



Web Services Security: Interop 1 Scenarios

Working Draft ~~065~~, ~~30-May~~6 Jun 2003

Document identifier:

wss-interop1-draft-0~~65~~.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>
Irving Reid, Baltimore <Irving.Reid@baltimore.com>
Thomas DeMartini, Contentguard <Thomas.DeMartini@contentguard.com>
Phillip H. Griffin, Griffin Consulting <phil.griffin@asn-1.com>
Peter Dapkus, BEA Systems <pdapkus@bea.com>
Jerry Schwarz, Oracle <jerry.schwarz@oracle.com>
Frederick Hirsch, Nokia <Frederick.Hirsch@nokia.com>
Eric Gravengaard, Reactivity <eric@reactivity.com>
Jan Alexander, Systinet <alex@systinet.com>
Ron Monzillo, Sun Microsystems <ronald.monzillo@sun.com>
NISHIMURA Toshihiro, Fujitsu <nishimura.toshi@jp.fujitsu.com>

Abstract:

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

33 Table of Contents

34	Introduction	4
35	1.1 Terminology.....	4
36	2 Test Application	5
37	3 Scenario #1	6
38	3.1 Agreements	6
39	3.2 Parameters.....	6
40	3.3 General Message Flow	6
41	3.4 First Message - Request	6
42	3.4.1 Message Elements and Attributes	6
43	3.4.2 Message Creation.....	7
44	3.4.3 Message Processing.....	7
45	3.4.4 Example (Non-normative)	7
46	3.5 Second Message - Response	8
47	3.5.1 Message Elements and Attributes	8
48	3.5.2 Message Creation.....	8
49	3.5.3 Message Processing.....	8
50	3.5.4 Example (Non-normative)	8
51	3.6 Other processing	8
52	3.6.1 Requester	8
53	3.6.2 Responder	8
54	3.7 Expected Security Properties.....	8
55	4 Scenario #2.....	9
56	4.1 Agreements	9
57	4.1.1 USERNAME-PASSWORD-LIST	9
58	4.1.2 CERT-VALUE	9
59	4.2 Parameters.....	9
60	4.2.1 MAX-CLOCK-SKEW.....	9
61	4.2.2 MAX-NONCE-AGE	9
62	4.3 General Message Flow	9
63	4.4 First Message - Request.....	9
64	4.4.1 Message Elements and Attributes	9
65	4.4.2 Message Creation.....	10
66	4.4.3 Message Processing.....	11
67	4.4.4 Example (Non-normative)	11
68	4.5 Second Message - Response	12
69	4.5.1 Message Elements and Attributes	12
70	4.5.2 Message Creation.....	12
71	4.5.3 Message Processing.....	13
72	4.5.4 Example (Non-normative)	13
73	4.6 Other processing	13
74	4.6.1 Requester	13
75	4.6.2 Responder	13

76	4.7 Expected Security Properties	13
77	5 Scenario #3	14
78	5.1 Agreements	14
79	5.1.1 CERT-VALUE	14
80	5.1.2 Signature Trust Root	14
81	5.2 Parameters	14
82	5.3 General Message Flow	14
83	5.4 First Message - Request	14
84	5.4.1 Message Elements and Attributes	14
85	5.4.2 Message Creation	15
86	5.4.3 Message Processing	17
87	5.4.4 Example (Non-normative)	17
88	5.5 Second Message - Response	18
89	5.5.1 Message Elements and Attributes	18
90	5.5.2 Message Creation	19
91	5.5.3 Message Processing	20
92	5.5.4 Example (Non-normative)	21
93	5.6 Other processing	22
94	5.6.1 Requester	22
95	5.6.2 Responder	22
96	5.7 Expected Security Properties	22
97	6 References	23
98	6.1 Normative	23
99	Appendix A. Ping Application WSDL File	24
100	Appendix B. Revision History	25
101	Appendix C. Notices	26
102		

103 Introduction

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

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

111 THESE SCENARIOS ARE NOT INTENDED TO REPRESENT REASONABLE OR USEFUL
112 PRACTICAL APPLICATIONS OF THE SPECIFICATIONS. THEY HAVE BEEN DESIGNED
113 PURELY FOR THE PURPOSES INDICATED ABOVE AND DO NOT NECESSARILY
114 REPRESENT EFFICIENT OR SECURE MEANS OF PERFORMING THE INDICATED
115 FUNCTIONS. IN PARTICULAR THESE SCENARIOS ARE KNOWN TO VIOLATE SECURITY
116 BEST PRACTICES IN SOME RESPECTS AND IN GENERAL HAVE NOT BEEN EXTENSIVELY
117 VETTED FOR ATTACKS.

