



1

2

3

4

Web Services Security Rights Expression Language (REL) Token Profile

5

Committee Draft: 07 September 2004

6

Document Identifier:

7

urn:oasis:names:tc:WSS:1.0:profiles:REL

8

Document Location:

9

<http://docs.oasis-open.org/wss/2004/###oasis-####-wss-REL-token-profile-1.0>

10

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

11

Errata Location:

12

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

13

Editors:

Thomas	DeMartini	ContentGuard, Inc.
Anthony	Nadalin	IBM
Chris	Kaler	Microsoft Corporation
Ronald	Monzillo	Sun Microsystems
Phillip	Hallam-Baker	Verisign

14

Contributors:

15

Current voting members of the WSS TC (as of 07 September 2004)

Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Laboratory
Hal	Lockhart	BEA Systems, Inc.
Corinna	Witt	BEA Systems, Inc.
Merlin	Hughes	Betrusted (Baltimore Technologies)
Davanum	Srinivas	Computer Associates
Thomas	DeMartini	ContentGuard, Inc.
Guillermo	Lao	ContentGuard, Inc.
Sam	Wei	Documentum
Tim	Moses	Entrust
Dana	Kaufman	Forum Systems, Inc.
Toshihiro	Nishimura	Fujitsu

Kefeng	Chen	GeoTrust
Irving	Reid	Hewlett-Packard
Kojiro	Nakayama	Hitachi
Paula	Austel	IBM
Derek	Fu	IBM
Maryann	Hondo	IBM
Kelvin	Lawrence	IBM
Michael	McIntosh	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Ron	Williams	IBM
Don	Flinn	Individual
Bob	Morgan	Internet2
Kate	Cherry	Lockheed Martin
Paul	Cotton	Microsoft Corporation
Vijay	Gajjala	Microsoft Corporation
Alan	Geller	Microsoft Corporation
Chris	Kaler	Microsoft Corporation
Richard	Levinson	Netegrity, Inc.
Prateek	Mishra	Netegrity, Inc.
Frederick	Hirsch	Nokia
Senthil	Sengodan	Nokia
Abbie	Barbir	Nortel Networks
Lloyd	Burch	Novell
Charles	Knouse	Oblix
Steve	Anderson	OpenNetwork
Vamsi	Motukuru	Oracle
Ramana	Turlapati	Oracle
Ben	Hammond	RSA Security
Andrew	Nash	RSA Security
Rob	Philpott	RSA Security
Martijn	de Boer	SAP
Blake	Dournaee	Sarvega
Coumara	Radja	Sarvega
Pete	Wenzel	SeeBeyond Technology Corporation
Jeff	Hodges	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Symon	Chang	Tibco
John	Weiland	US Dept of the Navy
Phillip	Hallam-Baker	Verisign
Maneesh	Sahu	Westbridge Technology

17 **Contributors of input documents (if not already listed above):**

TJ	Pannu	ContentGuard, Inc.
Xin	Wang	ContentGuard, Inc.
Hiroshi	Maruyama	IBM
John	Shewchuk	Microsoft Corporation
Hemma	Prafullchandra	Verisign

18 **Abstract:**
19 This document describes how to use ISO/IEC 21000-5 Rights Expressions with the Web
20 Services Security: SOAP Message Security [WS-Security] specification.

21 **Status:**
22 This is a Committee Draft. Please send comments to the editors.
23 Committee members should send comments on this specification to the
24 <mailto:wss@lists.oasis-open.org> list. Others should subscribe to and send comments to
25 the wss-comment@lists.oasis-open.org list. To subscribe, visit
26 <http://lists.oasis-open.org/ob/adm.pl>.
27 For information on whether any patents have been disclosed that may be essential to
28 implementing this specification, and any offers of patent licensing terms, please refer to
29 the Intellectual Property Rights section of the Web Services Security TC web page
30 (<http://www.oasis-open.org/committees/wss/ipr.php>).

