



1

2

3

4

5

6

Web Services Security SOAP Messages with Attachments (SwA) Profile 1.0

Interop 1 Scenarios

Working Draft 02, 18 Oct 2004

7

Document identifier:

8

swa-interop1-draft-01.doc

9

Location:

10

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

11

Editor:

12

Blake Dournaee, Sarvega Inc. <blake@sarvega.com>

13

Contributors:

14

Bruce Rich, IBM <brich@us.ibm.com>

15

Abstract:

16

This document formalizes the interoperability scenarios to be used in the first Web Services Security SwA Profile interoperability event.

17

18

Status:

19

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.

20

21

22

23

24 Table of Contents

25	Introduction	5
26	1.1 Terminology.....	5
27	2 Test Application	6
28	2.1 Example Ping Element.....	6
29	2.2 Example PingResponse Element.....	6
30	2.3 SOAP Message Packages.....	6
31	3 Scenario #1: Attachment Signature	7
32	3.1 Attachment Properties.....	7
33	3.2 Agreements	7
34	3.2.1 CERT-VALUE	7
35	3.2.2 Signature Trust Root.....	7
36	3.3 Parameters.....	7
37	3.4 General Message Flow	7
38	3.5 First Message – Request	8
39	3.5.1 Message Elements and Attributes	8
40	3.5.2 Message Creation.....	8
41	3.5.3 Responder Message Processing.....	9
42	3.5.4 Example (Non-normative).....	10
43	3.6 Second Message - Response	10
44	3.6.1 Message Elements and Attributes	10
45	3.6.2 Message Creation.....	11
46	3.6.3 Message Processing.....	11
47	3.6.4 Example (Non-normative).....	11
48	3.7 Other processing	11
49	3.7.1 Requester	11
50	3.7.2 Responder	11
51	3.8 Expected Security Properties.....	11
52	4 Scenario #2 – Attachment Encryption	12
53	4.1 Attachment Properties.....	12
54	4.2 Agreements	12
55	4.2.1 CERT-VALUE	12
56	4.2.2 Signature Trust Root.....	12
57	4.3 Parameters.....	12
58	4.4 General Message Flow	12
59	4.5 First Message - Request.....	13
60	4.5.1 Message Elements and Attributes	13
61	4.5.2 Message Creation.....	13
62	4.5.3 Responder Message Processing.....	15
63	4.5.4 Example (Non-normative).....	15
64	4.6 Second Message - Response	16
65	4.6.1 Message Elements and Attributes	16
66	4.6.2 Message Creation.....	16

67	4.6.3 Message Processing.....	17
68	4.6.4 Example (Non-normative).....	17
69	4.7 Other processing.....	17
70	4.7.1 Requester	17
71	4.7.2 Responder	17
72	4.8 Expected Security Properties.....	17
73	5 Scenario #3 – Attachment Signature and Encryption.....	18
74	5.1 Attachment Properties.....	18
75	5.2 Agreements.....	18
76	5.2.1 CERT-VALUE	18
77	5.2.2 Signature Trust Root.....	18
78	5.3 Parameters.....	18
79	5.4 General Message Flow	19
80	5.5 First Message - Request.....	19
81	5.5.1 Message Elements and Attributes	19
82	5.5.2 Message Creation.....	20
83	5.5.3 Responder Message Processing.....	22
84	5.5.4 Example (Non-normative).....	22
85	5.6 Second Message - Response	24
86	5.6.1 Message Elements and Attributes	24
87	5.6.2 Message Creation.....	24
88	5.6.3 Message Processing.....	24
89	5.6.4 Example (Non-normative).....	24
90	5.7 Other processing.....	25
91	5.7.1 Requester	25
92	5.7.2 Responder	25
93	5.8 Expected Security Properties.....	25
94	6 Scenario #4 – Attachment Signature and Encryption with MIME Headers	26
95	6.1 Attachment Properties.....	26
96	6.2 Agreements.....	26
97	6.2.1 CERT-VALUE	26
98	6.2.2 Signature Trust Root.....	26
99	6.3 Parameters.....	26
100	6.4 General Message Flow	27
101	6.5 First Message - Request.....	27
102	6.5.1 Message Elements and Attributes	27
103	6.5.2 Message Creation.....	28
104	6.5.3 Responder Message Processing.....	30
105	6.5.4 Example (Non-normative).....	30
106	6.6 Second Message - Response	32
107	6.6.1 Message Elements and Attributes	32
108	6.6.2 Message Creation.....	32
109	6.6.3 Message Processing.....	32
110	6.6.4 Example (Non-normative).....	33

111	6.7 Other processing.....	33
112	6.7.1 Requester	33
113	6.7.2 Responder	33
114	6.8 Expected Security Properties.....	33
115	7 References.....	34
116	7.1 Normative	34
117	Appendix A. Ping Application WSDL File.....	35
118	Appendix B. Revision History	37
119	Appendix C. Notices	38
120		

121 Introduction

122 This document describes the message exchanges to be tested during the first interoperability
123 event of the Web Services Security SOAP Message with Attachments Profile. All scenarios use
124 the Request/Response Message Exchange Pattern (MEP) with no intermediaries. All scenarios
125 invoke the same simple application. To avoid confusion, they are called Scenario #1 through
126 Scenario #4.

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

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

137 1.1 Terminology

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

140 2 Test Application

141 All four scenarios use the same, simple application.

142 The Requester sends a Ping element with a value of a string as the single child to a SOAP
143 request. The value should be the name of the organization that has developed the software and
144 the number of the scenario, e.g. "Acme Corp. – Scenario #1".

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

146 Each interaction will also include a SOAP attachment secured via one of the content level
147 security mechanisms described in **[WSS-SwA]**. For the purpose of these interoperability
148 scenarios, the Ping request and response elements will not have security properties applied to
149 them; they are used only to keep track of the specific scenarios.

