



1

2

3

4

5

6

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

Interop 1 Scenarios

Working Draft 01, 7 Sept 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

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	4
26	1.1 Terminology.....	4
27	2 Test Application	5
28	2.1 Example Ping Element.....	5
29	2.2 Example PingResponse Element.....	5
30	2.3 SOAP Message Packages.....	5
31	3 Scenario #1: Attachment Signature	6
32	3.1 Attachment Properties.....	6
33	3.2 Agreements	6
34	3.2.1 CERT-VALUE	6
35	3.2.2 Signature Trust Root.....	6
36	3.3 Parameters.....	6
37	3.4 General Message Flow	6
38	3.5 First Message – Request	7
39	3.5.1 Message Elements and Attributes	7
40	3.5.2 Message Creation.....	7
41	3.5.3 Responder Message Processing.....	8
42	3.5.4 Example (Non-normative).....	9
43	3.6 Second Message - Response	9
44	3.6.1 Message Elements and Attributes	9
45	3.6.2 Message Creation.....	10
46	3.6.3 Message Processing.....	10
47	3.6.4 Example (Non-normative).....	10
48	3.7 Other processing	10
49	3.7.1 Requester	10
50	3.7.2 Responder	10
51	3.8 Expected Security Properties.....	10
52	4 Scenario #2 – Attachment Encryption	11
53	4.1 Attachment Properties.....	11
54	4.2 Agreements	11
55	4.2.1 CERT-VALUE	11
56	4.2.2 Signature Trust Root.....	11
57	4.3 Parameters.....	11
58	4.4 General Message Flow	11
59	4.5 First Message - Request.....	12
60	4.5.1 Message Elements and Attributes	12
61	4.5.2 Message Creation.....	12
62	4.5.3 Responder Message Processing.....	13
63	4.5.4 Example (Non-normative).....	14
64	4.6 Second Message - Response	15
65	4.6.1 Message Elements and Attributes	15
66	4.6.2 Message Creation.....	15

67	4.6.3 Message Processing.....	16
68	4.6.4 Example (Non-normative).....	16
69	4.7 Other processing.....	16
70	4.7.1 Requester	16
71	4.7.2 Responder	16
72	4.8 Expected Security Properties.....	16
73	5 Scenario #3 – Attachment Signature and Encryption.....	17
74	5.1 Attachment Properties.....	17
75	5.2 Agreements.....	17
76	5.2.1 CERT-VALUE	17
77	5.2.2 Signature Trust Root.....	17
78	5.3 Parameters.....	17
79	5.4 General Message Flow	17
80	5.5 First Message - Request.....	18
81	5.5.1 Message Elements and Attributes	18
82	5.5.2 Message Creation.....	19
83	5.5.3 Responder Message Processing.....	20
84	5.5.4 Example (Non-normative).....	21
85	5.6 Second Message - Response	23
86	5.6.1 Message Elements and Attributes	23
87	5.6.2 Message Creation.....	23
88	5.6.3 Message Processing.....	23
89	5.6.4 Example (Non-normative).....	23
90	5.7 Other processing.....	24
91	5.7.1 Requester	24
92	5.7.2 Responder	24
93	5.8 Expected Security Properties.....	24
94	6 References.....	25
95	6.1 Normative	25
96	Appendix A. Ping Application WSDL File.....	26
97	Appendix B. Revision History	28
98	Appendix C. Notices	29
99		

100 Introduction

101 This document describes the message exchanges to be tested during the first interoperability
102 event of the Web Services Security SOAP Message with Attachments Profile. All scenarios use
103 the Request/Response Message Exchange Pattern (MEP) with no intermediaries. All scenarios
104 invoke the same simple application. To avoid confusion, they are called Scenario #1 through
105 Scenario #3.

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

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

116 1.1 Terminology

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

119 2 Test Application

120 All three scenarios use the same, simple application.

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

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

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

129 2.1 Example Ping Element

```
130 <Ping xmlns="http://xmlsoap.org/Ping">  
131   <text>Acme Corp. - Scenario #1</text>  
132 </Ping>
```

133 2.2 Example PingResponse Element

```
134 <PingResponse xmlns="http://xmlsoap.org/Ping">  
135   <text> Acme Corp. - Scenario #1</text>  
136 </PingResponse>
```