118 1.1 Terminology

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

121 **2 Test Application**

122 All three scenarios use the same, simple application.

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

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

125 3 Scenario #1

126 The Request header contains a Username and Password. The response does not contain a
127 security header.

128 3.1 Agreements

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

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

133 3.2 Parameters

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

136 No parameters are required.

137 3.3 General Message Flow

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

139 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
140 ~~As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including~~
141 ~~a null string may be used. The recipient SHOULD ignore the value. NOT be used.~~ The request
142 contains a plaintext password. The receiver checks the message and issues a Fault if any errors
143 are found. Otherwise it returns the response without any security mechanisms.

144 3.4 First Message - Request

145 3.4.1 Message Elements and Attributes

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

150

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
UsernameToken	Mandatory
Username	Mandatory
Password	Mandatory
Body	Mandatory

151

152 **3.4.2 Message Creation**

153 **3.4.2.1 Security**

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

155 **3.4.2.2 UsernameToken**

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

158 **3.4.2.3 Body**

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

160 **3.4.3 Message Processing**

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

163 **3.4.3.1 Security**

164 **3.4.3.2 UsernameToken**

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

167 **3.4.3.3 Body**

168 The body is passed to the application without modification.

169 **3.4.4 Example (Non-normative)**

170 Here is an example request.

```
171 <?xml version="1.0" encoding="utf-8" ?>
172 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
173 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
174 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
175   <soap:Header>
176     <wsse:Security soap:mustUnderstand="1"
177 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
178       <wsse:UsernameToken>
179         <wsse:Username>Chris</wsse:Username>
180         <wsse:Password
181           Type="wsse:PasswordText">sirhC</wsse:Password>
182       </wsse:UsernameToken>
183     </wsse:Security>
184   </soap:Header>
185   <soap:Body>
186     <Ping xmlns="http://xmlsoap.org/Ping">
187       <text>EchoString</text>
188     </Ping>
189   </soap:Body>
190 </soap:Envelope>
```

191 3.5 Second Message - Response

192 3.5.1 Message Elements and Attributes

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

196

Name	Mandatory?
Body	Mandatory

197

198 3.5.2 Message Creation

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

201 3.5.3 Message Processing

202 The body is passed to the application without modification.

203 3.5.4 Example (Non-normative)

204 Here is an example response.

```
205 <?xml version="1.0" encoding="utf-8" ?>  
206 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
207 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
208 xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
209   <soap:Body>  
210     <PingResponse xmlns="http://xmlsoap.org/Ping">  
211       <text>EchoString</text>  
212     </PingResponse>  
213   </soap:Body>  
214 </soap:Envelope>
```

215 3.6 Other processing

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

217 3.6.1 Requester

218 No additional processing is required.

219 3.6.2 Responder

220 No additional processing is required.

221 3.7 Expected Security Properties

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

225

226 4 Scenario #2

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

229 4.1 Agreements

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

232 4.1.1 USERNAME-PASSWORD-LIST

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

234 4.1.2 CERT-VALUE

235 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
236 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
237 KeyUsage extension. If the KeyUsage extension is present, it SHOULD include the values of
238 keyEncipherment and dataEncipherment.

239 The Responder MUST have access to the Private key corresponding to the Public key in the
240 certificate.

241 4.2 Parameters

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

244 4.2.1 MAX-CLOCK-SKEW

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

246 4.2.2 MAX-NONCE-AGE

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

248 4.3 General Message Flow

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

250 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
251 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including
252 a null string may be used. The recipient SHOULD ignore the value. NOT be used. The request
253 contains an encrypted username token containing a plaintext password. The Responder decrypts
254 the token and checks the username and password. If no errors are detected it returns the
255 response without any security mechanisms.

256 4.4 First Message - Request

257 4.4.1 Message Elements and Attributes

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

260 Items marked optional MAY be generated and MUST be processed if present. Items MUST
 261 appear in the order specified, except as noted.
 262

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	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

263

264 **4.4.2 Message Creation**

265 **4.4.2.1 Security**

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

267 **4.4.2.2 EncryptedKey**

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

269 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
 270 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
 271 MUST have the value of CERT-VALUE.

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

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

276 **4.4.2.3 EncryptedData**

277 The Type MUST have the value of #Element.

278 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
279 – CBC.

