAP body.

1190 **6.5.2.4.1 SignedInfo**

1191 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
1192 be RSA-SHA1.

1193 The Reference MUST specify a relative URI that refers to the SOAP Body element. The only
1194 Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be SHA1.

1195 **6.5.2.4.2 SignatureValue**

1196 The SignatureValue MUST be calculated as specified by the specification, using the private key
1197 corresponding to the public key specified in the certificate in the BinarySecurityToken.

1198 **6.5.2.4.3 KeyInfo**

1199 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
1200 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
1201 MUST have the value of CERT-VALUE.

1202 **6.5.2.5 Timestamp**

1203 The Created element within the Timestamp SHOULD contain the current local time at the sender
1204 expressed in the UTC time zone.

1205 **6.5.2.6 Body**

1206 The body element MUST be first signed and then its contents encrypted.

1207 **6.5.2.7 EncryptedData**

- 1208 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
1209 EncryptedKey.
- 1210 The Type MUST have the value of #Content.
- 1211 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
1212 – CBC.
- 1213 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
1214 using the specified algorithm.

1215 **6.5.3 Message Processing**

- 1216 This section describes the processing performed by the Responder. If an error is detected, the
1217 Responder MUST cease processing the message and report the fault locally with a value of
1218 FailedAuthentication.

1219 **6.5.3.1 Security**

1220 **6.5.3.2 BinarySecurityToken**

- 1221 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
1222 authorized entity. The certificate is used to identify the private key to be used for decryption.

1223 **6.5.3.3 EncryptedKey**

- 1224 The random key contained in the CipherData MUST be decrypted using the private key
1225 corresponding to the certificate specified by the Reference, using the specified algorithm.

1226 **6.5.3.4 Timestamp**

- 1227 The Timestamp element MUST be ignored.

1228 **6.5.3.5 Body**

- 1229 The contents of the body MUST first be decrypted and then the signature verified.

1230 **6.5.3.6 EncryptedData**

- 1231 The message body contents contained in the EncryptedData, referenced by the ReferenceList
1232 MUST be decrypted using the random key, using the specified algorithm.

1233 **6.5.3.7 Signature**

- 1234 The body after decryption, MUST be verified against the signature using the specified algorithms
1235 and transforms and the indicated public key.

1236 **6.5.4 Example (Non-normative)**

- 1237 Here is an example response.

```
1238 <?xml version="1.0" encoding="utf-8" ?>  
1239 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
1240 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1241 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
1242 <soap:Header>  
1243 <wsse:Security soap:mustUnderstand="1"  
1244 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
```

```

1245     <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1246     EncodingType="wsse:Base64Binary"
1247     xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1248     wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1249     <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1250     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1251     />
1252     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1253     <wsse:SecurityTokenReference>
1254     <wsse:Reference URI="#myCert" />
1255     </wsse:SecurityTokenReference>
1256     </KeyInfo>
1257     <xenc:CipherData>
1258     <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
1259     </xenc:CipherData>
1260     <xenc:ReferenceList>
1261     <xenc:DataReference URI="#enc" />
1262     </xenc:ReferenceList>
1263     </xenc:EncryptedKey>
1264     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1265     <SignedInfo>
1266     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1267     />
1268     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1269     <Reference URI="#body">
1270     <Transforms>
1271     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1272     </Transforms>
1273     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1274     <DigestValue>KxW...5B</DigestValue>
1275     </Reference>
1276     </SignedInfo>
1277     <SignatureValue>8Hkd...al7</SignatureValue>
1278     <KeyInfo>
1279     <wsse:SecurityTokenReference>
1280     <wsse:KeyIdentifier
1281     ValueType="wsse:X509v3">B39R...mY</wsse:KeyIdentifier>
1282     </wsse:SecurityTokenReference>
1283     </KeyInfo>
1284     </Signature>
1285     <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1286     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1287     </wsu:Timestamp>
1288     </wsse:Security>
1289     </soap:Header>
1290     <soap:Body wsu:Id="body"
1291     xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1292     <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1293     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1294     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
1295     cbc" />
1296     <xenc:CipherData>
1297     <xenc:CipherValue>d2s...GQ</xenc:CipherValue>
1298     </xenc:CipherData>
1299     </xenc:EncryptedData>
1300     </soap:Body>
1301     </soap:Envelope>

```

1302

1303 6.6 Other processing

1304 This section describes processing that occurs outside of generating or processing a message.

1305 6.6.1 Requester

1306 No additional processing is required.

1307 **6.6.2 Responder**

1308 No additional processing is required.

1309 **6.7 Expected Security Properties**

1310 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
1311 of the request is protected against modification and interception. The response is Authenticated
1312 and protected against modification and interception. The signature over the signature token binds
1313 it to the message, preventing a repudiation attack by certificate substitution.

1314 The Responder must not draw any inferences about what party encrypted the message, it
1315 particular it should not be assumed it was the same party who signed it.

1316 **7 References**

1317 **7.1 Normative**

1318 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1319 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.