137 2.3 SOAP Message Packages

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

144 **3 Scenario #1: Attachment Signature**

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

148 **3.1 Attachment Properties**

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

153 **3.2 Agreements**

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

156 **3.2.1 CERT-VALUE**

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

161 **3.2.2 Signature Trust Root**

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

165 **3.3 Parameters**

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

168 No parameters are required.

169 **3.4 General Message Flow**

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

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

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

178 3.5 First Message – Request

179 3.5.1 Message Elements and Attributes

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

184

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

185 3.5.2 Message Creation

186 3.5.2.1 Security

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

188 3.5.2.2 BinarySecurityToken

189 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
190 be labeled with an Id so it can be referenced by the signature. The value MUST be a public key
191 certificate suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage
192 extension. If it does contain a KeyUsage extension, it SHOULD include the value of
193 digitalSignature. The Requester must have access to the private key corresponding to the public
194 key in the certificate.

195 **3.5.2.3 Signature**

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

198 **3.5.2.3.1 SignedInfo**

199 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
200 be RSA-SHA1.

201 **3.5.2.3.2 Reference**

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

205 **3.5.2.3.3 SignatureValue**

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

208 **3.5.2.3.4 KeyInfo**

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

212 **3.5.2.4 Post Operation Attachment Properties**

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

217 **3.5.3 Responder Message Processing**

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

221 **3.5.3.1 Security**

222 **3.5.3.2 BinarySecurityToken**

223 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
224 authorized entity. The public key in the certificate MUST be retained for verification of the
225 signature.

226 **3.5.3.3 Signature**

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

229 **3.5.3.4 Attachment**

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

231 3.5.4 Example (Non-normative)

```
232 Content-Type: multipart/related; boundary="sig-example" type=text/xml
233 --sig-example
234 Content-Type: text/xml
235
236 <?xml version="1.0" encoding="utf-8" ?>
237 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
238   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
239   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
240   <soap:Header>
241     <wsse:Security soap:mustUnderstand="1"
242       xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
243       secext-1.0.xsd">
244
245       <!-- This is the certificate used to verify the signature -->
246       <wsse:BinarySecurityToken ValueType="wsse:X509v3"
247         EncodingType="wsse:Base64Binary" xmlns:wsu="http://docs.oasis
248         open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
249         wsu:Id="mySigCert">MII...hk</wsse:BinarySecurityToken>
250
251       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
252         <SignedInfo>
253           <CanonicalizationMethod
254             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
255           <SignatureMethod
256             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
257           <Reference URI="cid:signature">
258             <Transforms>
259               <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
260               2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform"/>
261             </Transforms>
262             <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
263             <DigestValue>QTV...dw</DigestValue>
264           </Reference>
265         </SignedInfo>
266         <SignatureValue>H+x0...gUw</SignatureValue>
267         <KeyInfo>
268           <wsse:SecurityTokenReference>
269             <wsse:Reference URI="#mySigCert" />
270           </wsse:SecurityTokenReference>
271         </KeyInfo>
272       </Signature>
273     </wsse:Security>
274   </soap:Header>
275   <soap:Body wsu:Id="body">
276     <Ping xmlns="http://xmlsoap.org/Ping">
277       <text>Acme Corp. - Scenario #1</text>
278     </Ping>
279   </soap:Body>
280 </soap:Envelope>
281
282 --sig-example
283 Content-Type: image/jpeg
284 Content-Id: <signature>
285 Content-Transfer-Encoding: base64
286
287 Dcg3AdGFcFs3764fddSArk
```

288

289 3.6 Second Message - Response

290 3.6.1 Message Elements and Attributes

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

294

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

295

296 **3.6.2 Message Creation**

297 **3.6.2.1 Security**

298 There are no security properties on the response message

299 **3.6.2.2 Body**

300 The body element MUST be not be signed or encrypted

301 **3.6.3 Message Processing**

302 The response is passed to the application without modification.

303 **3.6.4 Example (Non-normative)**

304 Here is an example response.

```
305 <?xml version="1.0" encoding="utf-8" ?>
306 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
307   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
308   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
309   <soap:Body>
310     <PingResponse xmlns="http://xmlsoap.org/Ping">
311       <text> Acme Corp. - Scenario #1</text>
312     </PingResponse>
313   </soap:Body>
314 </soap:Envelope>
```