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

282 **4.4.2.4 UsernameToken**

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

287 **4.4.2.5 Body**

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

289 **4.4.3 Message Processing**

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

293 **4.4.3.1 Security**

294 **4.4.3.2 EncryptedKey**

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

297 **4.4.3.3 EncryptedData**

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

300 **4.4.3.4 UsernameToken**

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

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

306 **4.4.3.5 Body**

307 The body is passed to the application without modification.

308 **4.4.4 Example (Non-normative)**

309 Here is an example of the UsernameToken before encryption.

```
310 <wsse:UsernameToken>  
311   <wsse:Username>Chris</wsse:Username>  
312   <wsse:Password  
313     Type="wsse:PasswordText">sirhC</wsse:Password>  
314   <wsse:Nonce>ykEFh55E52hCeJk5vDdUBQ==</wsse:Nonce>
```

```
315 <wsu:Created>2003-03-18T19:50:33Z</wsu:Created>
316 </wsse:UsernameToken>
```

317 Here is an example of the request.

```
318 <soap:Envelope xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext "
319 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
320 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
321 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
322 <soap:Header>
323 <wsse:Security soap:mustUnderstand="1"
324 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
325 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
326 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
327 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
328 <wsse:SecurityTokenReference>
329 <wsse:KeyIdentifier ValueType="wsse:X509v3">B39R...=</wsse:KeyIdentifier>
330 </wsse:SecurityTokenReference>
331 </KeyInfo>
332 <xenc:CipherData>
333 <xenc:CipherValue>pPzyO...XlM=</xenc:CipherValue>
334 </xenc:CipherData>
335 <xenc:ReferenceList>
336 <xenc:DataReference URI="#enc-un" />
337 </xenc:ReferenceList>
338 </xenc:EncryptedKey>
339 <xenc:EncryptedData wsu:Id="enc-un"
340 Type="http://www.w3.org/2001/04/xmlenc#Element "
341 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
342 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
343 cbc" />
344 <xenc:CipherData>
345 <xenc:CipherValue>A/ufDw...chA==</xenc:CipherValue>
346 </xenc:CipherData>
347 </xenc:EncryptedData>
348 </wsse:Security>
349 </soap:Header>
350 <soap:Body>
351 <Ping xmlns="http://xmlsoap.org/Ping">
352 <text>EchoString</text>
353 </Ping>
354 </soap:Body>
355 </soap:Envelope>
```

356 4.5 Second Message - Response

357 4.5.1 Message Elements and Attributes

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

361

Name	Mandatory?
Body	Mandatory

362

363 4.5.2 Message Creation

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

366 **4.5.3 Message Processing**

367 The body is passed to the application without modification.

368 **4.5.4 Example (Non-normative)**

369 Here is an example response.

```
370 <?xml version="1.0" encoding="utf-8" ?>
371 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
372 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
373 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
374 <soap:Body>
375 <PingResponse xmlns="http://xmlsoap.org/Ping">
376 <text>EchoString</text>
377 </PingResponse>
378 </soap:Body>
379 </soap:Envelope>
```

380 **4.6 Other processing**

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

382 **4.6.1 Requester**

383 No additional processing is required.

384 **4.6.2 Responder**

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

387 **4.7 Expected Security Properties**

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

392

393 5 Scenario #3

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

398 5.1 Agreements

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

401 5.1.1 CERT-VALUE

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

406 The Responder MUST have access to the Private key corresponding to the Public key in the
407 certificate.

408 5.1.2 Signature Trust Root

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

412 5.2 Parameters

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

415 No parameters are required.

416 5.3 General Message Flow

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

418 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
419 ~~As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including~~
420 ~~a null string may be used. The recipient SHOULD ignore the value. NOT be used.~~ The request
421 contains a body, which is signed and then encrypted. The certificate for signing is included in the
422 message. The certificate for encryption is provided externally. The Responder decrypts the body
423 and then verifies the signature. If no errors are detected it returns the response signing and
424 encrypting the message body. The roles of the key pairs are reversed from that of the request,
425 using the signing key to encrypt and the encryption key to sign.

426 5.4 First Message - Request

427 5.4.1 Message Elements and Attributes

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

430 Items marked optional MAY be generated and MUST be processed if present. Items MUST
 431 appear in the order specified, except as noted.
 432

Name	Mandatory?
Timestamp	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
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

433

434 **5.4.2 Message Creation**

435 **5.4.2.1 Timestamp**

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

437 **5.4.2.2 Security**

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

439 **5.4.2.3 EncryptedKey**

440 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.
441 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
442 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
443 MUST have the value of CERT-VALUE.

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

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

448 **5.4.2.4 BinarySecurityToken**

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

455 **5.4.2.5 Signature**

456 The signature is over the entire SOAP body.

457 **5.4.2.5.1 SignedInfo**

458 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
459 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
460 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
461 MUST be SHA1.