150 2.1 Example Ping Element

```
151 <Ping xmlns="http://xmlsoap.org/Ping">  
152   <text>Acme Corp. - Scenario #1</text>  
153 </Ping>
```

154 2.2 Example PingResponse Element

```
155 <PingResponse xmlns="http://xmlsoap.org/Ping">  
156   <text> Acme Corp. - Scenario #1</text>  
157 </PingResponse>
```

158 2.3 SOAP Message Packages

159 When SOAP attachments are used as specified in **[SwA]** the main SOAP payload is
160 accompanied by a MIME header and possibly multiple boundary parts. This is known as a SOAP
161 message package. All interoperability scenarios in this document assume that a proper SOAP
162 message package is constructed using the MIME headers appropriate to **[SwA]**. Interoperability
163 of the SOAP message package format, including the appropriate use of the MIME header and
164 boundary semantics, is outside the scope of this interoperability document.

165 **3 Scenario #1: Attachment Signature**

166 Scenario #1 tests the interoperability of a signed attachment using an X.509 certificate. The
167 certificate used to verify the signature shall be present in the SOAP header. No security
168 properties are applied to any part of the SOAP envelope..

169 **3.1 Attachment Properties**

170 This section specifies the attachment properties BEFORE security operations are applied. The
171 Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be
172 base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment.
173 The generation of the Content-Id header is out of scope.

174 **3.2 Agreements**

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

177 **3.2.1 CERT-VALUE**

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

182 **3.2.2 Signature Trust Root**

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

186 **3.3 Parameters**

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

189 No parameters are required.

190 **3.4 General Message Flow**

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

192 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
193 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**. As
194 required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a
195 null string may be used. The recipient SHOULD ignore the value. The request contains a signed
196 attachment. The certificate used for signing is included in the message.

197 The Responder verifies the signature over the attachment. If no errors are detected it returns the
198 response with no additional security properties.

199 **3.5 First Message – Request**

200 **3.5.1 Message Elements and Attributes**

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

205

Name	Mandatory?
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

206 **3.5.2 Message Creation**

207 **3.5.2.1 Security**

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

209 **3.5.2.2 BinarySecurityToken**

210 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
211 be labeled with an Id so it can be referenced by the signature. The value MUST be a public key
212 certificate suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage
213 extension. If it does contain a KeyUsage extension, it SHOULD include the value of
214 digitalSignature. The Requester must have access to the private key corresponding to the public
215 key in the certificate.

216 **3.5.2.3 Signature**

217 The signature is over the attachment content only, using the #Attachment-Content-Only-
218 Transform

219 **3.5.2.3.1 SignedInfo**

220 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
221 be RSA-SHA1.

222 **3.5.2.3.2 Reference**

223 The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the
224 attachment. The only Transform specified MUST be #Attachment-Content-Only. The
225 DigestMethod MUST be SHA1.

226 **3.5.2.3.3 SignatureValue**

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

229 **3.5.2.3.4 KeyInfo**

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

233 **3.5.2.4 Post Operation Attachment Properties**

234 This section specifies the attachment properties AFTER security operations are applied. The
235 Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be
236 base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment.
237 The generation of the Content-Id header is out of scope.

238 **3.5.3 Responder Message Processing**

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

242 **3.5.3.1 Security**

243 **3.5.3.2 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 **3.5.3.3 Signature**

248 The attachment MUST be verified against the signature using the specified algorithms and
249 transforms and the retained public key.

250 **3.5.3.4 Attachment**

251 After the attachment's signature has been verified, it should be passed to the application.

252 3.5.4 Example (Non-normative)

```
253 Content-Type: multipart/related; boundary="sig-example"; type="text/xml"
254 --sig-example
255 Content-Type: text/xml; charset=utf-8
256
257 <?xml version="1.0" encoding="utf-8" ?>
258 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
259   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
260   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
261   <soap:Header>
262     <wsse:Security soap:mustUnderstand="1"
263       xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
264       secext-1.0.xsd">
265
266       <!-- This is the certificate used to verify the signature -->
267       <wsse:BinarySecurityToken ValueType="wsse:X509v3"
268         EncodingType="wsse:Base64Binary" xmlns:wsu="http://docs.oasis
269         open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
270         wsu:Id="mySigCert">MII...hk</wsse:BinarySecurityToken>
271
272       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
273         <SignedInfo>
274           <CanonicalizationMethod
275             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
276           <SignatureMethod
277             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
278           <Reference URI="cid:signature">
279             <Transforms>
280               <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
281               2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform"/>
282             </Transforms>
283             <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
284             <DigestValue>QTV...dw</DigestValue>
285           </Reference>
286         </SignedInfo>
287         <SignatureValue>H+x0...gUw</SignatureValue>
288         <KeyInfo>
289           <wsse:SecurityTokenReference>
290             <wsse:Reference URI="#mySigCert" />
291           </wsse:SecurityTokenReference>
292         </KeyInfo>
293       </Signature>
294     </wsse:Security>
295   </soap:Header>
296   <soap:Body>
297     <Ping xmlns="http://xmlsoap.org/Ping">
298       <text>Acme Corp. - Scenario #1</text>
299     </Ping>
300   </soap:Body>
301 </soap:Envelope>
302
303 --sig-example
304 Content-Type: image/jpeg
305 Content-Id: <signature>
306 Content-Transfer-Encoding: base64
307
308 Dcg3AdGFcFs3764fddSArk
```

309

310 3.6 Second Message - Response

311 3.6.1 Message Elements and Attributes

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

315

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

316

317 **3.6.2 Message Creation**

318 **3.6.2.1 Security**

319 There are no security properties on the response message

320 **3.6.2.2 Body**

321 The body element MUST be not be signed or encrypted

322 **3.6.3 Message Processing**

323 The response is passed to the application without modification.

324 **3.6.4 Example (Non-normative)**

325 Here is an example response.

```
326 <?xml version="1.0" encoding="utf-8" ?>
327 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
328   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
329   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
330   <soap:Body>
331     <PingResponse xmlns="http://xmlsoap.org/Ping">
332       <text> Acme Corp. - Scenario #1</text>
333     </PingResponse>
334   </soap:Body>
335 </soap:Envelope>
```

336

337 **3.7 Other processing**

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

339 **3.7.1 Requester**

340 No additional processing is required.

341 **3.7.2 Responder**

342 No additional processing is required.

343 **3.8 Expected Security Properties**

344 Use of the service is restricted to authorized parties that sign the attachment. The attachment of
345 the request is protected against modification and interception. The response does not have any
346 security properties.