315

316 **3.7 Other processing**

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

318 **3.7.1 Requester**

319 No additional processing is required.

320 **3.7.2 Responder**

321 No additional processing is required.

322 **3.8 Expected Security Properties**

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

326 4 Scenario #2 – Attachment Encryption

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

331 4.1 Attachment Properties

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

336 4.2 Agreements

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

339 4.2.1 CERT-VALUE

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

344 The Responder MUST have access to the Private key corresponding to the Public key in the
345 certificate.

346 4.2.2 Signature Trust Root

347 There is no digital signature operation for this scenario

348 4.3 Parameters

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

351 No parameters are required.

352 4.4 General Message Flow

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

354 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
355 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**.
356 The Content-Transfer-Encoding for the encrypted attachment MUST be base64. As required by
357 SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string
358 may be used. The recipient SHOULD ignore the value. The request contains an encrypted SOAP
359 attachment. The attachment is encrypted with a random symmetric key, which is encrypted using
360 a public key certificate. The certificate used for the encryption is provided to the Requestor out of
361 band. The Responder decrypts the attachment using the symmetric key which is decrypted with
362 the matching private key. If no errors are detected it returns the response without any security
363 properties.

364 **4.5 First Message - Request**

365 **4.5.1 Message Elements and Attributes**

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

370

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
Body	Mandatory
Ping	Mandatory

371 **4.5.2 Message Creation**

372 **4.5.2.1 Security**

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

374 **4.5.2.2 BinarySecurityToken**

375 The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST
376 be labeled with an Id so it can be referenced by the signature. The value MUST be a Public Key

377 certificate suitable for symmetric key encryption. The certificate SHOULD NOT have a KeyUsage
378 extension. If it does contain a KeyUsage extension, it SHOULD include the values of
379 keyEncipherment and dataEncipherment. The Responder must have access to the private key
380 corresponding to the public key in the certificate.

381 **4.5.2.3 EncryptedKey**

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

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

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

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

390 **4.5.2.4 EncryptedData**

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

395 **4.5.2.5 EncryptionMethod**

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

397 **4.5.2.6 CipherData**

398 The CipherData MUST refer to the encrypted attachment with a CipherReference element. The
399 CipherReference element MUST refer to the attachment using a URI with a cid scheme. The
400 CipherReference must have a Transforms child with a single Transform sub child with the value
401 of #Attachment-Content-Only-Transform.

402 **4.5.2.7 Body**

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

404 **4.5.2.8 Ping**

405 The Ping element should contain the scenario number and the name of the entity performing the
406 request.

407 **4.5.2.9 Post Operation Attachment Properties**

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

413 **4.5.3 Responder Message Processing**

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

417 4.5.3.1 Security

418 4.5.3.2 BinarySecurityToken

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

421 4.5.3.3 EncryptedKey

422 The random key contained in the CipherData MUST be decrypted using the private key
423 corresponding to the certificate specified by the SecurityTokenReference, using the specified
424 algorithm.

425 4.5.3.4 EncryptedData

426 The attachment referred to by the EncryptedData MUST be decrypted using the encrypted
427 symmetric key.

428 4.5.3.5 Attachment

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

430 4.5.4 Example (Non-normative)

431 Here is an example request.

```
432 Content-Type: multipart/related; boundary="enc-example" type=text/xml
433 --enc-example
434 Content-Type: text/xml
435
436 <?xml version="1.0" encoding="utf-8" ?>
437 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
438 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
439 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
440 <soap:Header>
441 <wsse:Security soap:mustUnderstand="1"
442 xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
443 secext-1.0.xsd">
444
445 <!-- This certificate is used for symmetric key encryption -->
446 <wsse:BinarySecurityToken
447 Value="X509v3"
448 EncodingType="wsse:Base64Binary"
449 xmlns:wsu="http://docs.oasis
450 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
451 wsu:Id="myEncCert">MII...hk</wsse:BinarySecurityToken>
452
453 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
454 <xenc:EncryptionMethod
455 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
456 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
457 <wsse:SecurityTokenReference>
458 <wsse:Reference URI="#myEncCert" />
459 </wsse:SecurityTokenReference>
460 </KeyInfo>
461 <xenc:CipherData>
462 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
463 </xenc:CipherData>
464 <xenc:ReferenceList>
465 <xenc:DataReference URI="#encrypted-attachment" />
466 </xenc:ReferenceList>
467 </xenc:EncryptedKey>
468
469 <!-- The EncryptedData portion here refers to content of the attachment -->
```