462 **5.4.2.5.2 SignatureValue**

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

465 **5.4.2.5.3 KeyInfo**

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

469 **5.4.2.6 Body**

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

471 **5.4.2.7 EncryptedData**

472 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
473 EncryptedKey.

474 The Type MUST have the value of #Content.

475 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
476 – CBC.

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

479 **5.4.3 Message Processing**

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

483 **5.4.3.1 Timestamp**

484 The Timestamp element MUST be ignored.

485 **5.4.3.2 Security**

486 **5.4.3.3 EncryptedKey**

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

489 **5.4.3.4 Body**

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

492 **5.4.3.5 EncryptedData**

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

495 **5.4.3.6 BinarySecurityToken**

496 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
497 authorized entity. The public key in the certificate MUST be retained for verification of the
498 signature.

499 **5.4.3.7 Signature**

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

502 **5.4.4 Example (Non-normative)**

503 Here is an example request.

```
504 <?xml version="1.0" encoding="utf-8" ?>
505 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
506 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
507 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
508 <soap:Header>
509 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
510 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
511 </wsu:Timestamp>
512 <wsse:Security soap:mustUnderstand="1"
513 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
514 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
515 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
516 />
517 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
518 <wsse:SecurityTokenReference>
519 <wsse:KeyIdentifier
520 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
521 </wsse:SecurityTokenReference>
522 </KeyInfo>
523 <xenc:CipherData>
```

```

524     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
525 </xenc:CipherData>
526 <xenc:ReferenceList>
527   <xenc:DataReference URI="#enc" />
528 </xenc:ReferenceList>
529 </xenc:EncryptedKey>
530 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
531 EncodingType="wsse:Base64Binary"
532 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
533   wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
534 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
535   <SignedInfo>
536     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
537   />
538     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
539     <Reference URI="#body">
540       <Transforms>
541         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
542       </Transforms>
543       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
544       <DigestValue>QTV...dw=</DigestValue>
545     </Reference>
546   </SignedInfo>
547   <SignatureValue>H+x0...gUw=</SignatureValue>
548   <KeyInfo>
549     <wsse:SecurityTokenReference>
550       <wsse:Reference URI="#myCert" />
551     </wsse:SecurityTokenReference>
552   </KeyInfo>
553 </Signature>
554 </wsse:Security>
555 </soap:Header>
556 <soap:Body wsu:Id="body"
557 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
558   <xenc:EncryptedData wsu:Id="enc"
559   Type="http://www.w3.org/2001/04/xmldsig#Content"
560   xmlns:xenc="http://www.w3.org/2001/04/xmldsig#">
561     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmldsig#tripleDES-
562     cbc" />
563     <xenc:CipherData>
564       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
565     </xenc:CipherData>
566   </xenc:EncryptedData>
567 </soap:Body>
568 </soap:Envelope>

```

569

570 5.5 Second Message - Response

571 5.5.1 Message Elements and Attributes

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

575

Name	Mandatory?
Timestamp	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

576

577 **5.5.2 Message Creation**

578 **5.5.2.1 Timestamp**

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

580 **5.5.2.2 Security**

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

583 **5.5.2.3 BinarySecurityToken**

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

587 **5.5.2.4 EncryptedKey**

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

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

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

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

596 **5.5.2.5 Signature**

597 The signature is over the entire SOAP body.

598 **5.5.2.5.1 SignedInfo**

599 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
600 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
601 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
602 MUST be SHA1.

603 **5.5.2.5.2 SignatureValue**

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

606 **5.5.2.5.3 KeyInfo**

607 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
608 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
609 MUST have the value of CERT-VALUE.

610 **5.5.2.6 Body**

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

612 **5.5.2.7 EncryptedData**

613 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
614 EncryptedKey.

615 The Type MUST have the value of #Content.

616 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
617 – CBC.

