



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

OASIS Draft 14, 15 November 2004

Document identifier:

wss-swa-profile-1.0-draft-14

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss

Editors:

Frederick Hirsch, Nokia

Contributors:

Michael Hu, Actional
Maneesh Sahu, Actional
Thomas DeMartini, ContentGuard
Dale Moberg, Cyclone Commerce
Dana S. Kaufman, Forum Systems, Inc
Dan Smiley, Forum Systems, Inc
Michael McIntosh, IBM
Frederick Hirsch, Nokia
Jerry Schwarz, Oracle
Blake Dournaee, Sarvega, Inc.
Pete Wenzel, SeeBeyond

Abstract:

This specification defines how to use the OASIS Web Services Security: SOAP Message Security standard [WSS-Sec] with SOAP Messages with Attachments [SwA].

Status:

This is a Draft proposal and has no standing.

Committee members should submit comments and potential errata to the wss@lists.oasis-open.org list. Others should submit them to the wss-comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to wss-comment-subscribe@lists.oasis-open.org with "subscribe" in the body) or use other OASIS-supported means of submitting comments.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the WSS TC (<http://www.oasis-open.org/committees/wss/ipr.php>).

38 Table of Contents

39	1 Introduction.....	3
40	1.1 Notations and Terminology.....	3
41	1.1.1 Notational Conventions.....	3
42	1.1.2 Namespaces.....	4
43	1.1.3 Acronyms and Abbreviations.....	4
44	2 MIME Processing.....	5
45	3 XML Attachments.....	6
46	4 Securing SOAP With Attachments.....	7
47	4.1 Primary SOAP Envelope.....	7
48	4.2 Referencing Attachments.....	7
49	4.3 MIME Part Reference Transforms.....	7
50	4.3.1 Attachment-Content-Only Reference Transform.....	8
51	4.3.2 Attachment-Complete Reference Transform.....	8
52	4.4 Integrity and Data Origin Authentication	8
53	4.4.1 MIME header canonicalization.....	9
54	4.4.2 MIME Content Canonicalization.....	10
55	4.4.3 Protecting against attachment insertion threat.....	10
56	4.4.4 Processing Rules for Attachment Signing.....	10
57	4.4.5 Processing Rules for Attachment Signature Verification.....	11
58	4.4.6 Example Signed Message.....	12
59	4.5 Encryption.....	12
60	4.5.1 MIME Part CipherReference.....	13
61	4.5.2 Encryption Processing Rules.....	13
62	4.5.3 Decryption Processing Rules.....	14
63	4.5.4 Example.....	15
64	4.6 Signing and Encryption.....	16
65	5 References.....	17

1 Introduction

66

67 This document describes how to use the OASIS Web Services Security: SOAP Message Security
68 standard [WSS-Sec] with SOAP Messages with Attachments [SwA]. More specifically, it describes how a
69 web service consumer can secure SOAP attachments using SOAP Message Security for attachment
70 integrity, confidentiality and origin authentication, and how a receiver may process such a message.

71 A broad range of industries - automotive, insurance, financial, pharmaceutical, medical, retail, etc - require
72 that their application data be secured from its originator to its ultimate consumer. While some of this data
73 will be XML, quite a lot of it will not be. In order for these industries to deploy web service solutions, they
74 need an interoperable standard for end-to-end security for both their XML data and their non-XML data.

75 Profiling SwA security may help interoperability between the firms and trading partners using attachments
76 to convey non-XML data that is not necessarily linked to the XML payload. Many industries, such as the
77 insurance industry require free-format document exchange in conjunction with web services messages.
78 This profile of SwA should be of value in these cases.

79 In addition, some content that could be conveyed as part of the SOAP body may be conveyed as an
80 attachment due to its large size to reduce the impact on message and XML processing, and may be
81 secured as described in this profile.

82 This profile is applicable to using SOAP Message Security in conjunction with SOAP Messages with
83 Attachments (SwA). This means the scope is limited to SOAP 1.1, the scope of SwA.

84 The existence of this profile does not preclude using other mechanisms to secure attachments conveyed
85 in conjunction with SOAP messages, including the use of XML security technologies at the application
86 layer or the use of security for the XML Infoset before a serialization that uses attachment technology
87 [MTOM]. The requirements in this profile only apply when securing SwA attachments explicitly according
88 to this profile.

89 **Note that in this document, lists of processing steps are descriptive in that an**
90 **implementation may use a different procedure as long as the result is the same.**

1.1 Notations and Terminology

91

92 This section specifies the notations, namespaces, and terminology used in this specification.

1.1.1 Notational Conventions

93

94 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
95 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
96 described in IETF RFC 2119 [RFC2119].

97 `Listings of productions or other normative code appear like this.`

98 `Example code listings appear like this.`

99 **Note: Non-normative notes and explanations appear like this.**

100 When describing abstract data models, this specification uses the notational convention used by the XML
101 Infoset. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

102 When describing concrete XML schemas [XML-Schema], this specification uses the notational convention
103 of OASIS Web Services Security: SOAP Message Security. Specifically, each member of an element's
104 [children] or [attributes] property is described using an XPath-like [XPath] notation (e.g., /
105 x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard
106 (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

107 Commonly used security terms are defined in the Internet Security Glossary [SECGLO]. Readers are
108 presumed to be familiar with the terms in this glossary as well as the definitions in the SOAP Message
109 Security specification [WSS-Sec] .

110 1.1.2 Namespaces

111 Namespace URIs (of the general form "some-URI") represent application-dependent or context-
112 dependent URIs as defined in RFC 2396 [URI]. This specification is designed to work with the SOAP 1.1
113 [SOAP11] message structure and message processing model, the version of SOAP supported by SOAP
114 Messages with Attachments. The current SOAP 1.1 namespace URI is used herein to provide detailed
115 examples.