347 **4 Scenario #2 – Attachment Encryption**

348 The SOAP request has an attachment that has been encrypted. The encryption is done using a
349 symmetric cipher. The symmetric encryption key is further encrypted for a specific recipient
350 identified by an X.509 certificate. The certificate associated with the key encryption is provided to
351 the requestor out-of-band. No security properties are applied to any part of the SOAP envelope.

352 **4.1 Attachment Properties**

353 This section specifies the attachment properties BEFORE security operations are applied. The
354 Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be
355 base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment.
356 The generation of the Content-Id header is out of scope.

357 **4.2 Agreements**

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

360 **4.2.1 CERT-VALUE**

361 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
362 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
363 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of
364 keyEncipherment.

365 The Responder MUST have access to the Private key corresponding to the Public key in the
366 certificate.

367 **4.2.2 Signature Trust Root**

368 There is no digital signature operation for this scenario

369 **4.3 Parameters**

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

372 No parameters are required.

373 **4.4 General Message Flow**

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

375 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
376 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**.

377 The Content-Transfer-Encoding for the encrypted attachment MUST be base64. As required by
378 SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string
379 may be used. The recipient SHOULD ignore the value. The request contains an encrypted SOAP
380 attachment. The attachment is encrypted with a random symmetric key, which is encrypted using
381 a public key certificate. The certificate used for the encryption is provided to the Requestor out of
382 band. The Responder decrypts the attachment using the symmetric key which is decrypted with
383 the matching private key. If no errors are detected it returns the response without any security
384 properties.

385 **4.5 First Message - Request**

386 **4.5.1 Message Elements and Attributes**

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

391

Name	Mandatory?
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory
CipherReference	Mandatory
Transforms	Mandatory
Transform	Mandatory
Transform	Mandatory
Body	Mandatory
Ping	Mandatory

392 **4.5.2 Message Creation**

393 **4.5.2.1 Security**

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

395 **4.5.2.2 BinarySecurityToken**

396 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
397 be labeled with an Id so it can be referenced by the signature. The value MUST be a Public Key
398 certificate suitable for symmetric key encryption. The certificate SHOULD NOT have a KeyUsage
399 extension. If it does contain a KeyUsage extension, it SHOULD include the values of
400 keyEncipherment and dataEncipherment. The Responder must have access to the private key
401 corresponding to the public key in the certificate.

402 **4.5.2.3 EncryptedKey**

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

404 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
405 indicates the BinarySecurityToken containing the certificate which will be used to decrypt the
406 symmetric key.

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

409 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
410 refers to the EncryptedData element that refers to the encrypted attachment.

411 **4.5.2.4 EncryptedData**

412 The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be
413 present and it MUST have a value of #Attachment-Content-Only. The EncryptedData element
414 MUST be referenced by the ReferenceList element in the EncryptedKey element. The
415 EncryptedData MUST have a MimeType attribute with the value of image/jpeg.

416 **4.5.2.5 EncryptionMethod**

417 The encryption method MUST be Triple-DES in CBC mode.

418 **4.5.2.6 CipherData**

419 The CipherData MUST refer to the encrypted attachment with a CipherReference element. The
420 CipherReference element MUST refer to the attachment using a URI with a cid scheme. The
421 CipherReference must have a Transforms child with two Transform child elements. The first
422 transform must have an Algorithm value of #Attachment-Content-Only-Transform and the second
423 transform MUST have an Algorithm value of <http://www.w3.org/2000/09/xmldsig#base64>.

424

425 **4.5.2.7 Body**

426 The body element MUST not have any security operations applied to it.

427 **4.5.2.8 Ping**

428 The Ping element should contain the scenario number and the name of the entity performing the
429 request.

430 **4.5.2.9 Post Operation Attachment Properties**

431 This section specifies the attachment properties AFTER security operations are applied. The
432 Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-
433 Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely
434 identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id
435 MUST match the Content-Id before encryption.

436 4.5.3 Responder Message Processing

437 This section describes the processing performed by the Responder. If an error is detected, the
438 Responder MUST cease processing the message and issue a Fault with a value of
439 FailedDecryption.

440 4.5.3.1 Security

441 4.5.3.2 BinarySecurityToken

442 The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation
443 of the public key is required. The responder MUST have the matching private key.

444 4.5.3.3 EncryptedKey

445 The random key contained in the CipherData MUST be decrypted using the private key
446 corresponding to the certificate specified by the SecurityTokenReference, using the specified
447 algorithm.

448 4.5.3.4 EncryptedData

449 The attachment referred to by the EncryptedData MUST be decrypted using the encrypted
450 symmetric key.

451 4.5.3.5 Attachment

452 After decrypting the attachment, it should be passed to the application

453 4.5.4 Example (Non-normative)

