



1

2

Web Services Security:

3

Interop 1 Scenarios

4

Working Draft 01, 17 April 2003

5

Document identifier:

6

wss-interop1-draft-01.doc

7

Location:

8

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

9

Editor:

10

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

11

Contributors:

12

Chris Kaler, Microsoft <ckaler@microsoft.com>

13

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

14

Abstract:

15

This document documents the three scenarios to be used in the first WSS Interoperability Event.

16

17

Status:

18

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.

19

20

21

22

23 Table of Contents

24	Introduction	4
25	1.1 Terminology.....	4
26	2 Test Application	5
27	3 Scenario #1	6
28	3.1 Agreements	6
29	3.2 Parameters.....	6
30	3.3 General Message Flow	6
31	3.4 First Message - Request	6
32	3.4.1 Message Elements and Attributes	6
33	3.4.2 Message Creation.....	7
34	3.4.3 Message Processing.....	7
35	3.4.4 Example (Non-normative).....	7
36	3.5 Second Message - Response	8
37	3.5.1 Message Elements and Attributes	8
38	3.5.2 Message Creation.....	8
39	3.5.3 Message Processing.....	8
40	3.5.4 Example (Non-normative).....	8
41	3.6 Other processing.....	8
42	3.6.1 Requester	8
43	3.6.2 Responder	8
44	3.7 Expected Security Properties.....	8
45	4 Scenario #2.....	9
46	4.1 Agreements	9
47	4.1.1 USERNAME-PASSWORD-LIST	9
48	4.1.2 CERT-VALUE	9
49	4.2 Parameters.....	9
50	4.2.1 MAX-CLOCK-SKEW.....	9
51	4.2.2 MAX-NONCE-AGE	9
52	4.3 General Message Flow	9
53	4.4 First Message - Request.....	9
54	4.4.1 Message Elements and Attributes	9
55	4.4.2 Message Creation.....	10
56	4.4.3 Message Processing.....	11
57	4.4.4 Example (Non-normative).....	11
58	4.5 Second Message - Response	12
59	4.5.1 Message Elements and Attributes	12
60	4.5.2 Message Creation.....	12
61	4.5.3 Message Processing.....	12
62	4.5.4 Example (Non-normative).....	13
63	4.6 Other processing.....	13
64	4.6.1 Requester	13
65	4.6.2 Responder	13

66	4.7 Expected Security Properties.....	13
67	5 Scenario #3.....	14
68	5.1 Agreements.....	14
69	5.1.1 CERT-VALUE.....	14
70	5.1.2 Signature Trust Root.....	14
71	5.2 Parameters.....	14
72	5.3 General Message Flow.....	14
73	5.4 First Message - Request.....	14
74	5.4.1 Message Elements and Attributes.....	14
75	5.4.2 Message Creation.....	15
76	5.4.3 Message Processing.....	17
77	5.4.4 Example (Non-normative).....	17
78	5.5 Second Message - Response.....	18
79	5.5.1 Message Elements and Attributes.....	18
80	5.5.2 Message Creation.....	19
81	5.5.3 Message Processing.....	20
82	5.5.4 Example (Non-normative).....	21
83	5.6 Other processing.....	22
84	5.6.1 Requester.....	22
85	5.6.2 Responder.....	22
86	5.7 Expected Security Properties.....	22
87	6 References.....	23
88	6.1 Normative.....	23
89	Appendix A. Revision History.....	24
90	Appendix B. Notices.....	25
91		

92 Introduction

93 This document describes the three message exchanges to be tested during the first
94 interoperability event of the WSS TC. All three use the Request/Response Message Exchange
95 Pattern (MEP) with no intermediaries. All three invoke the same simple application. The scenarios
96 build in complexity. Scenario #1 is the simplest and Scenario #3 is the most complex.

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

100 THESE SCENARIOS ARE NOT INTENDED TO REPRESENT REASONABLE OR USEFUL
101 PRACTICAL APPLICATIONS OF THE SPECIFICATIONS. THEY HAVE BEEN DESIGNED
102 PURELY FOR THE PURPOSES INDICATED ABOVE AND DO NOT NECESSARILY
103 REPRESENT EFFICIENT OR SECURE MEANS OF PERFORMING THE INDICATED
104 FUNCTIONS. IN PARTICULAR THESE SCENARIOS ARE KNOWN TO VIOLATE SECURITY
105 BEST PRACTICES IN SOME RESPECTS AND IN GENERAL HAVE NOT BEEN EXTENSIVELY
106 VETTED FOR ATTACKS.