116 The namespaces used in this document are shown in the following table (note that for brevity, the
117 examples use the prefixes listed below but do *not* include the URIs – those listed below are assumed).

Prefix	Namespace
S11	http://schemas.xmlsoap.org/soap/envelope/
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
wsswa	http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0.xsd

118 The URLs provided for the *wsse* and *wsu* namespaces can be used to obtain the schema files.

119 **Note: When this document is finalized the wsswa URL will be updated, replacing**
120 **XX values and possibly making other changes.**

121 1.1.3 Acronyms and Abbreviations

122 The following (non-normative) table defines acronyms and abbreviations for this document, beyond those
123 defined in the SOAP Message Security standard.

Term	Definition
CID	Content ID scheme for URLs. Refers to Multipart MIME body part, that includes both MIME headers and content for that part. [RFC2392]
SwA	SOAP Messages with Attachments [SwA]

2 MIME Processing

125 This profile is concerned with the securing SOAP messages with attachments, attachments that are
126 conveyed as MIME parts in a multi-part MIME message as outlined in SOAP Messages with Attachments.
127 In effect this combines two processing layers, a SOAP messaging layer and a MIME wrapping. A SOAP
128 sender effectively transmits a SOAP message and corresponding attachments by passing them to a
129 MIME layer that serializes them. A SOAP receiver receives a message and attachments after the MIME
130 layer processes the MIME serialization. This is important since certain aspects of the MIME processing
131 may be changed at different intermediary transport nodes, yet remain transparent to the SOAP layer. For
132 example, a MIME processing node may change the transfer encoding of a MIME part, transparently to the
133 SOAP nodes. The MIME layer may translate to and from a transfer encoding upon serialization and de-
134 serialization.

135 The importance to this profile is two-fold. First, it means that certain aspects of MIME processing, such as
136 transfer encoding processing, are out of scope of the profile and do not need to be addressed. Secondly, it
137 means that many of the MIME headers are also out of scope of the profile and the profile does not support
138 integrity protection of these headers, since they are expected to change. If more security protection is
139 required then it must occur at a protocol layer below the MIME layer, for example transport security (with
140 the understanding that such security may not always apply end-end).

141 SOAP message security is intended to provide security at the SOAP messaging layer, including support
142 for SOAP intermediaries. Thus this profile supports securing the attachment content, possibly including
143 MIME headers that are associated directly with the content (such as Content-Type, Content-Length and
144 other Content related MIME headers) and not MIME headers associated with MIME serialization. This
145 simplifies the profile and also delineates the layering.

146

3 XML Attachments

147 A SOAP Messages with Attachments multi-part MIME structure contains a primary SOAP envelope in the
148 root part and one or more attachments in additional MIME parts. Some of these attachments may be have
149 a content type corresponding to XML, but do not contain the primary SOAP envelope to be processed.

150 Attachments associated with the SOAP body are targeted at the SOAP Ultimate Receiver along with the
151 SOAP body, and may be processed at the application layer along with the body. This means that XML
152 processing may not be required for such XML media type MIME attachments until application layer
153 processing is performed. For this reason the SOAP message layer may not need to perform XML
154 canonicalization or parsing for such attachments and SOAP Message layer security may treat these
155 attachments as text.

156 Attachments might also be associated with SOAP headers and targeted toward specific SOAP
157 intermediaries, or actors. For SOAP headers specific to an application the attachment content is
158 processed at the application layer, logically after SOAP message processing is complete.

159 This profile assumes that SOAP attachments (not including the root part containing the primary SOAP
160 envelope) need not be processed as XML at the SOAP messaging layer, so do not require SOAP
161 canonicalization or XML parsing and may be treated as opaque data by the SOAP Message Security layer
162 security processing. MIME part canonicalization (as described below) is required to enable effective SOAP
163 Message Security signatures that include SOAP with Attachments.

164 4 Securing SOAP With Attachments

165 Attachments may be associated with SOAP messages, as outlined in SOAP Messages with Attachments
166 (SwA). This profile defines how such attachments may be secured for integrity and confidentiality using
167 the OASIS Web Services Security: SOAP Message Security standard. This does not preclude using other
168 techniques. The requirements in this profile only apply when securing SwA attachments explicitly
169 according to this profile.

170 This profile considers all attachments as opaque whether they are XML or some other content type. It is
171 the sole responsibility of the application to perform further interpretation of attachments that happen to be
172 XML, including the ability to sign or encrypt portions of those attachments.

173 4.1 Primary SOAP Envelope

174 When SOAP attachments are used as specified in [SwA] each SOAP message is accompanied by a
175 MIME header and possibly multiple boundary parts. This is known as a SOAP message package. This
176 document assumes that a proper SOAP message package is constructed using the HTTP and MIME
177 headers appropriate to [SwA].

178 The primary SOAP envelope SHOULD be conveyed in the first MIME part, but MAY be conveyed in
179 another MIME part when the start attribute is specified in the HTTP Multipart/Related header.

180 In particular, implementations should take care in distinguishing between the HTTP headers in the SOAP
181 message package and the start of the SOAP payload. For example, the following Multipart/Related
182 header belongs to the HTTP layer and not the main SOAP payload:

```
183 Content-Type: Multipart/Related; boundary=xyl; type="text/xml"; start="<foo>"
```

184 The main SOAP payload begins with the appropriate boundary. For example:

```
185 --xyl  
186 Content-Type: text/xml; charset=utf-8  
187 Content-ID: <foo>  
  
188 <?xml version='1.0' ?>  
189 <s11:Envelope xmlns:s11="http://schemas.xmlsoap.org/soap/envelope/" />
```

190 4.2 Referencing Attachments

191 SOAP Messages with Attachments defines two MIME mechanisms for referencing attachments. The first
192 mechanism uses a CID scheme URL to refer to the attachment that has a Content-ID MIME header with a
193 value corresponding to the URL, as defined in [RFC 2392]. For example, a content id of "foo" may be
194 specified in the MIME part with the MIME header "Content-ID: <foo>" and be referenced using the CID
195 Schema URL "cid:foo".