454 Here is an example request.

```
455 Content-Type: multipart/related; boundary="enc-example"; type="text/xml"
456 --enc-example
457 Content-Type: text/xml; charset=utf-8
458
459 <?xml version="1.0" encoding="utf-8" ?>
460 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
461 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
462 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
463 <soap:Header>
464 <wsse:Security soap:mustUnderstand="1"
465 xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
466 secext-1.0.xsd">
467
468 <!-- This certificate is used for symmetric key encryption -->
469 <wsse:BinarySecurityToken
470 Value="MIIE...hk"
471 EncodingType="wsse:Base64Binary"
472 xmlns:wsu="http://docs.oasis
473 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
474 wsu:Id="myEncCert">MIIE...hk</wsse:BinarySecurityToken>
475
476 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
477 <xenc:EncryptionMethod
478 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
479 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
480 <wsse:SecurityTokenReference>
481 <wsse:Reference URI="#myEncCert" />
482 </wsse:SecurityTokenReference>
483 </KeyInfo>
484 <xenc:CipherData>
485 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
486 </xenc:CipherData>
487 <xenc:ReferenceList>
```

```

488     <xenc:DataReference URI="#encrypted-attachment" />
489   </xenc:ReferenceList>
490 </xenc:EncryptedKey>
491
492   <!-- The EncryptedData portion here refers to content of the attachment -->
493
494   <xenc:EncryptedData wsu:Id="encrypted-attachment"
495     Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
496 1.0#Attachment-Content-Only" MimeType="image/jpeg">
497     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
498     <xenc:EncryptionMethod
499       Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc" />
500     <xenc:CipherData>
501       <xenc:CipherReference URI="cid:enc">
502         <xenc:Transforms>
503           <ds:Transform
504             Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
505 profile-1.0#Attachment-Content-Only-Transform" />
506           <ds:Transform
507             Algorithm="http://www.w3.org/2000/09/xmldsig#base64" />
508           </xenc:Transforms>
509         </xenc:CipherReference>
510       </xenc:CipherData>
511     </xenc:EncryptedData>
512
513   </wsse:Security>
514 </soap:Header>
515 <soap:Body>
516   <Ping xmlns="http://xmlsoap.org/Ping">
517     <text>Acme Corp. - Scenario #2</text>
518   </Ping>
519 </soap:Body>
520 </soap:Envelope>
521 --enc-example
522 Content-Type: application/octet-stream
523 Content-Id: <enc>
524 Content-Transfer-Encoding: base64
525
526 Dsh5SA3thsRh3Dh54wafDhjaq2

```

527

4.6 Second Message - Response

528

4.6.1 Message Elements and Attributes

529

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

533

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

534

4.6.2 Message Creation

535

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

538 **4.6.2.1 Security**

539 There are no security properties on the response message

540 **4.6.2.2 Body**

541 The body element MUST be not be signed or encrypted

542 **4.6.3 Message Processing**

543 The response is passed to the application without modification.

544 **4.6.4 Example (Non-normative)**

545 Here is an example response.

```
546 <?xml version="1.0" encoding="utf-8" ?>
547 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
548 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
549 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
550 <soap:Body>
551 <PingResponse xmlns="http://xmlsoap.org/Ping">
552 <text> Acme Corp. - Scenario #2</text>
553 </PingResponse>
554 </soap:Body>
555 </soap:Envelope>
```

556 **4.7 Other processing**

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

558 **4.7.1 Requester**

559 No additional processing is required.

560 **4.7.2 Responder**

561 No additional processing is required.

562 **4.8 Expected Security Properties**

563 The attachment content is private for the holder of the appropriate private key. There should be
564 no inferences made regarding the authenticity of the sender. The response is not protected in any
565 way.

566 **5 Scenario #3 – Attachment Signature and**
567 **Encryption**

568 The SOAP request contains an attachment that has been signed and then encrypted. The
569 certificate associated with the encryption is provided out-of-band to the requestor. The certificate
570 used to verify the signature is provided in the header. The Response Body is not signed or
571 encrypted. There are two certificates in the request message. One identifies the recipient of the
572 encrypted attachment and one identifies the signer.

573 **5.1 Attachment Properties**

574 This section specifies the attachment properties BEFORE security operations are applied. The
575 Content-Type of the attachment MUST be text/xml. The Content-Transfer-Encoding MUST be
576 8bit ASCII (8bit). The attachment MUST have a Content-Id header that uniquely identifies the
577 attachment. The generation of the Content-Id header is out of scope. An example of what the
578 attachment may look like before encryption and signing is shown as follows. This example is non-
579 normative.

```
580 --enc-sig-example  
581 Content-Type: text/xml; charset=utf-8  
582 Content-Transfer-Encoding: 8bit  
583 Content-ID: <encsignexample>  
584  
585 <?xml version=1.0" encoding="utf-8"?>  
586 <somexml/>
```

587

588 **5.2 Agreements**

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

591 **5.2.1 CERT-VALUE**

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

596 The Responder MUST have access to the private key corresponding to the public key in the
597 certificate.

598 **5.2.2 Signature Trust Root**

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

602 **5.3 Parameters**

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

605 No parameters are required.

606 5.4 General Message Flow

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

608 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.

609 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by [SwA].

610 The Content-Transfer-Encoding for the encrypted attachment MUST be base64. As required by

611 SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string

612 may be used. The recipient SHOULD ignore the value. The request contains an attachment,

613 which is signed and then encrypted. The certificate for encryption is provided externally to the

614 requestor but conveyed in the request message. The attachment is encrypted with a random

615 symmetric key that is encrypted with a public key certificate. The certificate for signing is included

616 in the message. The Responder decrypts the attachment using its private key and then verifies

617 the signature using the included public key certificate. If no errors are detected it returns the

618 Response with no security properties.

619 5.5 First Message - Request

620 5.5.1 Message Elements and Attributes

621 Items not listed in the following table MAY be present, but MUST NOT be marked with the

622 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.

623 Items marked optional MAY be generated and MUST be processed if present. Items MUST

624 appear in the order specified, except as noted.

625

Name	Mandatory?
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory
CipherReference	Mandatory
Transforms	Mandatory

Transform	Mandatory
Transform	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

626 **5.5.2 Message Creation**

627 **5.5.2.1 Security**

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

629 **5.5.2.2 BinarySecurityToken**

630 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
631 be labeled with an Id so it can be uniquely referenced. The value MUST be a PK certificate
632 suitable for encrypting the content. The certificate SHOULD NOT have a KeyUsage extension. If
633 it does contain a KeyUsage extension, it SHOULD include the value of keyEncipherment and
634 dataEncipherment.

635 **5.5.2.3 EncryptedKey**

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

637 The KeyInfo MUST contain a SecurityTokenReference with a Reference child that points to the
638 X.509 certificate of the recipient. The Reference child should point to a relative URI which
639 indicates the BinarySecurityToken containing the certificate which will be used to decrypt the
640 symmetric key.