31 **Table of Contents**

32 1 Introduction (Informative)..... 5

33 2 Notations and Terminology (Normative)..... 6

34 2.1 Notational Conventions 6

35 2.2 Namespaces 6

36 2.3 Terminology..... 7

37 3 Usage (Normative)..... 8

38 3.1 Token Types..... 8

39 3.2 Processing Model..... 8

40 3.3 Attaching Security Tokens 8

41 3.4 Identifying and Referencing Security Tokens 8

42 3.5 Authentication..... 11

43 3.5.1 <r:keyHolder> Principal..... 12

44 3.6 Confidentiality..... 14

45 3.6.1 <r:keyHolder> Principal..... 15

46 3.7 Error Codes 16

47 4 Types of Licenses (Informative)..... 17

48 4.1 Attribute Licenses..... 17

49 4.2 Sender Authorization..... 18

50 4.3 Issuer Authorization 18

51 5 Threat Model and Countermeasures (Informative)..... 21

52 5.1 Eavesdropping 21

53 5.2 Replay 21

54 5.3 Message Insertion 22

55 5.4 Message Deletion..... 22

56 5.5 Message Modification 22

57 5.6 Man-in-the-Middle 22

58 6 References..... 23

59 Appendix A: Revision History 24

60 Appendix B: Notices 25

61

62 **1 Introduction (Informative)**

63 The Web Services Security: SOAP Message Security [WS-Security] specification proposes a
64 standard set of SOAP extensions that can be used when building secure Web services to
65 implement message level integrity and confidentiality. This specification describes the use of
66 ISO/IEC 21000-5 Rights Expressions with respect to the WS-Security specification.

67 2 Notations and Terminology (Normative)

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

69 2.1 Notational Conventions

70 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
71 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be
72 interpreted as described in [KEYWORDS].

73 Namespace URIs (of the general form "some-URI") represent some application-dependent or
74 context-dependent URI as defined in [URI].

75 This specification is designed to work with the general SOAP message structure and message
76 processing model, and should be applicable to any version of SOAP. The current SOAP 1.2
77 namespace URI is used herein to provide detailed examples, but there is no intention to limit the
78 applicability of this specification to a single version of SOAP.

79 2.2 Namespaces

80 The XML namespace [XML-ns] URIs that MUST be used by implementations of this specification
81 are as follows (note that different elements in this specification are from different namespaces):

```
82     http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
83     wssecurity-secext-1.0.xsd  
84     http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
85     wssecurity-utility-1.0.xsd  
86     urn:mpeg:mpeg21:2003:01-REL-R-NS
```

87 The following namespaces are used in this document:

88

Prefix	Namespace
S	http://www.w3.org/2001/12/soap-envelope
ds	http://www.w3.org/2000/09/xmldsig#
xenc	http://www.w3.org/2001/04/xmlenc#
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

r	urn:mpeg:mpeg21:2003:01-REL-R-NS
sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS

89

Table 1 Namespace Prefixes

90

91 **2.3 Terminology**

92 This specification employs the terminology defined in the Web Services Security: SOAP Message
93 Security [WS-Security] Specification.

94 Defined below are the basic definitions for additional terminology used in this specification.

95 **License** – ISO/IEC 21000-5 Rights Expression

96 3 Usage (Normative)

97 This section describes the syntax and processing rules for the use of licenses with
98 the Web Services Security: Soap Message Security specification [WS-Security].

99 3.1 Token Types

100 When a URI value is used to indicate a license according to this profile, its value MUST be
101 <http://docs.oasis-open.org/wss/2004/###oasis-#####-wss-REL-token-profile-1.0#license>.

102 3.2 Processing Model

103 The processing model for WS-Security with licenses is no different from that of WS-
104 Security with other token formats as described in Web Services Security: SOAP Message
105 Security [WS-Security].

106 At the token level, a processor of licenses MUST conform to the required validation
107 and processing rules defined in ISO/IEC 21000-5 [REL].