196 The second mechanism is to use a URL to refer to an attachment containing a Content-Location MIME
197 header. In this case the URL may require resolution to determine the referenced attachment [RFC2557].

198 For simplicity and interoperability this profile limits WS-Security references to attachments to CID scheme
199 URLs. Attachments referenced from WS-Security signature references or cipher references MUST be
200 referenced using CID scheme URLs.

201 4.3 MIME Part Reference Transforms

202 By definition of RFC 2392, a URI reference to a MIME attachment includes the MIME headers associated

203 with that attachment as well as the MIME part content [RFC2392]. Since there may be some confusion as
204 to what is referenced, it is useful to clearly indicate what is included in the referenced attachment. In
205 addition, some applications may wish to only encrypt or include the attachment content in a signature
206 reference hash, and others may wish to include MIME headers and content.

207 For these reasons, this profile defines two transforms, allowing a clear and explicit statement of what is
208 included in a MIME reference. These transforms are called "MIME Part Reference Transforms".

209 **4.3.1 Attachment-Content-Only Reference Transform**

210 The Attachment-Content-Only transform indicates that only the content of a MIME part is referenced. This
211 transform MUST be identified using the URI value: [http://docs.oasis-open.org/wss/2004/XX/oasis-
212 2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform](http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform).

213 **Note: When this document is finalized this URL will be updated, replacing XX**
214 **values and possibly making other changes.**

215 When this transform is used the content of the MIME part should be MIME canonicalized as defined in
216 section 4.3.2.

217 **4.3.2 Attachment-Complete Reference Transform**

218 The Attachment-Complete transform indicates that both the content and selected headers of the MIME
219 part are referenced. This transform MUST be identified using the URI value: [http://docs.oasis-
220 open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Complete-Transform](http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Complete-Transform).

221 **Note: When this document is finalized this URL will be updated, replacing XX**
222 **values and possibly making other changes.**

223 This transform specifies that in addition to the content the following MIME headers are to be included
224 (when present):

- 225 • Content-Description
- 226 • Content-Disposition
- 227 • Content-ID
- 228 • Content-Location
- 229 • Content-Type

230 These headers are included because of their common use and the risks associated with inappropriate
231 modification. If other headers are to be protected, other mechanisms at the application level should be
232 used (such as copying values into a SOAP header) and this is out of scope of this profile.

233 Other MIME headers associated with the MIME part serialization are not referenced by the transform and
234 are not to be included in signature or encryption calculations.

235 When this transform is used the MIME headers should be canonicalized as defined in section 4.3.1 and
236 the MIME content should be MIME canonicalized as defined in section 4.3.2.

237 **4.4 Integrity and Data Origin Authentication**

238 Integrity and data origin authentication may be provided for SwA attachments using XML Digital
239 Signatures, as outlined in the SOAP Message Security standard as profiled in this document. This is
240 useful independent of the content of the MIME part – for example, it is possible to sign a MIME part that
241 already contains a signed object created by an application. It may still be sensible to sign such an
242 attachment as part of SOAP Message security so that the receiving SOAP node may verify that all
243 attachments are intact before delivering them to an application. A SOAP intermediary may also choose to
244 perform this verification, even if the attachments are not otherwise processed by the intermediary.

245 4.4.1 MIME header canonicalization

246 The result of MIME header canonicalization is a UTF-8 encoded octet stream.

247 Each of the MIME headers listed for the Attachment-Complete transform MUST be canonicalized as part
248 of that transform processing, as outlined in this section. This means the transform MUST perform the
249 following actions in interpreting the MIME headers for signature creation or verification (this order is not
250 prescriptive as long as the same result is obtained)

- 251 1. The transform MUST process MIME headers before the MIME content.
- 252 2. The transform MUST only process MIME headers that are explicitly present in the attachment part and
253 are listed in the Attachment-Complete transform section of this specification, except that a MIME part
254 without a Content-Type header MUST be treated as having a Content-Type header with the value
255 "Content-Type: text/plain; charset=us-ascii". MIME headers not listed in the Attachment-Complete
256 transform section of this specification are to be ignored by the transform.
- 257 3. The MIME headers MUST be processed by the Attachment-Complete transform in lexicographic order
258 (ascending).
- 259 4. The MIME header names MUST be processed by the transform as having the case according to the
260 MIME specifications (as shown in the Attachment-Complete section).
- 261 5. The MIME header values MUST be unfolded [RFC2822].
- 262 6. Any Content-Description MIME header containing RFC2047 encoding MUST be decoded [RFC2047].
- 263 7. Folding whitespace in structured MIME headers (e.g. Content-Disposition, Content-ID, Content-
264 Location, Content-Type) that is not within quotes MUST be removed. Folding whitespace in structured
265 MIME headers that is within quotes MUST be preserved. Folding whitespace in unstructured MIME
266 headers (e.g. Content-Description) MUST be preserved [RFC2822]. For example, whitespace
267 immediately following the colon delimiter in the structured Content-Type header MUST be removed,
268 but whitespace immediately following the colon delimiter in the unstructured Content-Description
269 header MUST be preserved.
- 270 8. Comments in MIME header values MUST be removed [RFC2822].
- 271 9. Case-insensitive MIME header values (e.g. media type/subtype values and disposition-type values)
272 MUST be converted to lowercase. Case-sensitive MIME header values MUST be left as is with
273 respect to case [RFC2045].
- 274 10. Quoted characters other than double-quote and backslash ("\") in quoted strings in structured MIME
275 headers (e.g. Content-ID) MUST be unquoted. Double-quote and backslash ("\") characters in quoted
276 strings in structured MIME headers MUST be character encoded [RFC2822].
- 277 11. Canonicalization of a MIME header MUST generate a UTF-8 encoded octet stream containing the
278 following: the MIME header name, a colon (":"), the MIME header value, and the result of
279 canonicalizing the MIME header parameters in lexicographic order (ascending) as described below.
- 280 12. MIME header parameter names MUST be converted to lowercase [RFC2045].
- 281 13. MIME parameter values containing RFC2184 character set, language, and continuations MUST be
282 decoded. The resulting canonical output MUST not contain the RFC2184 encoding [RFC2184].
- 283 14. Case-insensitive MIME header parameter values MUST be converted to lowercase. Case-sensitive
284 MIME header parameter values MUST be left as is with respect to case [RFC2045].
- 285 15. Enclosing double-quotes MUST be added to MIME header parameter values that do not already
286 contain enclosing quotes. Quoted characters other than double-quote and backslash ("\") in MIME
287 header parameter values MUST be unquoted. Double-quote and backslash characters in MIME
288 parameter values MUST be character encoded.
- 289 16. Canonicalization of a MIME header parameter MUST generate a UTF-8 encoded octet stream