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

643 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
644 refers to the EncryptedData element that refers to the encrypted attachment.

645 **5.5.2.4 EncryptedData**

646 The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be
647 present and it MUST have a value of #Attachment-Content-Only. The EncryptedData element

648 MUST be referenced by the ReferenceList element in the EncryptedKey element. The
649 EncryptedData MUST have a MimeType attribute with the value of text/xml.

650 **5.5.2.5 EncryptionMethod**

651 The encryption method MUST be Triple-DES in CBC mode.

652 **5.5.2.6 CipherData**

653 The CipherData MUST refer to the encrypted attachment with a CipherReference element. The
654 CipherReference element MUST refer to the attachment using a URI with a cid scheme. The
655 CipherReference must have a Transforms child with a single Transform sub child with the value
656 of #Attachment-Content-Only-Transform.

657 **5.5.2.7 BinarySecurityToken**

658 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
659 be labeled with an Id so it can be referenced by the signature. The value MUST be a PK
660 certificate suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage
661 extension. If it does contain a KeyUsage extension, it SHOULD include the values of
662 digitalSignature. The Requester must have access to the private key corresponding to the public
663 key in the certificate.

664 **5.5.2.8 Signature**

665 The signature is over the attachment content only, using the #Attachment-Content-Only-
666 Transform

667 **5.5.2.8.1 SignedInfo**

668 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
669 be RSA-SHA1.

670 **5.5.2.8.2 Reference**

671 The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the
672 attachment. The only Transform specified MUST be #Attachment-Content-Only. The
673 DigestMethod MUST be SHA1.

674 **5.5.2.8.3 SignatureValue**

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

677 **5.5.2.8.4 KeyInfo**

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

681 **5.5.2.9 Body**

682 The contents of the body MUST not be encrypted or signed

683 **5.5.2.10 Post Operation Attachment Properties**

684 This section specifies the attachment properties AFTER security operations are applied. The
685 Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-

686 Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely
687 identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id
688 MUST match the Content-Id before encryption.

689 **5.5.3 Responder Message Processing**

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

693 **5.5.3.1 Security**

694 **5.5.3.2 BinarySecurityToken**

695 The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation
696 of the public key is required. The responder MUST have the matching private key.

697 **5.5.3.3 EncryptedKey**

698 The random key contained in the CipherData MUST be decrypted using the private key
699 corresponding to the certificate specified by the SecurityTokenReference, using the specified
700 algorithm.

701 **5.5.3.4 EncryptedData**

702 The attachment referred to by the EncryptedData MUST be decrypted using the encrypted
703 symmetric key.

704 **5.5.3.5 Attachment**

705 After decrypting the attachment, it should have its signature verified

706 **5.5.3.6 BinarySecurityToken**

707 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
708 authorized entity. The public key in the certificate MUST be retained for verification of the
709 signature.

710 **5.5.3.7 Signature**

711 The attachment MUST be verified against the signature using the specified algorithms and
712 transforms and the retained public key.

713 **5.5.3.8 Attachment**

714 After the attachment's signature has been verified, it should be passed to the application

715 **5.5.4 Example (Non-normative)**

716 Here is an example request.

```
717 Content-Type: multipart/related; boundary="enc-sig-example"; type="text/xml"
718 --enc-sig-example
719 Content-Type: text/xml; charset=utf-8
720
721 <?xml version="1.0" encoding="utf-8" ?>
722 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
723 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
724 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
725 <soap:Header>
```

```

726 <wsse:Security soap:mustUnderstand="1"
727   xmlns:wss="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
728   secext-1.0.xsd">
729
730   <!-- This certificate is used for symmetric key encryption -->
731   <wsse:BinarySecurityToken
732     ValueType="wsse:X509v3"
733     EncodingType="wsse:Base64Binary"
734     xmlns:wsu="http://docs.oasis
735     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
736     wsu:Id="myEncCert">MII...hk</wsse:BinarySecurityToken>
737
738   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
739     <xenc:EncryptionMethod
740       Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
741     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
742       <wsse:SecurityTokenReference>
743         <wsse:Reference URI="#myEncCert" />
744       </wsse:SecurityTokenReference>
745     </KeyInfo>
746     <xenc:CipherData>
747       <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
748     </xenc:CipherData>
749     <xenc:ReferenceList>
750       <xenc:DataReference URI="#encrypted-signed-attachment" />
751     </xenc:ReferenceList>
752   </xenc:EncryptedKey>
753
754   <!-- The EncryptedData portion here refers to content of the attachment -->
755
756   <xenc:EncryptedData wsu:Id="encrypted-signed-attachment"
757     Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
758     1.0#Attachment-Content-Only" MimeType="text/xml">
759     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
760     <xenc:EncryptionMethod
761       Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
762     <xenc:CipherData>
763       <xenc:CipherReference URI="cid:encsignexample">
764         <xenc:Transforms>
765           <ds:Transform
766             Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
767             profile-1.0#Attachment-Content-Only-Transform" />
768           <ds:Transform
769             Algorithm="http://www.w3.org/2000/09/xmldsig#base64" />
770         </xenc:Transforms>
771       </xenc:CipherReference>
772     </xenc:CipherData>
773   </xenc:EncryptedData>
774
775   <!-- This certificate is used to verify the signature -->
776   <wsse:BinarySecurityToken ValueType="wsse:X509v3"
777     EncodingType="wsse:Base64Binary" xmlns:wsu="http://docs.oasis
778     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
779     wsu:Id="mySigCert">MII...hk</wsse:BinarySecurityToken>
780
781   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
782     <SignedInfo>
783       <CanonicalizationMethod
784         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
785       <SignatureMethod
786         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
787       <Reference URI="cid:encsignexample">
788         <Transforms>
789           <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
790           2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform" />
791         </Transforms>
792         <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
793         <DigestValue>QTV...dw</DigestValue>
794       </Reference>
795     </SignedInfo>
796     <SignatureValue>H+x0...gUw</SignatureValue>