107 1.1 Terminology

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

110 **2 Test Application**

111 All three scenarios use the same, simple application.

112 The Requester sends a Ping element with a value of a string.

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

114 3 Scenario #1

115 The Request header contains a Username and Password. The response does not contain a
116 security header.

117 3.1 Agreements

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

120 USERNAME-PASSWORD-LIST is a list of value pairs of usernames and their associated
121 passwords.

122 3.2 Parameters

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

125 No parameters are required.

126 3.3 General Message Flow

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

128 This contract covers a request/response MEP over the http binding. The request contains a
129 plaintext password. The receiver checks the message and issues a Fault if any errors are found.
130 Otherwise it returns the response without any security mechanisms.

131 3.4 First Message - Request

132 3.4.1 Message Elements and Attributes

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

136

Name	Mandatory?
Security	Mandatory
mustUnderstand="true"	Mandatory
UsernameToken	Mandatory
Username	Mandatory
Password	Mandatory
Body	Mandatory

137

138 **3.4.2 Message Creation**

139 **3.4.2.1 Security**

140 The Security element MUST contain the mustUnderstand="true" attribute.

141 **3.4.2.2 UsernameToken**

142 The Username and Password MUST match a username/password pair in the USERNAME-
143 PASSWORD-LIST.

144 **3.4.2.3 Body**

145 The body is not signed or encrypted in any way.

146 **3.4.3 Message Processing**

147 This section describes the processing performed by the receiver. If an error is detected, the
148 processing of this message stops and a Fault is issued.

149 **3.4.3.1 Security**

150 The presence of the Security element with mustUnderstand="true" is verified.

151 **3.4.3.2 UsernameToken**

152 The Username and Password MUST match one of the pairs in the USERNAME-PASSWORD-
153 LIST, otherwise it is an error.

154 **3.4.3.3 Body**

155 The body is passed to the application without modification.

156 **3.4.4 Example (Non-normative)**

157 Here is an example request.

```
158 <?xml version="1.0" encoding="utf-8" ?>
159 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
160 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
161 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
162   <soap:Header>
163     <wsse:Security soap:mustUnderstand="true"
164     xmlns:wsse="http://schemas.xmlsoap.org/ws/.../secext">
165       <wsse:UsernameToken>
166         <wsse:Username>Chris</wsse:Username>
167         <wsse:Password
168           Type="wsse:PasswordText">sirhC</wsse:Password>
169         </wsse:UsernameToken>
170       </wsse:Security>
171     </soap:Header>
172     <soap:Body>
173       <Ping xmlns="http://xmlsoap.org/Ping">
174         <text>EchoString</text>
175       </Ping>
176     </soap:Body>
177   </soap:Envelope>
```

178 **3.5 Second Message - Response**

179 **3.5.1 Message Elements and Attributes**

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

183

Name	Mandatory?
Body	Mandatory

184

185 **3.5.2 Message Creation**

186 The message MUST NOT contain a header.

187 **3.5.3 Message Processing**

188 The body is passed to the application without modification.

189 **3.5.4 Example (Non-normative)**

190 Here is an example response.

```
191 <?xml version="1.0" encoding="utf-8" ?>  
192 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
193 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
194 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
195   <soap:Body>  
196     <PingResponse xmlns="http://xmlsoap.org/Ping">  
197       <text>EchoString</text>  
198     </PingResponse>  
199   </soap:Body>  
200 </soap:Envelope>
```

201 **3.6 Other processing**

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

203 **3.6.1 Requester**

204 No additional processing is required.

205 **3.6.2 Responder**

206 No additional processing is required.

207 **3.7 Expected Security Properties**

208 Use of the service is restricted to parties that know how to construct a correct password value.
209 There is no protection against interception or replay of the password or of interception or
210 modification of the message body.