290 containing the following: a semi-colon (";"), the parameter name (lowercase), an equals sign ("="), and
291 the double-quoted parameter value.

292 17. Each header MUST be terminated by a single CRLF pair, without any trailing whitespace.

293 18. The last header MUST be followed by a single CRLF and then the MIME content.

294 **4.4.2 MIME Content Canonicalization**

295 Before including attachment content in a signature reference hash calculation, that MIME attachment may
296 need to be MIME canonicalized. The exact details of MIME part canonicalization depend on the Content-
297 Type of the MIME part. To quote the S/MIME specification (section 3.1.1 "Canonicalization") which deals
298 with this issue [[RFC2633](#)]:

299 The exact details of canonicalization depend on the actual MIME type and subtype of an
300 entity, and are not described here. Instead, the standard for the particular MIME type should
301 be consulted. For example, canonicalization of type text/plain is different from
302 canonicalization of audio/basic. Other than text types, most types have only one
303 representation regardless of computing platform or environment which can be considered
304 their canonical representation.

305 MIME types are registered. This registration includes a section on "Canonicalization and Format
306 Requirements" [[RFC2048](#)] and requires each MIME type to have a canonical representation.

307 The MIME "text" type canonical form is defined in the MIME conformance specification (See "Canonical
308 Encoding Model") [[RFC2049](#)]. Important aspects of "text" media type canonicalization include line ending
309 normalization to <CR><LF> and ensuring that the charset is a registered charset (see RFC 2633 section
310 "Canonicalization"). [[RFC2633](#), [CHARSETS](#), [RFC2045](#)].

311 MIME attachment parts (other than the part containing the primary SOAP envelope) that contain XML do
312 NOT require XML Canonicalization according to this profile, given the rationale in the previous section on
313 XML attachments. These parts MUST be MIME canonicalized according the MIME "text" part
314 requirements. MIME part canonicalization must be performed before signature hash generation or
315 verification is performed. Signature validation requires an identical hash of content requiring consistent
316 MIME part content.

317 **4.4.3 Protecting against attachment insertion threat**

318 Including an attachment in a signature calculation enables a receiver to detect modification of that
319 attachment. Including all attachments in a signature calculation, by providing a <ds:Reference> for each,
320 protects against the threat of attachment removal. This does not protect against insertion of a new
321 attachment.

322 The simplest protection against attachment insertion is for the receiver to know that all attachments
323 should be included in a signature calculation – unreferenced attachments are then an indication of an
324 attachment insertion attack.

325 Such information may be communicated in or out of band. Definition of these approaches is out of the
326 scope of this profile.

327 **4.4.4 Processing Rules for Attachment Signing**

328 The processing rule for signing is modified based on the SOAP Message Security rules.

329 After determining which attachments are to be included as references in a signature, create a
330 <ds:Signature> element in a <wsse:Security> header block targeted at the recipient, including a
331 <ds:Reference> for each attachment to be protected by the signature. Additional <ds:Reference>
332 elements may refer to content in the SOAP envelope to be included in the signature.

- 333 For each attachment Reference, perform the following steps:
- 334 1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part.
 - 335 2. If MIME headers are to be included in the signature, MIME part canonicalize the headers listed in this
336 profile as outlined above.
 - 337 3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL
338 attribute value to this URL.
 - 339 4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST
340 include a <ds:Transform> element with the Algorithm attribute having the URL value specified in this
341 profile - either Attachment-Complete or Attachment-Content-Only, depending on what is to be
342 included in the hash calculation. This MUST be the first transform listed.
 - 343 5. Extract the appropriate portion of the MIME part consistent with the selected transform.
 - 344 6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature
345 Recommendation.

346 **4.4.5 Processing Rules for Attachment Signature Verification**

347 Signature verification is performed as outlined in SOAP Message Security and the XML Digital Signature
348 Recommendation, with the following considerations for SwA attachments.

349 To verify <ds:Reference> hashes for SwA attachments, the following steps must be performed for each
350 reference to an attachment:

- 351 1. Find the attachment corresponding to the <ds:Reference> URL attribute value. This value MUST
352 correspond to the Content-ID for the attachment[SwA].
- 353 2. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part.
- 354 3. If MIME headers were included in the signature, canonicalize the headers listed in this profile as
355 outlined above.
- 356 4. Extract the appropriate portion of the MIME part according to the MIME Part Signature Transform
357 value.
- 358 5. Calculate the reference hash and verify the reference.