```

```

797     <KeyInfo>
798       <wsse:SecurityTokenReference>
799         <wsse:Reference URI="#mySigCert" />
800       </wsse:SecurityTokenReference>
801     </KeyInfo>
802   </Signature>
803 </wsse:Security>
804 </soap:Header>
805 <soap:Body>
806   <Ping xmlns="http://xmlsoap.org/Ping">
807     <text>Acme Corp. - Scenario #3</text>
808   </Ping>
809 </soap:Body>
810 </soap:Envelope>
811 --enc-sig-example
812 Content-Type: application/octet-stream
813 Content-Id: <encsignexample>
814 Content-Transfer-Encoding: base64
815 FEWMMIIIfc93ASjfdjsa358sa98xsjcx
816

```

817

818 5.6 Second Message - Response

819 5.6.1 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?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

824

825 5.6.2 Message Creation

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

828 5.6.2.1 Security

829 There are no security properties on the response message

830 5.6.2.2 Body

831 The body element MUST be not be signed or encrypted

832 5.6.3 Message Processing

833 The response is passed to the application without modification.

834 5.6.4 Example (Non-normative)

835 Here is an example response.


```
836 <?xml version="1.0" encoding="utf-8" ?>
837 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
838 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
839 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
840 <soap:Body>
841 <PingResponse xmlns="http://xmlsoap.org/Ping">
842 <text> Acme Corp. - Scenario #3</text>
843 </PingResponse>
844 </soap:Body>
845 </soap:Envelope>
```

846

847 **5.7 Other processing**

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

849 **5.7.1 Requester**

850 No additional processing is required.

851 **5.7.2 Responder**

852 No additional processing is required.

853 **5.8 Expected Security Properties**

854 Use of the service is restricted to authorized parties that sign the attachment. The request
855 attachment is protected against modification and interception. The response is not protected in
856 any way.

857 **6 Scenario #4 – Attachment Signature and**
858 **Encryption with MIME Headers**

859 The SOAP request contains an attachment that has been signed and then encrypted. The
860 certificate associated with the encryption is provided out-of-band to the requestor. The certificate
861 used to verify the signature is provided in the header. The Response Body is not signed or
862 encrypted. There are two certificates in the request message. One identifies the recipient of the
863 encrypted attachment and one identifies the signer. This scenario contrasts the first three
864 scenarios in that it covers MIME headers in the signature and encryption. This means that it uses
865 the Attachment-Complete Reference Transform and Attachment-Complete EncryptedData Type.
866 Aside from these two changes, this scenario is identical to Scenario #3.

867 **6.1 Attachment Properties**

868 This section specifies the attachment properties BEFORE security operations are applied. The
869 Content-Type of the attachment MUST be text/xml. The Content-Transfer-Encoding MUST be
870 8bit ASCII (8bit). The attachment MUST have a Content-Id header that uniquely identifies the
871 attachment. The generation of the Content-Id header is out of scope. An example of what the
872 attachment may look like before encryption and signature is shown as follows:

```
873 --enc-sig-headers-example  
874 Content-Type: text/xml; charset=UTF-8  
875 Content-Transfer-Encoding: 8bit  
876 Content-ID: <enc-sig-headers-example>  
877  
878 <?xml version=1.0" encoding="utf-8"?>  
879 <somexml/>
```

880 **6.2 Agreements**

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

883 **6.2.1 CERT-VALUE**

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

888 The Responder MUST have access to the private key corresponding to the public key in the
889 certificate.

890 **6.2.2 Signature Trust Root**

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

894 **6.3 Parameters**

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

897 No parameters are required.

898 6.4 General Message Flow

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

900 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.

901 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by [SwA].

902 The Content-Transfer-Encoding for the encrypted attachment MUST be base64. As required by

903 SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string

904 may be used. The recipient SHOULD ignore the value. The request contains an attachment,

905 which is signed and then encrypted. The certificate for encryption is provided externally to the

906 requestor but conveyed in the request message. The attachment is encrypted with a random

907 symmetric key that is encrypted with a public key certificate. The certificate for signing is included

908 in the message. The Responder decrypts the attachment using its private key and then verifies

909 the signature using the included public key certificate. If no errors are detected it returns the

910 Response with no security properties.

911 6.5 First Message - Request

912 6.5.1 Message Elements and Attributes

913 Items not listed in the following table MAY be present, but MUST NOT be marked with the

914 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.

915 Items marked optional MAY be generated and MUST be processed if present. Items MUST

916 appear in the order specified, except as noted.

917

Name	Mandatory?
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory
CipherReference	Mandatory
Transforms	Mandatory

Transform	Mandatory
Transform	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

918 **6.5.2 Message Creation**

919 **6.5.2.1 Security**

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

921 **6.5.2.2 BinarySecurityToken**

922 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
923 be labeled with an Id so it can be uniquely referenced. The value MUST be a PK certificate
924 suitable for encrypting the content. The certificate SHOULD NOT have a KeyUsage extension. If
925 it does contain a KeyUsage extension, it SHOULD include the value of keyEncipherment and
926 dataEncipherment.

927 **6.5.2.3 EncryptedKey**

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

929 The KeyInfo MUST contain a SecurityTokenReference with a Reference child that points to the
930 X.509 certificate of the recipient. The Reference child should point to a relative URI which
931 indicates the BinarySecurityToken containing the certificate which will be used to decrypt the
932 symmetric key.

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

935 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
936 refers to the EncryptedData element that refers to the encrypted attachment.

937 **6.5.2.4 EncryptedData**

938 The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be
939 present and it MUST have a value of #Attachment-Complete. The EncryptedData element MUST

940 be referenced by the ReferenceList element in the EncryptedKey element. The EncryptedData
941 MUST have a MimeType attribute with the value of text/xml.