Appendix A. Ping Application WSDL File

```

1321 <definitions xmlns:tns="http://xmlsoap.org/Ping"
1322 xmlns="http://schemas.xmlsoap.org/wsdl/"
1323 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1324 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1325 xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/06/utility"
1326 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
1327   <types>
1328     <schema targetNamespace="http://xmlsoap.org/Ping"
1329     xmlns="http://www.w3.org/2001/XMLSchema">
1330       <complexType name="ticketType">
1331         <xsd:sequence>
1332           <xsd:element name="text" type="xsd:string"
1333 nillable="true"/>
1334           <xsd:element name="ticket" type="tns:ticketType"
1335 minOccurs="0"/>
1336         </xsd:sequence>
1337
1338         <xsd:attribute ref="wsu:Id" use="optional"/>
1339       </complexType>
1340       <complexType name="ping">
1341         <sequence>
1342           <element name="ticket" type="ticketType"
1343 minOccurs="0"/>
1344           <element name="text" type="xsd:string"
1345 nillable="true"/>
1346         </sequence>
1347       </complexType>
1348       <complexType name="pingResponse">
1349         <sequence>
1350           <element name="text" type="xsd:string"
1351 nillable="true"/>
1352         </sequence>
1353       </complexType>
1354       <element name="Ping" type="tns:ping"/>
1355       <element name="PingResponse" type="tns:pingResponse"/>
1356     </schema>
1357   </types>
1358   <message name="PingRequest">
1359     <part name="ping" element="tns:Ping"/>
1360   </message>
1361   <message name="PingResponse">
1362     <part name="pingResponse" element="tns:PingResponse"/>
1363   </message>
1364   <portType name="PingPort">
1365     <operation name="Ping">
1366       <input message="tns:PingRequest"/>
1367       <output message="tns:PingResponse"/>
1368     </operation>
1369   </portType>
1370   <binding name="PingBinding" type="tns:PingPort">
1371     <soap:binding style="document"
1372 transport="http://schemas.xmlsoap.org/soap/http"/>
1373     <operation name="Ping">
1374       <soap:operation/>
1375       <input>
1376         <soap:body use="literal"/>
1377       </input>

```

1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389

1390

```
        <output>  
            <soap:body use="literal"/>  
        </output>  
    </operation>  
</binding>  
<service name="PingService">  
    <port name="PingPort" binding="tns:PingBinding">  
        <soap:address  
location="http://localhost:8080/pingejb/Ping"/>  
    </port>  
    </service>  
</definitions>
```

1391

Appendix B. Revision History

1392

Rev	Date	By Whom	What
wss-01	2003-07-28	Hal Lockhart	Initial version
wss-02	2003-08-25	Hal Lockhart	<p>Timestamp is created first – Appears as last element under Security</p> <p>Made c14n method a parameter to the STR Dereference Transform in scenario 7</p> <p>Scenario 5 is altered to have a single ping element as required by the WS-I BP, a ticket element is added to Ping to provide a target for the inner signature</p>
wss-03	2003-08-26	Hal Lockhart	<p>Correct syntax of c14n parameter to STR Dereference Transform</p> <p>Change scenario 7 to sign the STR referring to the signature token rather than the encryption token</p> <p>Added ticket element to Ping schema in WSDL file</p>
wss-04	2003-08-26	Hal Lockhart	<p>Fixed Ping Schema</p> <p>Fixed various typos</p>
wss-05	2003-09-19	Hal Lockhart	<p>Put organization and scenario # in Ping string for debugging</p> <p>Fixed typos in examples</p> <p>Fixed errors in WSDL / Ping schema</p> <p>Remove STR from signature in response in scenarion #7</p>

1393

1394

Appendix C. Notices

1395 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1396 that might be claimed to pertain to the implementation or use of the technology described in this
1397 document or the extent to which any license under such rights might or might not be available;
1398 neither does it represent that it has made any effort to identify any such rights. Information on
1399 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1400 website. Copies of claims of rights made available for publication and any assurances of licenses
1401 to be made available, or the result of an attempt made to obtain a general license or permission
1402 for the use of such proprietary rights by implementors or users of this specification, can be
1403 obtained from the OASIS Executive Director.

1404 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1405 applications, or other proprietary rights which may cover technology that may be required to
1406 implement this specification. Please address the information to the OASIS Executive Director.

1407 **Copyright © OASIS Open 2002. All Rights Reserved.**

1408 This document and translations of it may be copied and furnished to others, and derivative works
1409 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1410 published and distributed, in whole or in part, without restriction of any kind, provided that the
1411 above copyright notice and this paragraph are included on all such copies and derivative works.
1412 However, this document itself does not be modified in any way, such as by removing the
1413 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1414 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1415 Property Rights document must be followed, or as required to translate it into languages other
1416 than English.

1417 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1418 successors or assigns.

1419 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1420 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1421 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1422 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1423 PARTICULAR PURPOSE.