211

212 **4 Scenario #2**

213 The Request header contains a Username and Password that have been encrypted using a
214 public key provided out-of-band. The response does not contain a security header

215 **4.1 Agreements**

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

218 **4.1.1 USERNAME-PASSWORD-LIST**

219 This is a list of value pairs of usernames and their associated passwords.

220 **4.1.2 CERT-VALUE**

221 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
222 MUST be obtained by the Requester by unspecified means. The certificate SHOULD have a
223 KeyUsage extension that includes the value of keyEncipherment.

224 The Responder MUST have access to the Private key corresponding to the Public key in the
225 certificate.

226 **4.2 Parameters**

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

229 **4.2.1 MAX-CLOCK-SKEW**

230 This has the value of the assumed maximum skew between the local times of any two systems.

231 **4.2.2 MAX-NONCE-AGE**

232 This has the value of the length of time a previously received Nonce value will be stored.

233 **4.3 General Message Flow**

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

235 This contract covers a request/response MEP over the http binding. The request contains an
236 encrypted username token containing a plaintext password. The Responder decrypts the token
237 and checks the username and password. If no errors are detected it returns the response without
238 any security mechanisms.

239 **4.4 First Message - Request**

240 **4.4.1 Message Elements and Attributes**

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

244

Name	Mandatory?
Security	Mandatory
mustUnderstand="true"	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory
UsernameToken	Mandatory
Username	Mandatory
Password	Mandatory
Nonce	Mandatory
Created	Mandatory
Body	Mandatory

245

246 **4.4.2 Message Creation**

247 **4.4.2.1 Security**

248 The Security element MUST contain the mustUnderstand="true" attribute.

249 **4.4.2.2 EncryptedKey**

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

251 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
252 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
253 MUST have the value of CERT-VALUE.

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

256 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
257 refers to the encrypted UsernameToken.

258 **4.4.2.3 EncryptedData**

259 The Type MUST have the value of #Element.

260 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
261 – CBC.

262 The CypherData MUST contain the encrypted form of the UsernameToken, encrypted under a
263 random key, using the specified algorithm.

264 **4.4.2.4 UsernameToken**

265 The Username and Password MUST match a username/password pair in the USERNAME-
266 PASSWORD-LIST. The Nonce MUST have a value that is unique for at least a 24-hour period,
267 coded in base 64. The Created MUST have the value of the local time when the message is
268 created.

269 **4.4.2.5 Body**

270 The body is not signed or encrypted in any way.

271 **4.4.3 Message Processing**

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

275 **4.4.3.1 Security**

276 The presence of the Security element with mustUnderstand="true" is verified.

277 **4.4.3.2 EncryptedKey**

278 The random key contained in the CipherData MUST be decrypted using the Private Key
279 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

280 **4.4.3.3 EncryptedData**

281 The UsernameToken contained in the EncryptedData, referenced by the ReferenceList MUST be
282 decrypted using the random key, using the specified algorithm.

283 **4.4.3.4 UsernameToken**

284 The Username and Password MUST match one of the pairs in the USERNAME-PASSWORD-
285 LIST, otherwise it is an error. If the Nonce value matches any stored Nonce value it is an error. If
286 the Created value is older than the current local time minus MAX-NONCE-AGE plus MAX-
287 CLOCK-SKEW, it is an error.

288 If there is no error, the Nonce and Created values from the message are stored.

289 **4.4.3.5 Body**

290 The body is passed to the application without modification.

291 **4.4.4 Example (Non-normative)**

292 Here is an example of the UsernameToken before encryption.

```
293 <wsse:UsernameToken>  
294   <wsse:Username>Chris</wsse:Username>  
295   <wsse:Password  
296     Type="wsse:PasswordText">sirhC</wsse:Password>  
297   <wsse:Nonce>ykEFh55E52hCeJk5vDdUBQ==</wsse:Nonce>  
298   <wsu:Created>2003-03-18T19:50:33Z</wsu:Created>  
299 </wsse:UsernameToken>
```