359 4.4.6 Example Signed Message

```
360 Content-Type: multipart/related; boundary="arggh" type="text/xml"
361 --arggh
362 Content-Type: text/xml
363 <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
364 xmlns:ds="..." xmlns:xenc="...">
365   <S11:Header>
366     <wsse:Security>
367       <ds:Signature>
368         <ds:SignedInfo>
369           <ds:CanonicalizationMethod Algorithm=
370 'http://www.w3.org/2001/10/xml-exc-c14n#'/>
371           <ds:SignatureMethod Algorithm=
372 'http://www.w3.org/2000/09/xmldsig#rsa-shal' />
373           <ds:Reference URI="cid:bar">
374             <ds:Transforms>
375               <ds:Transform Algorithm="http://docs.oasis-
376 open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-
377 Only-Transform"/>
378             </ds:Transforms>
379             <ds:DigestMethod Algorithm=
380 "http://www.w3.org/2000/09/xmldsig#sha1"/>
381             <ds:DigestValue>j6lwx3rvEP00vKtMup4NbeVu8nk=</ds:DigestValue>
382           </ds:Reference>
383         </ds:SignedInfo>
384         <ds:SignatureValue>DeadBeef</ds:SignatureValue>
385       </ds:Signature>
386     </wsse:Security>
387   </S11:Header>
388   <S11:Body>
389     some items
390   </S11:Body>
391 </S11:Envelope>
392 --arggh
393 Content-Type: image/png
394 Content-ID: <bar>
395 Content-Transfer-Encoding: base64
396 the image
```

397 4.5 Encryption

398 A SwA attachment may be encrypted for confidentiality protection, protecting either the MIME part content
399 including selected MIME headers, or only the MIME part content.

400 This is done using XML Encryption to encrypt the attachment, placing the resulting cipher text in the
401 updated attachment body replacing the original content, and placing a new <xenc:EncryptedData>
402 element in the <wsse:Security> header. An <xenc:CipherReference> MUST link the
403 <xenc:EncryptedData> element with the cipher data.

404 The key used for encryption MAY be conveyed using an <xenc:EncryptedKey> element in the
405 <wsse:Security> header. In this case the <xenc:ReferenceList> element in the <xenc:EncryptedKey>
406 element MUST contain an <xenc:DataReference> with a URI attribute specifying the
407 <xenc:EncryptedData> element in the <wsse:Security> header corresponding to the attachment.

408 When the same <xenc:EncryptedKey> corresponds to multiple <xenc:EncryptedData> elements, the
409 <xenc:ReferenceList> in the <xenc:EncryptedKey> element SHOULD contain an <xenc:DataReference>
410 for each <xenc:EncryptedData> element, both for attachments and encrypted items in the primary SOAP
411 envelope. References should be ordered to correspond to ordering of the security header elements.

412 When an <xenc:EncryptedKey> element is not used when encrypting an attachment, then the

413 <xenc:EncryptedData> element MAY contain a <ds:KeyInfo> element to specify a key as outlined in the
414 SOAP Message Security standard. Different deployments may have different requirements on how keys
415 are referenced. When an <xenc:EncryptedKey> element is used the <xenc:EncryptedData> element
416 MUST NOT contain a <ds:KeyInfo> element.

417 When an attachment is encrypted, no <xenc:ReferenceList> element is placed as a direct child of the
418 <wsse:Security> header, since the <xenc:EncryptedData> element is present in the header, eliminating
419 the need for this reference. Although the SOAP Message Security standard recommends the use of
420 <xenc:ReferenceList>, this is only necessary when the <xenc:EncryptedData> element is not present in
421 the <wsse:Security> header. (As mentioned, when the key is conveyed in an <xenc:EncryptedKey>
422 element, then this element will have a ReferenceList Reference to the <xenc:EncryptedData> element).

423 **Note: The same CID is used to refer to the attachment before encryption and after.**
424 **This avoids the need to rewrite references to the attachment, avoiding issues**
425 **related to generating unique CIDs and relating to preserving the correspondence to**
426 **the original WSDL definition.**

427 4.5.1 MIME Part CipherReference

428 This profile requires that <xenc:EncryptedData> elements corresponding to encrypted SwA attachments
429 use a <xenc:CipherReference> to refer to the cipher text, to be conveyed in the attachment. Upon
430 encryption the MIME part attachment content is replaced with the encoded cipher text.

431 The <xenc:CipherReference> MUST have a <xenc:Transforms> child element. This element MUST have
432 a <ds:Transform> child having an Algorithm attribute with a URI value specifying the Content-Only MIME
433 Part Reference Transform. This transform explicitly indicates that when dereferencing the MIME part
434 reference that only the MIME part content is to be used as the cipher value.

435 The cipher data conveyed in the attachment MAY be base64 encoded, in which case one of the following
436 statements MUST be followed:

- 437 1. The base64 transform MUST be specified in the CipherReference transform list, OR
- 438 2. the base64 encoding MUST be processed transparently through appropriate transport mechanisms
439 defined outside the scope of this specification, i.e. Content-Transfer-Encoding: base64.

440 The base64 transform and base64 Content-Transfer-Encoding SHOULD NOT be specified together as
441 this would indicate that the cipher data would be double encoded.

442 When the base64 transform algorithm is specified, the following URI MUST be used for the ds:Transform
443 Algorithm attribute value: <http://www.w3.org/2000/09/xmldsig#base64> .

444 The cipher data conveyed in the attachment MAY be an unencoded octet stream, in which case a base64
445 transform MUST NOT be specified.