470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501

```
<xenc:EncryptedData wsu:Id="encrypted-attachment"  
  Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-  
1.0#Attachment-Content-Only" MimeType="image/jpeg">  
  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:enc">  
      <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:Transforms>  
      </xenc:CipherReference>  
    </xenc:CipherData>  
  </xenc:EncryptedData>  
  
</wsse:Security>  
</soap:Header>  
<soap:Body>  
  <Ping xmlns="http://xmlsoap.org/Ping">  
    <text>Acme Corp. - Scenario #2</text>  
  </Ping>  
</soap:Body>  
</soap:Envelope>  
--enc-example  
Content-Type: application/octet-stream  
Content-Id: <enc>  
Content-Transfer-Encoding: base64  
Dsh5SA3thsRh3Dh54wafDhjaq2
```

502

503 4.6 Second Message - Response

504 4.6.1 Message Elements and Attributes

505 Items not listed in the following table MUST NOT be created or processed. Items marked
506 mandatory MUST be generated and processed. Items marked optional MAY be generated and
507 MUST be processed if present. Items MUST appear in the order specified, except as noted.
508

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

509

510 4.6.2 Message Creation

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

513 4.6.2.1 Security

514 There are no security properties on the response message

515 **4.6.2.2 Body**

516 The body element MUST be not be signed or encrypted

517 **4.6.3 Message Processing**

518 The response is passed to the application without modification.

519 **4.6.4 Example (Non-normative)**

520 Here is an example response.

```
521 <?xml version="1.0" encoding="utf-8" ?>
522 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
523 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
524 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
525 <soap:Body>
526 <PingResponse xmlns="http://xmlsoap.org/Ping">
527 <text> Acme Corp. - Scenario #2</text>
528 </PingResponse>
529 </soap:Body>
530 </soap:Envelope>
```

531 **4.7 Other processing**

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

533 **4.7.1 Requester**

534 No additional processing is required.

535 **4.7.2 Responder**

536 No additional processing is required.

537 **4.8 Expected Security Properties**

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

541 **5 Scenario #3 – Attachment Signature and**
542 **Encryption**

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

548 **5.1 Attachment Properties**

549 This section specifies the attachment properties BEFORE security operations are applied. The
550 Content-Type of the attachment MUST be text/xml. The Content-Transfer-Encoding MUST be 8-
551 bit ASCII (8-bit). The attachment MUST have a Content-Id header that uniquely identifies the
552 attachment. The generation of the Content-Id header is out of scope.

553 **5.2 Agreements**

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

556 **5.2.1 CERT-VALUE**

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

561 The Responder MUST have access to the private key corresponding to the public key in the
562 certificate.

563 **5.2.2 Signature Trust Root**

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

567 **5.3 Parameters**

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

570 No parameters are required.

571 **5.4 General Message Flow**

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

573 This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used.
574 The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**.
575 The Content-Transfer-Encoding for the encrypted attachment MUST be 8-bit. As required by
576 SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string
577 may be used. The recipient SHOULD ignore the value. The request contains an attachment,
578 which is signed and then encrypted. The certificate for encryption is provided externally to the
579 requestor but conveyed in the request message. The attachment is encrypted with a random

580 symmetric key that is encrypted with a public key certificate. The certificate for signing is included
 581 in the message. The Responder decrypts the attachment using its private key and then verifies
 582 the signature using the included public key certificate. If no errors are detected it returns the
 583 Response with no security properties.

584 5.5 First Message - Request

585 5.5.1 Message Elements and Attributes

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

590

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
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory

Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

591 **5.5.2 Message Creation**

592 **5.5.2.1 Security**

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

594 **5.5.2.2 BinarySecurityToken**

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

600 **5.5.2.3 EncryptedKey**

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

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

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

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

610 **5.5.2.4 EncryptedData**

611 The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be
612 present and it MUST have a value of #Attachment-Content-Only. The EncryptedData element
613 MUST be referenced by the ReferenceList element in the EncryptedKey element. The
614 EncryptedData MUST have a MimeType attribute with the value of text/xml.

615 **5.5.2.5 EncryptionMethod**

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

617 **5.5.2.6 CipherData**

618 The CipherData MUST refer to the encrypted attachment with a CipherReference element. The
619 CipherReference element MUST refer to the attachment using a URI with a cid scheme. The
620 CipherReference must have a Transforms child with a single Transform sub child with the value
621 of #Attachment-Content-Only-Transform.

622 **5.5.2.7 BinarySecurityToken**

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

629 **5.5.2.8 Signature**

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

632 **5.5.2.8.1 SignedInfo**

633 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
634 be RSA-SHA1.