942 **6.5.2.5 EncryptionMethod**

943 The encryption method MUST be Triple-DES in CBC mode.

944 **6.5.2.6 CipherData**

945 The CipherData MUST refer to the encrypted attachment with a CipherReference element. The
946 CipherReference element MUST refer to the attachment using a URI with a cid scheme. The
947 CipherReference must have a Transforms child with a single Transform sub child with the value
948 of #Attachment-Content-Only-Transform.

949 **6.5.2.7 BinarySecurityToken**

950 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
951 be labeled with an Id so it can be referenced by the signature. The value MUST be a PK
952 certificate suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage
953 extension. If it does contain a KeyUsage extension, it SHOULD include the values of
954 digitalSignature. The Requester must have access to the private key corresponding to the public
955 key in the certificate.

956 **6.5.2.8 Signature**

957 The signature is over the attachment content only, using the #Attachment-Content-Only-
958 Transform

959 **6.5.2.8.1 SignedInfo**

960 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
961 be RSA-SHA1.

962 **6.5.2.8.2 Reference**

963 The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the
964 attachment. The only Transform specified MUST be #Attachment-Content-Only. The
965 DigestMethod MUST be SHA1.

966 **6.5.2.8.3 SignatureValue**

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

969 **6.5.2.8.4 KeyInfo**

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

973 **6.5.2.9 Body**

974 The contents of the body MUST not be encrypted or signed

975 **6.5.2.10 Post Operation Attachment Properties**

976 This section specifies the attachment properties AFTER security operations are applied. The
977 Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-

978 Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely
979 identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id
980 MUST match the Content-Id before encryption.

981 **6.5.3 Responder Message Processing**

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

985 **6.5.3.1 Security**

986 **6.5.3.2 BinarySecurityToken**

987 The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation
988 of the public key is required. The responder MUST have the matching private key.

989 **6.5.3.3 EncryptedKey**

990 The random key contained in the CipherData MUST be decrypted using the private key
991 corresponding to the certificate specified by the SecurityTokenReference, using the specified
992 algorithm.

993 **6.5.3.4 EncryptedData**

994 The attachment referred to by the EncryptedData MUST be decrypted using the encrypted
995 symmetric key.

996 **6.5.3.5 Attachment**

997 After decrypting the attachment, it should have its signature verified

998 **6.5.3.6 BinarySecurityToken**

999 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
1000 authorized entity. The public key in the certificate MUST be retained for verification of the
1001 signature.

1002 **6.5.3.7 Signature**

1003 The attachment MUST be verified against the signature using the specified algorithms and
1004 transforms and the retained public key.

1005 **6.5.3.8 Attachment**

1006 After the attachment's signature has been verified, it should be passed to the application

1007 **6.5.4 Example (Non-normative)**

1008 Here is an example request.

```
1009 Content-Type: multipart/related; boundary="enc-sig-headers-example";  
1010 type="text/xml"  
1011 --enc-sig-headers-example  
1012 Content-Type: text/xml; charset=utf-8  
1013  
1014 <?xml version="1.0" encoding="utf-8" ?>  
1015 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
1016 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1017 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088

```
<soap:Header>
  <wsse:Security soap:mustUnderstand="1"
    xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
    secext-1.0.xsd">
    <!-- This certificate is used for symmetric key encryption -->
    <wsse:BinarySecurityToken
      ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary"
      xmlns:wsu="http://docs.oasis
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="myEncCert">MII...hk</wsse:BinarySecurityToken>

    <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#myEncCert" />
        </wsse:SecurityTokenReference>
      </KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#encrypted-signed-attachment" />
      </xenc:ReferenceList>
    </xenc:EncryptedKey>

    <!-- The EncryptedData portion here refers to content of the attachment -->

    <xenc:EncryptedData wsu:Id="encrypted-signed-attachment-headers"
      Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
      1.0#Attachment-Complete" MimeType="text/xml">
      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
      <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc" />
      <xenc:CipherData>
        <xenc:CipherReference URI="cid:encsign-headers-example">
          <xenc:Transforms>
            <ds:Transform
              Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
              profile-1.0#Attachment-Content-Only-Transform" />
            <ds:Transform
              Algorithm="http://www.w3.org/2000/09/xmldsig#base64" />
          </xenc:Transforms>
        </xenc:CipherReference>
      </xenc:CipherData>
    </xenc:EncryptedData>

    <!-- This certificate is used to verify the signature -->
    <wsse:BinarySecurityToken ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary" xmlns:wsu="http://docs.oasis
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="mySigCert">MII...hk</wsse:BinarySecurityToken>

    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <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="cid:encsign-headers-example">
          <Transforms>
            <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
            2004XX-wss-swa-profile-1.0#Attachment-Complete-Transform" />
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <DigestValue>QTV...dw</DigestValue>
        </Reference>
      </SignedInfo>
```

1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110

```
<SignatureValue>H+x0...gUw=</SignatureValue>
<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#mySigCert" />
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
</wsse:Security>
</soap:Header>
<soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
    <text>Acme Corp. - Scenario #4</text>
  </Ping>
</soap:Body>
</soap:Envelope>
--enc-sig-headers-example
Content-Type: application/octet-stream
Content-Id: <encsign-headers-example>
Content-Transfer-Encoding: base64
MW4dsa59fdsaSDr5hjdsKxhMW4dsa59ffds
```

6.6 Second Message - Response

6.6.1 Message Elements and Attributes

1113 Items not listed in the following table MUST NOT be created or processed. Items marked
1114 mandatory MUST be generated and processed. Items marked optional MAY be generated and
1115 MUST be processed if present. Items MUST appear in the order specified, except as noted.
1116

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

1117

6.6.2 Message Creation

1119 The response message MUST NOT contain a <wsse:Security> header. Any other header
1120 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

6.6.2.1 Security

1122 There are no security properties on the response message

6.6.2.2 Body

1124 The body element MUST be not be signed or encrypted

6.6.3 Message Processing

1126 The response is passed to the application without modification.

1127 **6.6.4 Example (Non-normative)**

1128 Here is an example response.