300 Here is an example of the request.

```
301 <soap:Envelope xmlns:wssse="http://schemas.xmlsoap.org/ws/.../secext"  
302 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:Header>  
306 <wssse:Security soap:mustUnderstand="true"  
307 xmlns:wssse="http://schemas.xmlsoap.org/ws/.../secext">  
308 <xenc:EncryptedKey Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"  
309 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">  
310 xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />  
311 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
312 <wssse:SecurityTokenReference>  
313 <wssse:KeyIdentifier ValueType="wssse:X509v3">B39R...=</wssse:KeyIdentifier>  
314 </wssse:SecurityTokenReference>  
315 </KeyInfo>  
316 <xenc:CipherData>  
317 <xenc:CipherValue>pPzyO...XlM=</xenc:CipherValue>  
318 </xenc:CipherData>  
319 <xenc:ReferenceList>  
320 <xenc:DataReference URI="#enc-un" />  
321 </xenc:ReferenceList>  
322 </xenc:EncryptedKey>  
323 <xenc:EncryptedData Id="enc-un" Type="http://www.w3.org/2001/04/xmlenc#Element"  
324 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">  
325 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#triplede-  
326 cbc" />  
327 <xenc:CipherData>  
328 <xenc:CipherValue>A/ufDw...chA==</xenc:CipherValue>  
329 </xenc:CipherData>  
330 </xenc:EncryptedData>  
331 </wssse:Security>  
332 </soap:Header>  
333 <soap:Body>  
334 <Ping xmlns="http://xmlsoap.org/Ping">  
335 <text>EchoString</text>  
336 </Ping>  
337 </soap:Body>  
338 </soap:Envelope>
```

339 4.5 Second Message - Response

340 4.5.1 Message Elements and Attributes

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

344

Name	Mandatory?
Body	Mandatory

345

346 4.5.2 Message Creation

347 The message MUST NOT contain a header.

348 4.5.3 Message Processing

349 The body is passed to the application without modification.

350 **4.5.4 Example (Non-normative)**

351 Here is an example response.

```
352 <?xml version="1.0" encoding="utf-8" ?>
353 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
354 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
355 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
356 <soap:Body>
357 <PingResponse xmlns="http://xmlsoap.org/Ping">
358 <text>EchoString</text>
359 </PingResponse>
360 </soap:Body>
361 </soap:Envelope>
```

362 **4.6 Other processing**

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

364 **4.6.1 Requester**

365 No additional processing is required.

366 **4.6.2 Responder**

367 Periodically, stored Nonce values which are older than the current local time minus MAX-
368 NONCE-AGE plus MAX-CLOCK-SKEW MAY be discarded.

369 **4.7 Expected Security Properties**

370 Use of the service is restricted to parties that know how to construct a correct username
371 password pair. The password is protected against interception and replay. The other headers and
372 body are not protected against interception or modification. Encrypting such a short and likely to
373 be known value creates the risk of a known plaintext attack.

374

375 5 Scenario #3

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

380 5.1 Agreements

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

383 5.1.1 CERT-VALUE

384 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
385 MUST be obtained by the Requester by unspecified means. The certificate SHOULD have a
386 KeyUsage extension that includes the values of keyEncipherment and digitalSignature.

387 The Responder MUST have access to the Private key corresponding to the Public key in the
388 certificate.

389 5.1.2 Signature Trust Root

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

393 5.2 Parameters

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

396 No parameters are required.

397 5.3 General Message Flow

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

399 This contract covers a request/response MEP over the http binding. The request contains a body,
400 which is signed and then encrypted. The certificate for signing is included in the message. The
401 certificate for encryption is provided externally. The Responder decrypts the body and then
402 verifies the signature. If no errors are detected it returns the response without any security
403 mechanisms.

404 5.4 First Message - Request

405 5.4.1 Message Elements and Attributes

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

409

Name	Mandatory?
------	------------

Timestamp	Mandatory
Security	Mandatory
mustUnderstand="true"	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
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

410

411 **5.4.2 Message Creation**

412 **5.4.2.1 Timestamp**

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

414 **5.4.2.2 Security**

415 The Security element MUST contain the mustUnderstand="true" attribute.

416 **5.4.2.3 EncryptedKey**

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

418 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
419 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
420 MUST have the value of CERT-VALUE.
421 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
422 Key specified in the specified X.509 certificate, using the specified algorithm.
423 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
424 refers to the encrypted body of the message.