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

620 **5.5.3 Message Processing**

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

624 **5.5.3.1 Timestamp**

625 The Timestamp element MUST be ignored.

626 5.5.3.2 Security

627 5.5.3.3 BinarySecurityToken

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

630 5.5.3.4 EncryptedKey

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

633 5.5.3.5 Body

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

635 5.5.3.6 EncryptedData

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

638 5.5.3.7 Signature

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

641 5.5.4 Example (Non-normative)

642 Here is an example response.

```
643 <?xml version="1.0" encoding="utf-8" ?>
644 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
645 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
646 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
647 <soap:Header>
648 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
649 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
650 </wsu:Timestamp>
651 <wsse:Security soap:mustUnderstand="1"
652 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
653 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
654 EncodingType="wsse:Base64Binary"
655 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
656 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
657 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
658 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
659 />
660 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
661 <wsse:SecurityTokenReference>
662 <wsse:Reference URI="#myCert" />
663 </wsse:SecurityTokenReference>
664 </KeyInfo>
665 <xenc:CipherData>
666 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
667 </xenc:CipherData>
668 <xenc:ReferenceList>
669 <xenc:DataReference URI="#enc" />
670 </xenc:ReferenceList>
671 </xenc:EncryptedKey>
672 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
673 <SignedInfo>
674 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
675 />
676 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
677 <Reference URI="#body">
```

```
678     <Transforms>
679       <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
680     </Transforms>
681     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
682     <DigestValue>KxW...5B=</DigestValue>
683   </Reference>
684 </SignedInfo>
685 <SignatureValue>8Hkd...al7=</SignatureValue>
686 <KeyInfo>
687   <wsse:SecurityTokenReference>
688     <wsse:KeyIdentifier
689 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
690   </wsse:SecurityTokenReference>
691 </KeyInfo>
692 </Signature>
693 </wsse:Security>
694 </soap:Header>
695 <soap:Body wsu:Id="body"
696 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
697   <xenc:EncryptedData wsu:Id="enc"
698 Type="http://www.w3.org/2001/04/xmlenc#Content"
699   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
700     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
701 cbc" />
702     <xenc:CipherData>
703       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
704     </xenc:CipherData>
705   </xenc:EncryptedData>
706 </soap:Body>
707 </soap:Envelope>
```

708

709 5.6 Other processing

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

711 5.6.1 Requester

712 No additional processing is required.

713 5.6.2 Responder

714 No additional processing is required.

715 5.7 Expected Security Properties

716 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
717 of the request is protected against modification and interception. The response is Authenticated
718 and protected against modification and interception.

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

723 **6 References**

724 **6.1 Normative**

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

Appendix A. Ping Application WSDL File

```

728 <definitions xmlns:tns="http://xmlsoap.org/Ping"
729 xmlns="http://schemas.xmlsoap.org/wsdl/"
730 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
731 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
732 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
733   <types>
734     <schema targetNamespace="http://xmlsoap.org/Ping"
735     xmlns="http://www.w3.org/2001/XMLSchema">
736       <complexType name="ping">
737         <sequence>
738           <element name="text" type="xsd:string"
739 nillable="true"/>
740         </sequence>
741       </complexType>
742       <complexType name="pingResponse">
743         <sequence>
744           <element name="text" type="xsd:string"
745 nillable="true"/>
746         </sequence>
747       </complexType>
748       <element name="Ping" type="tns:ping"/>
749       <element name="PingResponse" type="tns:pingResponse"/>
750     </schema>
751   </types>
752   <message name="PingRequest">
753     <part name="ping" element="tns:Ping"/>
754   </message>
755   <message name="PingResponse">
756     <part name="pingResponse" element="tns:PingResponse"/>
757   </message>
758   <portType name="PingPort">
759     <operation name="Ping">
760       <input message="tns:PingRequest"/>
761       <output message="tns:PingResponse"/>
762     </operation>
763   </portType>
764   <binding name="PingBinding" type="tns:PingPort">
765     <soap:binding style="document"
766 transport="http://schemas.xmlsoap.org/soap/http"/>
767     <operation name="Ping">
768       <soap:operation/>
769       <input>
770         <soap:body use="literal"/>
771       </input>
772       <output>
773         <soap:body use="literal"/>
774       </output>
775     </operation>
776   </binding>
777   <service name="PingService">
778     <port name="PingPort" binding="tns:PingBinding">
779       <soap:address
780 location="http://localhost:8080/pingejb/Ping"/>
781     </port>
782   </service>
783 </definitions>

```


785

Appendix B. Revision History

786

Rev	Date	By Whom	What
wss-01	2003-04-17	Hal Lockhart	Initial version
wss-02	2003-04-29	Hal Lockhart	Minor changes based on comments
wss-03	2003-05-19	Hal Lockhart	More minor changes
wss-04	2003-05-23	Hal Lockhart	Fix errors in description of Scenario 3
wss-05	2003-05-30	Hal Lockhart	Fix errors related to signatures and encryption, add new Appendix containing Ping WSDL
<u>wss-06</u>	<u>2003-06-06</u>	<u>Hal Lockhart</u>	<u>Correct SOAPAction, namespace for Id</u>

787

Appendix C. Notices

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

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

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

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

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

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