```
1129 <?xml version="1.0" encoding="utf-8" ?>
1130 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1131 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1132 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1133 <soap:Body>
1134 <PingResponse xmlns="http://xmlsoap.org/Ping">
1135 <text> Acme Corp. - Scenario #4</text>
1136 </PingResponse>
1137 </soap:Body>
1138 </soap:Envelope>
```

1139

1140 **6.7 Other processing**

1141 This section describes processing that occurs outside of generating or processing a message.

1142 **6.7.1 Requester**

1143 No additional processing is required.

1144 **6.7.2 Responder**

1145 No additional processing is required.

1146 **6.8 Expected Security Properties**

1147 Use of the service is restricted to authorized parties that sign the attachment. The request
1148 attachment is protected against modification and interception. The response is not protected in
1149 any way.

1150

1151 **7 References**

1152 **7.1 Normative**

- 1153 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1154 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1155 **[SwA]** W3C Note, "SOAP Messages with Attachments", 11 December 2000,
1156 <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-2001211>.
- 1157 **[WSS-SwA]** Hirsch, Frederick, *Web Services Security SOAP Message with Attachments*
1158 Profile 1.0, OASIS Draft 8 2004

Appendix A. Ping Application WSDL File

```

1160 <definitions xmlns:tns="http://xmlsoap.org/Ping" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1161 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1162 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
1163   <types>
1164     <schema targetNamespace="http://xmlsoap.org/Ping" xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
1165       <import namespace="http://schemas.xmlsoap.org/ws/2003/06/utility" schemaLocation="utility.xsd"/>
1166       <!--
1167       <complexType name="ticketType">
1168         <sequence>
1169           <element name="ticket" type="xsd:string"/>
1170         </sequence>
1171         <attribute ref="wsu:Id"/>
1172       </complexType>
1173       -->
1174       <complexType name="ticketType">
1175         <xsd:simpleContent>
1176           <xsd:extension base="xsd:string">
1177             <xsd:attribute ref="wsu:Id"/>
1178           </xsd:extension>
1179         </xsd:simpleContent>
1180       </complexType>
1181       <element name="ticket" type="tns:ticketType"/>
1182       <element name="text" type="xsd:string" nillable="true"/>
1183       <complexType name="ping">
1184         <sequence>
1185           <element ref="tns:text"/>
1186           <element ref="tns:ticket" minOccurs="0"/>
1187         </sequence>
1188       </complexType>
1189       <complexType name="pingResponse">
1190         <sequence>
1191           <element ref="tns:text"/>
1192         </sequence>
1193       </complexType>
1194       <element name="Ping" type="tns:ping"/>
1195       <element name="PingResponse" type="tns:pingResponse"/>
1196     </schema>
1197   </types>
1198   <message name="PingRequest">
1199     <part name="ping" element="tns:Ping"/>
1200   </message>
1201   <message name="PingResponse">
1202     <part name="pingResponse" element="tns:PingResponse"/>
1203   </message>
1204   <portType name="PingPort">
1205     <operation name="Ping">
1206       <input message="tns:PingRequest"/>
1207       <output message="tns:PingResponse"/>
1208     </operation>
1209   </portType>
1210   <binding name="PingBinding" type="tns:PingPort">
1211     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
1212     <operation name="Ping">
1213       <soap:operation/>
1214       <input>
1215         <soap:body use="literal"/>
1216       </input>
1217       <output>
1218         <soap:body use="literal"/>
1219       </output>
1220     </operation>
1221   </binding>
1222   <service name="PingService">
1223     <port name="Ping1" binding="tns:PingBinding">
1224       <soap:address location="http://localhost:9080/pingservice/Ping1"/>
1225     </port>
1226     <port name="Ping2" binding="tns:PingBinding">
1227       <soap:address location="http://localhost:9080/pingservice/Ping2"/>
1228     </port>
1229     <port name="Ping3" binding="tns:PingBinding">
1230       <soap:address location="http://localhost:9080/pingservice/Ping3"/>
1231     </port>
1232     <port name="Ping4" binding="tns:PingBinding">
1233       <soap:address location="http://localhost:9080/pingservice/Ping4"/>
1234     </port>
1235     <port name="Ping5" binding="tns:PingBinding">
1236       <soap:address location="http://localhost:9080/pingservice/Ping5"/>
1237     </port>
1238     <port name="Ping6" binding="tns:PingBinding">

```

```
1239     <soap:address location="http://localhost:9080/pingservice/Ping6"/>
1240   </port>
1241   <port name="Ping7" binding="tns:PingBinding">
1242     <soap:address location="http://localhost:9080/pingservice/Ping7"/>
1243   </port>
1244 </service>
1245 </definitions>
1246
1247
```

1248

Appendix B. Revision History

1249

Rev	Date	By Whom	What
01	2004-09-07	Blake Dournaee	Initial version

1250

1251

Appendix C. Notices

1252 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1253 that might be claimed to pertain to the implementation or use of the technology described in this
1254 document or the extent to which any license under such rights might or might not be available;
1255 neither does it represent that it has made any effort to identify any such rights. Information on
1256 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1257 website. Copies of claims of rights made available for publication and any assurances of licenses
1258 to be made available, or the result of an attempt made to obtain a general license or permission
1259 for the use of such proprietary rights by implementors or users of this specification, can be
1260 obtained from the OASIS Executive Director.

1261 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1262 applications, or other proprietary rights which may cover technology that may be required to
1263 implement this specification. Please address the information to the OASIS Executive Director.

1264 **Copyright © OASIS Open 2004. All Rights Reserved.**

1265 This document and translations of it may be copied and furnished to others, and derivative works
1266 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1267 published and distributed, in whole or in part, without restriction of any kind, provided that the
1268 above copyright notice and this paragraph are included on all such copies and derivative works.
1269 However, this document itself does not be modified in any way, such as by removing the
1270 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1271 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1272 Property Rights document must be followed, or as required to translate it into languages other
1273 than English.

1274 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1275 successors or assigns.

1276 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1277 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1278 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1279 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1280 PARTICULAR PURPOSE.