425 **5.4.2.4 BinarySecurityToken**

426 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
427 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
428 suitable for verifying the signature and encrypting the response. The certificate SHOULD have a
429 KeyUsage extension that includes the values of keyEncipherment and digitalSignature. The
430 Requester must have access to the private key corresponding to the public key in the certificate.

431 **5.4.2.5 Signature**

432 The signature is over the entire SOAP body.

433 **5.4.2.5.1 SignedInfo**

434 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
435 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
436 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
437 MUST be SHA1.

438 **5.4.2.5.2 SignatureValue**

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

441 **5.4.2.5.3 KeyInfo**

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

445 **5.4.2.6 Body**

446 The Body MUST be first signed and then encrypted.

447 **5.4.2.7 EncryptedData**

448 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
449 EncryptedKey.

450 The Type MUST have the value of #Element.

451 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
452 – CBC.

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

455 **5.4.3 Message Processing**

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

459 **5.4.3.1 Timestamp**

460 The Timestamp element MUST be ignored.

461 **5.4.3.2 Security**

462 The presence of the Security element with mustUnderstand="true" MUST be verified.

463 **5.4.3.3 EncryptedKey**

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

466 **5.4.3.4 Body**

467 The body MUST first be decrypted and then the signature verified. If no errors are detected, the
468 body MUST be passed to the application.

469 **5.4.3.5 EncryptedData**

470 The UsernameToken contained in the EncryptedData, referenced by the ReferenceList MUST be
471 decrypted using the random key, using the specified algorithm.

472 **5.4.3.6 BinarySecurityToken**

473 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
474 authorized entity. The public key in the certificate MUST be retained for verification of the
475 signature.

476 **5.4.3.7 Signature**

477 The Body contents after decryption MUST be verified against the signature using the specified
478 algorithms and transforms and the retained public key.

479 **5.4.4 Example (Non-normative)**

480 Here is an example request.

```
481 <?xml version="1.0" encoding="utf-8" ?>  
482 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
483 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
484 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
485 <soap:Header>  
486 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility">  
487 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>  
488 </wsu:Timestamp>  
489 <wsse:Security soap:mustUnderstand="true"  
490 xmlns:wsse="http://schemas.xmlsoap.org/ws/.../secext">  
491 <xenc:EncryptedKey Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"  
492 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">  
493 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1 5"  
494 />  
495 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
496 <wsse:SecurityTokenReference>  
497 <wsse:KeyIdentifier  
498 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
```

```

499     </wsse:SecurityTokenReference>
500 </KeyInfo>
501 <xenc:CipherData>
502   <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
503 </xenc:CipherData>
504 <xenc:ReferenceList>
505   <xenc:DataReference URI="#enc" />
506 </xenc:ReferenceList>
507 </xenc:EncryptedKey>
508 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
509 EncodingType="wsse:Base64Binary"
510 xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility"
511   wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
512 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
513   <SignedInfo>
514     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
515   />
516     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
517     <Reference URI="#body">
518       <Transforms>
519         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
520       </Transforms>
521       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
522       <DigestValue>QTV...dw=</DigestValue>
523     </Reference>
524   </SignedInfo>
525   <SignatureValue>H+x0...gUw=</SignatureValue>
526   <KeyInfo>
527     <wsse:SecurityTokenReference>
528       <wsse:Reference URI="#myCert" />
529     </wsse:SecurityTokenReference>
530   </KeyInfo>
531 </Signature>
532 </wsse:Security>
533 </soap:Header>
534 <soap:Body wsu:Id="body" xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility">
535   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Element"
536   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
537     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
538   cbc" />
539     <xenc:CipherData>
540       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
541     </xenc:CipherData>
542   </xenc:EncryptedData>
543 </soap:Body>
544 </soap:Envelope>

```

545

546 5.5 Second Message - Response

547 5.5.1 Message Elements and Attributes

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

551

Name	Mandatory?
Timestamp	Mandatory
Security	Mandatory
mustUnderstand="true"	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

552

553 **5.5.2 Message Creation**

554 **5.5.2.1 Timestamp**

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

556 **5.5.2.2 Security**

557 The Security element MUST contain the mustUnderstand="true" attribute.

558 **5.5.2.3 BinarySecurityToken**

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

562 **5.5.2.4 EncryptedKey**

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

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

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

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

571 **5.5.2.5 Signature**

572 The signature is over the entire SOAP body.

573 **5.5.2.5.1 SignedInfo**

574 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
575 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
576 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
577 MUST be SHA1.