635 **5.5.2.8.2 Reference**

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

639 **5.5.2.8.3 SignatureValue**

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

642 **5.5.2.8.4 KeyInfo**

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

646 **5.5.2.9 Body**

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

648 **5.5.2.10 Post Operation Attachment Properties**

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

654 **5.5.3 Responder Message Processing**

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

658 **5.5.3.1 Security**

659 **5.5.3.2 BinarySecurityToken**

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

662 **5.5.3.3 EncryptedKey**

663 The random key contained in the CipherData MUST be decrypted using the private key
664 corresponding to the certificate specified by the SecurityTokenReference, using the specified
665 algorithm.

666 **5.5.3.4 EncryptedData**

667 The attachment referred to by the EncryptedData MUST be decrypted using the encrypted
668 symmetric key.

669 **5.5.3.5 Attachment**

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

671 **5.5.3.6 BinarySecurityToken**

672 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
673 authorized entity. The public key in the certificate MUST be retained for verification of the
674 signature.

675 **5.5.3.7 Signature**

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

678 **5.5.3.8 Attachment**

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

680 **5.5.4 Example (Non-normative)**

681 Here is an example request.

```
682 Content-Type: multipart/related; boundary="enc-sig-example" type=text/xml
683 --enc-sig-example
684 Content-Type: text/xml
685
686 <?xml version="1.0" encoding="utf-8" ?>
687 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
688 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
689 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
690 <soap:Header>
691 <wsse:Security soap:mustUnderstand="1"
692 xmlns:wsse="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
693 secext-1.0.xsd">
694
695 <!-- This certificate is used for symmetric key encryption -->
696 <wsse:BinarySecurityToken
697 Value="MIIE...hk"
698 ValueType="wsse:X509v3"
699 EncodingType="wsse:Base64Binary"
700 xmlns:wsu="http://docs.oasis
701 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
702 wsu:Id="myEncCert">MIIE...hk</wsse:BinarySecurityToken>
703
704 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
```

```

704 <xenc:EncryptionMethod
705 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
706 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
707 <wsse:SecurityTokenReference>
708 <wsse:Reference URI="#myEncCert" />
709 </wsse:SecurityTokenReference>
710 </KeyInfo>
711 <xenc:CipherData>
712 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
713 </xenc:CipherData>
714 <xenc:ReferenceList>
715 <xenc:DataReference URI="#encrypted-signed-attachment" />
716 </xenc:ReferenceList>
717 </xenc:EncryptedKey>
718
719 <!-- The EncryptedData portion here refers to content of the attachment -->
720
721 <xenc:EncryptedData wsu:Id="encrypted-signed-attachment"
722 Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
723 1.0#Attachment-Content-Only" MimeType="text/xml">
724 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
725 <xenc:EncryptionMethod
726 Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc" />
727 <xenc:CipherData>
728 <xenc:CipherReference URI="cid:encsignexample">
729 <xenc:Transforms>
730 <ds:Transform
731 Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
732 profile-1.0#Attachment-Content-Only-Transform" />
733 </ds:Transforms>
734 </xenc:CipherReference>
735 </xenc:CipherData>
736 </xenc:EncryptedData>
737
738 <!-- This certificate is used to verify the signature -->
739 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
740 EncodingType="wsse:Base64Binary" xmlns:wsu="http://docs.oasis
741 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
742 wsu:Id="mySigCert">MII...hk</wsse:BinarySecurityToken>
743
744 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
745 <SignedInfo>
746 <CanonicalizationMethod
747 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
748 <SignatureMethod
749 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
750 <Reference URI="cid:encsignexample">
751 <Transforms>
752 <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
753 2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform" />
754 </Transforms>
755 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
756 <DigestValue>QTV...dw=</DigestValue>
757 </Reference>
758 </SignedInfo>
759 <SignatureValue>H+x0...gUw=</SignatureValue>
760 <KeyInfo>
761 <wsse:SecurityTokenReference>
762 <wsse:Reference URI="#mySigCert" />
763 </wsse:SecurityTokenReference>
764 </KeyInfo>
765 </Signature>
766 </wsse:Security>
767 </soap:Header>
768 <soap:Body>
769 <Ping xmlns="http://xmlsoap.org/Ping">
770 <text>Acme Corp. - Scenario #3</text>
771 </Ping>
772 </soap:Body>
773 </soap:Envelope>
774 --enc-sig-example

```

```

775 Content-Type: application/octet-stream
776 Content-Id: <encsignexample>
777 Content-Transfer-Encoding: base64
778 FEWMMIIfc93ASjfdjsa358sa98xsjcx
779