446 4.5.2 Encryption Processing Rules

447 The order of the following steps is not normative, although the result should be the same as if this order
448 were followed.

- 449 1. When encrypting both attachments and primary SOAP envelope content using the same key, perform
450 the attachment processing first.

451 **Note: The SOAP Message Security standard states that elements should be**
452 **prepended to the security header. This processing rule supports putting the**
453 **<xenc:EncryptedData> element first in the header with <xenc:EncryptedKey> and**
454 **tokens following. Thus, a receiver should be able to process the**
455 **<xenc:EncryptedKey> before the <xenc:EncryptedData> element for the**
456 **attachment.**

- 457 2. Encrypt the attachment part using XML Encryption, according to the rules of XML Encryption. Encrypt
458 either the attachment including content and selected MIME headers or only the attachment content.
- 459 3. Set the <xenc:EncryptedData> Type attribute value to a URI that specifies adherence to his profile and
460 that specifies what was encrypted (MIME content or entire MIME part including headers). The following
461 URIs MUST be used for this purpose:
- 462 • Content Only: <http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only>.
 - 464 • Content and headers: <http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Complete>
- 466 **Note: When this document is finalized these URLs will be updated, replacing XX**
467 **values and possibly making other changes. Note that these URLs should match the**
468 **related transforms apart from -Transform.**
- 469 4. Set the <xenc:EncryptedData> MimeType attribute to match the attachment MIME part Content-Type
470 header before encryption when Content-Only URI is specified for the Type attribute value. The
471 MimeType attribute value may also be set when the AttachmentComplete Type attribute value is
472 specified.
- 473 5. Set the <xenc:EncryptedData> <xenc:CipherReference> to the same reference URL for the
474 attachment that was used before encryption . This MUST be a CID scheme URL referring to the
475 attachment part Content-ID. Ensure this MIME header is in the part conveying the cipher data after
476 encryption.
- 477 6. Include the Content-Only MIME Part Reference Transform in the <xenc:CipherReference>
478 <xenc:Transforms> list. If the cipher data is base64 encoded then either the base64 transform MUST
479 also be included, following the Content-Only Transform, or other means MUST be used to convey the
480 fact that the cipher data is base64 encoded. If the cipher data is not base64 encoded then the base64
481 transform MUST NOT be included in the MIME Part CipherReference transform. See section 4.4.1.
- 482 7. Prepend the <xenc:EncryptedData> element to the <wsse:Security> SOAP header block. Do NOT add
483 a <xenc:ReferenceList> element to the SOAP header block (even though recommended by SOAP
484 Message Security).
- 485 8. Update the attachment MIME part, replacing the original content with the cipher text generated by the
486 XML Encryption step.
- 487 9. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the
488 cipher data.

489 4.5.3 Decryption Processing Rules

490 The <xenc:CipherReference> URL MUST be a URL that refers to the MIME part containing the cipher
491 text, and must also correspond to the reference value of the original attachment that was encrypted. This
492 MUST be a CID scheme URL.

493 Decryption may be initiated upon locating the <xenc:EncryptedData> element in the <wsse:Security>
494 header.

495 The following decryption steps must be performed so that the result is as if they were performed in this
496 order:

- 497 1. Extract the cipher text from the attachment referenced by the <xenc:CipherReference> URL attribute.
498 The MIME Part CipherReference Transform defined in this profile indicates that the MIME part content
499 is extracted.

- 500 2. If the cipher data is base64 encoded then base64 decode it. It is base64 encoded if the base64
501 transform is included in the <xenc:CipherReference> <xenc:Transforms> list or if other means are
502 used to indicate base64 encoding, as discussed in section 4.4.1.
- 503 3. Decrypt the cipher text using the information present in the appropriate <xenc:EncryptedData> element
504 and possibly other out of band information, according to the XML Encryption Standard.
- 505 4. If the <xenc:EncryptedData>Type attribute indicates that selected MIME headers were encrypted, then
506 those MIME headers MUST be replaced by the result of decryption, as well as the MIME part content.
- 507 5. If the <xenc:EncryptedData>Type attribute indicates that only the content of the MIME part was
508 encrypted, then the cipher text content of the attachment part MUST be replaced by the result of
509 decryption. In this case the MIME part Content-Type header value MUST be replaced by the
510 <xenc:EncryptedData> MimeType attribute value.

511 4.5.4 Example

512 This example shows encryption of the primary SOAP envelope body as well as an attachment using a
513 single symmetric key conveyed using an EncryptedKey element.

```

514 Content-Type: multipart/related; boundary="arggh" type="text/xml"
515 --arggh
516 Content-Type: text/xml

517 <S11:Envelope
518   xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
519   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
520 wsswssecurity-secext-1.0.xsd"
521   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
522   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

523   <S11:Header>
524     <wsse:Security>

525       <wsse:BinarySecurityToken wsu:Id="Acert"
526         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
527 200401-wss-soap-message-security-1.0#Base64Binary"
528         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
529 200401-wss-x509-token-profile-1.0#x509v3">
530         ...
531       </wsse:BinarySecurityToken>

532     <xenc:EncryptedKey Id='EK'>
533       <EncryptionMethod
534         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
535       <ds:KeyInfo Id="keyinfo">
536         <wsse:SecurityTokenReference>
537           <ds:X509Data>
538             <ds:X509IssuerSerial>
539               <ds:X509IssuerName>
540                 DC=ACMECorp, DC=com
541               </ds:X509IssuerName>
542             <ds:X509SerialNumber>12345678</ds:X509SerialNumber>
543             </ds:X509IssuerSerial>
544           </ds:X509Data>
545         </wsse:SecurityTokenReference>
546       </ds:KeyInfo>
547       <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
548       <ReferenceList>
549         <DataReference URI='#EA' />
550         <DataReference URI='#ED' />
551       </ReferenceList>
552     </EncryptedKey>