578 **5.5.2.5.2 SignatureValue**

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

581 **5.5.2.5.3 KeyInfo**

582 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
583 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
584 MUST have the value of CERT-VALUE.

585 **5.5.2.6 Body**

586 The Body MUST be first signed and then encrypted.

587 **5.5.2.7 EncryptedData**

588 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
589 EncryptedKey.

590 The Type MUST have the value of #Element.

591 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
592 – CBC.

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

595 **5.5.3 Message Processing**

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

599 **5.5.3.1 Timestamp**

600 The Timestamp element MUST be ignored.

601 **5.5.3.2 Security**

602 The presence of the Security element with mustUnderstand="true" MUST be verified.

603 **5.5.3.3 BinarySecurityToken**

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

606 **5.5.3.4 EncryptedKey**

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

609 **5.5.3.5 Body**

610 The body MUST first be decrypted and then the signature verified.

611 **5.5.3.6 EncryptedData**

612 The UsernameToken contained in the EncryptedData, referenced by the ReferenceList MUST be
613 decrypted using the random key, using the specified algorithm.

614 **5.5.3.7 Signature**

615 The Body contents after decryption MUST be verified against the signature using the specified
616 algorithms and transforms and the indicated public key.

617 **5.5.4 Example (Non-normative)**

618 Here is an example response.

```
619 <?xml version="1.0" encoding="utf-8" ?>
620 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
621 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
622 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
623 <soap:Header>
624 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility">
625 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
626 </wsu:Timestamp>
627 <wsse:Security soap:mustUnderstand="true"
628 xmlns:wsse="http://schemas.xmlsoap.org/ws/.../secext">
629 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
630 EncodingType="wsse:Base64Binary"
631 xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility"
632 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
633 <xenc:EncryptedKey Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"
634 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
635 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
636 />
637 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
638 <wsse:SecurityTokenReference>
639 <wsse:Reference URI="#myCert" />
640 </wsse:SecurityTokenReference>
641 </KeyInfo>
642 <xenc:CipherData>
643 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
644 </xenc:CipherData>
645 <xenc:ReferenceList>
646 <xenc:DataReference URI="#enc" />
647 </xenc:ReferenceList>
648 </xenc:EncryptedKey>
649 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
650 <SignedInfo>
651 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
652 />
653 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
654 <Reference URI="#body">
655 <Transforms>
656 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
```

```
657     </Transforms>
658     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
659     <DigestValue>KxW...5B=</DigestValue>
660   </Reference>
661 </SignedInfo>
662 <SignatureValue>8Hkd...al7=</SignatureValue>
663 <KeyInfo>
664   <wsse:SecurityTokenReference>
665     <wsse:KeyIdentifier
666 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
667   </wsse:SecurityTokenReference>
668 </KeyInfo>
669 </Signature>
670 </wsse:Security>
671 </soap:Header>
672 <soap:Body wsu:Id="body" xmlns:wsu="http://schemas.xmlsoap.org/ws/.../utility">
673   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Element"
674     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
675     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
676 cbc" />
677     <xenc:CipherData>
678       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
679     </xenc:CipherData>
680   </xenc:EncryptedData>
681 </soap:Body>
682 </soap:Envelope>
```

683

684 **5.6 Other processing**

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

686 **5.6.1 Requester**

687 No additional processing is required.

688 **5.6.2 Responder**

689 No additional processing is required.

690 **5.7 Expected Security Properties**

691 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
692 of the request is protected against modification and interception. The response is Authenticated
693 and protected against modification and interception.

694 Encrypting such a short and likely to be known value creates the risk of a known plaintext attack.
695 The cleartext SignatureValue may also assist a known plaintext attack. The Responder must not
696 draw any inferences about what party encrypted the message, in particular it should not be
697 assumed it was the same party who signed it.

698 **6 References**

699 **6.1 Normative**

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

702 **Appendix A. Revision History**

703

Rev	Date	By Whom	What
wss-00	2003-04-17	Hal Lockhart	Initial version

704

705

Appendix B. Notices

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

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

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

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

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

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