```

780

781 5.6 Second Message - Response

782 5.6.1 Message Elements and Attributes

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

786

Name	Mandatory?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

787

788 5.6.2 Message Creation

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

791 5.6.2.1 Security

792 There are no security properties on the response message

793 5.6.2.2 Body

794 The body element MUST be not be signed or encrypted

795 5.6.3 Message Processing

796 The response is passed to the application without modification.

797 5.6.4 Example (Non-normative)

798 Here is an example response.

```

799 <?xml version="1.0" encoding="utf-8" ?>
800 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
801 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
802 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
803 <soap:Body>
804 <PingResponse xmlns="http://xmlsoap.org/Ping">
805 <text> Acme Corp. - Scenario #3</text>
806 </PingResponse>
807 </soap:Body>
808 </soap:Envelope>

```

809

810 **5.7 Other processing**

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

812 **5.7.1 Requester**

813 No additional processing is required.

814 **5.7.2 Responder**

815 No additional processing is required.

816 **5.8 Expected Security Properties**

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

820 **6 References**

821 **6.1 Normative**

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

Appendix A. Ping Application WSDL File

```

829 <definitions xmlns:tns="http://xmlsoap.org/Ping" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
830 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
831 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
832   <types>
833     <schema targetNamespace="http://xmlsoap.org/Ping" xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
834       <import namespace="http://schemas.xmlsoap.org/ws/2003/06/utility" schemaLocation="utility.xsd"/>
835       <!--
836       <complexType name="ticketType">
837         <sequence>
838           <element name="ticket" type="xsd:string"/>
839         </sequence>
840         <attribute ref="wsu:Id"/>
841       </complexType>
842     -->
843     <!--
844     <complexType name="ticketType">
845       <xsd:simpleContent>
846         <xsd:extension base="xsd:string">
847           <xsd:attribute ref="wsu:Id"/>
848         </xsd:extension>
849       </xsd:simpleContent>
850     </complexType>
851     <element name="ticket" type="tns:ticketType"/>
852     <element name="text" type="xsd:string" nillable="true"/>
853     <complexType name="ping">
854       <sequence>
855         <element ref="tns:text"/>
856         <element ref="tns:ticket" minOccurs="0"/>
857       </sequence>
858     </complexType>
859     <complexType name="pingResponse">
860       <sequence>
861         <element ref="tns:text"/>
862       </sequence>
863     </complexType>
864     <element name="Ping" type="tns:ping"/>
865     <element name="PingResponse" type="tns:pingResponse"/>
866   </types>
867   <message name="PingRequest">
868     <part name="ping" element="tns:ping"/>
869   </message>
870   <message name="PingResponse">
871     <part name="pingResponse" element="tns:PingResponse"/>
872   </message>
873   <portType name="PingPort">
874     <operation name="Ping">
875       <input message="tns:PingRequest"/>
876       <output message="tns:PingResponse"/>
877     </operation>
878   </portType>
879   <binding name="PingBinding" type="tns:PingPort">
880     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
881     <operation name="Ping">
882       <soap:operation/>
883       <input>
884         <soap:body use="literal"/>
885       </input>
886       <output>
887         <soap:body use="literal"/>
888       </output>
889     </operation>
890   </binding>
891   <service name="PingService">
892     <port name="Ping1" binding="tns:PingBinding">
893       <soap:address location="http://localhost:9080/pingservice/Ping1"/>
894     </port>
895     <port name="Ping2" binding="tns:PingBinding">
896       <soap:address location="http://localhost:9080/pingservice/Ping2"/>
897     </port>
898     <port name="Ping3" binding="tns:PingBinding">
899       <soap:address location="http://localhost:9080/pingservice/Ping3"/>
900     </port>
901     <port name="Ping4" binding="tns:PingBinding">
902       <soap:address location="http://localhost:9080/pingservice/Ping4"/>
903     </port>
904     <port name="Ping5" binding="tns:PingBinding">
905       <soap:address location="http://localhost:9080/pingservice/Ping5"/>
906     </port>
907     <port name="Ping6" binding="tns:PingBinding">

```

```
908     <soap:address location="http://localhost:9080/pingservice/Ping6"/>
909   </port>
910   <port name="Ping7" binding="tns:PingBinding">
911     <soap:address location="http://localhost:9080/pingservice/Ping7"/>
912   </port>
913 </service>
914 </definitions>
915
916
```

917

Appendix B. Revision History

918

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

919

920

Appendix C. Notices

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

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

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

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

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

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