```

```

553     <xenc:EncryptedData
554         Id='EA'
555         Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
556 profile-1.0#Attachment-Content-Only"
557         MimeType="image/png">
558         <xenc:EncryptionMethod
559             Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc' />
560         <xenc:CipherData>
561             <xenc:CipherReference URI=cid:bar">
562                 <xenc:Transforms>
563                     <ds:Transform Algorithm="http://docs.oasis-
564 open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-
565 Only-Transform"/>
566                 </xenc:Transforms>
567             </xenc:CipherReference>
568         </xenc:CipherData>
569     </xenc:EncryptedData>
570
571 </wsse:Security>
572 </S11:Header>
573 <S11:Body>
574     <xenc:EncryptedData Id='ED'
575         <xenc:EncryptionMethod
576             Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc' />
577         <xenc:CipherData>
578             <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
579         </xenc:CipherData>
580     </xenc:EncryptedData>
581 </S11:Body>
582 </S11:Envelope>
583 --arggh
584 Content-Type: application/octet-stream
585 Content-ID: <bar>
586 Content-Transfer-Encoding: binary
587 BinaryCipherData

```

587 4.6 Signing and Encryption

588 When portions of content are both signed and encrypted, there is possible confusion as to whether
589 encrypted content need first be decrypted before signature verification. This confusion can occur when
590 the order of operations is not clear [[DecryptT](#)]. This problem may be avoided with SOAP Message Security
591 for SwA attachments when attachments and corresponding signatures and encryptions are targeted for a
592 single SOAP recipient (actor). The SOAP Message Security standard explicitly states that there may not
593 be two <wsse:Security> headers targeted at the same actor, nor may there be two headers without a
594 designated actor. In this case the SOAP Message Security and SwA profile processing rules may
595 eliminate ambiguity since each signing or encryption produces an element in the <wsse:Security> header,
596 and these elements are ordered. (Signing produces <ds:Signature> elements and encryption produces
597 <xenc:EncryptedData> elements).

598 If an application produces different <wsse:Security> headers targeted at different recipients, these are
599 processed independently by the recipients. Thus there is no need to correlate activities between distinct
600 headers – the order is inherent in the SOAP node model represented by the distinct actors.

5 References

601

- 602 **[CHARSETS]** Character sets assigned by IANA. See <ftp://ftp.isi.edu/in->
603 [notes/iana/assignments/character-sets](ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets).
- 604 **[DecryptT]** M. Hughes et al, "Decryption Transform for XML Signature", W3C Recommendation 10
605 December 2002 <http://www.w3.org/TR/xmlenc-decrypt/>.
- 606 **[MTOM]** *Work in Progress – subject to change*. SOAP Message Transmission Optimization
607 Mechanism, W3C Candidate Recommendation 26 August 2004,
608 <http://www.w3.org/TR/soap12-mtom/>.
- 609 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
610 Bodies, IETF RFC 2045, November 1996, <http://www.ietf.org/rfc/rfc2045.txt>.
- 611 **[RFC2046]** Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, IETF RFC 2046,
612 November 1996, <http://www.ietf.org/rfc/rfc2046.txt>.
- 613 **[RFC2047]** Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions
614 for Non-ASCII Text, IETF RFC 2047, November 1996, <http://www.ietf.org/rfc/rfc2047.txt>.
- 615 **[RFC2048]** Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures,
616 <http://www.ietf.org/rfc/rfc2048.txt>.
- 617 **[RFC2049]** Multipurpose Internet Mail Extensions(MIME) Part Five: Conformance Criteria and
618 Examples, <http://www.ietf.org/rfc/rfc2049.txt>.
- 619 **RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119,
620 March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 621 **[RFC2392]** E. Levinson, *Content-ID and Message-ID Uniform Resource Locators*, IETF RFC 2392,
622 <http://www.ietf.org/rfc/rfc2392.txt>
- 623 **[RFC2557]** MIME Encapsulation of Aggregate Documents, such as HTML (MHTML), IETF RFC
624 2557, March 1999, <http://www.ietf.org/rfc/rfc2557.txt>.
- 625 **[RFC2633]** Ramsdell B., "S/MIME Version 3 Message Specification", Standards Track RFC 2633,
626 June 1999. <http://www.ietf.org/rfc/rfc2633.txt>
- 627 **[RFC2822]** Internet Message Format, IETF RFC 2822, April 2001, <http://www.ietf.org/rfc/rfc2822.txt>.
- 628 **[SECGLO]** Informational RFC 2828, "Internet Security Glossary," May 2000.
- 629 **[SOAP11]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
- 630 **[SwA]** W3C Note, "SOAP Messages with Attachments", 11 December 2000,
631 <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211> .
- 632 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic
633 Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998,
634 <http://www.ietf.org/rfc/rfc2396.txt>.
- 635 **[WS-I-AP]** *Final Material* Attachments Profile Version 1.0, 2004-08-24, [http://www.ws-](http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html)
636 [i.org/Profiles/AttachmentsProfile-1.0.html](http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html)
- 637 **[WSS-Sec]** A. Nadalin et al., Web Services Security: SOAP Message Security 1.0 (WS-Security
638 2004), OASIS Standard 200401, March 2004, [http://docs.oasis-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
639 [open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 640 **[XML-Schema]** W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001,
641 <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.
- 642 W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001,
643 <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.
- 644 **[XPath]** W3C Recommendation, "XML Path Language", 16 November 1999,
645 <http://www.w3.org/TR/xpath>.

646 **A. Acknowledgments**

647 The editors would like to acknowledge the contributions of the OASIS WSS Technical Committee, whose
648 voting members at the time of publication were:

- 649 • TBD

B. Revision History

Rev	Date	By Whom	What
1	05/25/04	Frederick Hirsch	Initial version, put draft proposal into profile format.
2	05/26/04	Frederick Hirsch	Editorial and namespace suggestions from Michael McIntosh. Added rationale for SwA support to introduction. Completely rewrote processing rules for encryption and decryption.
3	05/28/04	Frederick Hirsch	Rewrote signature section, fixed cid references and Content-IDs, added examples.
4	06/12/04	Frederick Hirsch	Added Decrypt Transform section, added All-Attachments-Complete transform, changed MIME reference to v3, minor editorial changes.
5	07/07/04	Frederick Hirsch	Removed Decrypt transform material, since it is generally not needed and the approach had issues. Reorganized signatures section. Eliminated incorrect All-Attachments-Complete transform and replaced with discussion of attachment insertion threat. Clarified that only one wsse:Security header per actor/role minimizes signing, encryption confusion possibility. Added section for MIME Part CipherReference Transform. Editorial fixes.
6	07/14/04	Frederick Hirsch	<p>** Allow use of Content-Location, consistent with SwA.</p> <p>** Proposed update to signature Content-Transfer-Encoding processing rules. Needs review.</p> <p>Revised section on MIME canonicalization, added section on XML attachments. Only support SOAP 1.1. Clarified introduction. Added MTOM and additional MIME references. (Issue 297 should be closed – removed section on decryption transform and updated section on signing and encryption in version 5) Issue 303 – fixed, (see 3.2.4 example), Issue 306 – revised section on MIME canonicalization to close this issue. Issue 307 – revised to refer to SOAP 1.1 only, added section on XML attachments, defined MTOM and added reference. Editorial fixes.</p>

Rev	Date	By Whom	What
7	07/30/04	Frederick Hirsch	Incorporate feedback from WS-I BSP. Limit MIME headers included in signature or encryption to those listed in profile. Clarify MIME layering approach. Remove processing rules associated with Content-Transfer-Encoding. Editorial correction throughout document to allow both CID and Content-Location references to attachments. Editorial revision to pull attachment referencing and reference transforms into section applicable to both signatures and encryption. Incorporated feedback from Pete Wenzel and Toshihiro Nishimura – separate URL for transform and encryption type, used Content-Only reference transform for Cipherdata as well.
8	08/23/04	Frederick Hirsch	Address issue 312 by clarifying use of Reference within EncryptedData element to EncryptedData for attachment when EncryptedKey is used. Processing rule related to encryption of both attachment and primary SOAP envelope items. (http://www.oasis-open.org/archives/wss/200408/msg00046.html) Changed encryption example to show encryption of both primary SOAP envelop body and attachment. Include EncryptionMethod, addressing issue 309. Fix Transforms namespace to be xenc for within xenc:CipherReference (http://www.oasis-open.org/archives/wss/200408/msg00048.html)
9	09/02/04	Frederick Hirsch	Clarify that XML attachments are opaque and remove text about XML canonicalization of attachment content. Fix typo at line 356, should state that no KeyInfo should be in EncryptedData element when EncryptedKey is used. Clarify that cipher data is base64 encoded octet stream and require CipherReference base64 transform. Revise MIME headers to be included in Attachment-Complete Reference, for signature protection. Allow continuations for these MIME headers.

Rev	Date	By Whom	What
10	10/02/04	Frederick Hirsch	<p>Proposed resolutions for WSS issue-list items:</p> <p>Issue 326 part 1 – corrected case of Content-ID throughout document.</p> <p>Issue 326 part 2 - : Clarify MIME header name case, Resolution to use case per MIME specifications. See 4.3.1 item 4.</p> <p>Issue 326 part 3- Clarify transform handling of MIME parameter quoting. Retain quoting, if any, as is. Resolution in 4.3.1 item 7.</p> <p>Issue 326 part 4 - Address RFC 2047 encoding. Require transform to perform RFC2047 decoding as needed. Resolution in 4.3.1, items 4-7.</p> <p>Issue 329 part 1 – Strip or compress white space. No change made apart apart from preserve all whitespace in quoted strings, 4.3.1. item 10.</p> <p>Issue 329 part 2 – Order header processing alphabetically. Resolution in 4.3.1 item 3 and 4.2.2.</p> <p>Issue 329 part 3 – Show all ds:Signature elements in example in 4.3.6.</p>
11	10/02/04	Frederick Hirsch	Issue 326, 329 – revision of section 4.3.1 based on feedback from Dana Kaufman and Forum Systems.
12	10/21/04	Frederick Hirsch	<p>Allow cipher data to be binary data, and not use base64 transform in this case. Clarify that for base64 encoded cipher data transform or other means should be used to convey this information. Updated 4.4.1 through 4.4.4.</p> <p>Quoted “text/xml” in examples in 4.3.6, 4.4.4 to resolve issue 325.</p>
13	10/29/04	Frederick Hirsch	<p>Replace “7-bit” with “binary” in example 4.4.4</p> <p>Add clarification to sections 4.2.1 and 4.2.2 that MIME canonicalization is to be associated with the transforms, as defined in 4.3.1 and 4.3.2.</p>
14	11/15/04	Frederick Hirsch	<ol style="list-style-type: none"> 1. Only allow CID references for WS-Security references, for simplicity and interoperability. 2. Constrain statement on RFC2047 encoding in section 4.4.1, #6. 3. Clarify use of <xenc:EncryptedData> MimeType attribute in 4.5.2, #4. 4. Add statement from interop document regarding MIME boundary for primary SOAP envelope 5. Editorial changes to make MAY/MUST/SHOULDs capitalized where possible, other editorial fixes.

651

C. Notices

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

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

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

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

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

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

678 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
679 other countries.