108 3.3 Attaching Security Tokens

109 Licenses are attached to SOAP messages using WS-Security by placing the license
110 element inside the `<wsse:Security>` header. The following example illustrates a
111 SOAP message with a license.

```
112 <S:Envelope xmlns:S="...">  
113   <S:Header>  
114     <wsse:Security xmlns:wsse="...">  
115       <r:license xmlns:r="...">  
116         ...  
117       </r:license>  
118       ...  
119     </wsse:Security>  
120   </S:Header>  
121   <S:Body>  
122     ...  
123   </S:Body>  
124 </S:Envelope>
```

125 3.4 Identifying and Referencing Security Tokens

126 The Web Services Security: SOAP Message Security [WS-Security] specification defines the
127 `wsu:id` attribute as the common mechanism for identifying security tokens (the specification
128 describes the reasons for this). Licenses have an additional identification mechanism available:
129 their `licenseId` attribute, the value of which is a URI. The following example shows a license that
130 uses both mechanisms:

131
132
133
134
135

```
<r:license xmlns:r="..." xmlns:wssu="..."
  licenseId="urn:foo:SecurityToken:ef375268"
  wssu:Id="SecurityToken-ef375268">
  ...
</r:license>
```

136 Licenses can be referenced either according to their location or their licenseld. Location
137 references are dependent on location and can be either local or remote. Licenseld references
138 are not dependent on location.

139 Local location references are RECOMMENDED when they can be used. Remote location
140 references are OPTIONAL for cases where it is not feasible to transmit licenses with the SOAP
141 message. Licenseld references are OPTIONAL for cases where location is unknown or cannot
142 be indicated.

143 WS-Security specifies that tokens are referenced using the <wsse:SecurityTokenReference>
144 element.

145 Implementations compliant with this profile SHOULD set the
146 /wsse:SecurityTokenReference/wsse:Reference/@ValueType attribute to http://docs.oasis-
147 open.org/wss/2004/##/oasis-####-wss-REL-token-profile-1.0#license when using
148 wsse:SecurityTokenReference to refer to a license by licenseld. This is OPTIONAL when
149 referring to a license by location.

150 The following table demonstrates the use of the <wsse:SecurityTokenReference> element to
151 refer to licenses.

By Location	Local	<pre><wsse:SecurityTokenReference> <wsse:Reference URI="#SecurityToken-ef375268" /> </wsse:SecurityTokenReference></pre>
	Remote	<pre><wsse:SecurityTokenReference> <wsse:Reference URI="http://www.foo.com/ef375268.xml" /> </wsse:SecurityTokenReference></pre>
By licenseld		<pre><wsse:SecurityTokenReference> <wsse:Reference URI="urn:foo:SecurityToken:ef375268" ValueType="http://docs.oasis- open.org/wss/2004/##/oasis-####-wss-REL-token- profile-1.0#license" /> </wsse:SecurityTokenReference></pre>

152

Table 2. <wsse:SecurityTokenReference>

153 The following example demonstrates how a <wsse:SecurityTokenReference> can be used to
154 indicate that the message parts specified inside the <ds:SignedInfo> element were signed using
155 a key from the license referenced by licenseld in the <ds:KeyInfo> element.

156
157

```
<S:Envelope xmlns:S="...">
  <S:Header>
```

```

158     <wsse:Security xmlns:wsse="...">
159         <r:license xmlns:r="..."
160 licenseId="urn:foo:SecurityToken:ef375268" xmlns:wsu="..."
161 wsu:Id="SecurityToken-ef375268">
162             ...
163         </r:license>
164         ...
165         <ds:Signature>
166             <ds:SignedInfo>
167                 ...
168             </ds:SignedInfo>
169             <ds:SignatureValue>...</ds:SignatureValue>
170             <ds:KeyInfo>
171                 <wsse:SecurityTokenReference>
172                     <wsse:Reference
173                         URI="#SecurityToken-ef375268"
174                     />
175                 </wsse:SecurityTokenReference>
176             </ds:KeyInfo>
177         </ds:Signature>
178     </wsse:Security>
179 </S:Header>
180 <S:Body>
181     ...
182 </S:Body>
183 </S:Envelope>

```

184 The following example shows a signature over a local license using a location reference to that
185 license. The example demonstrates how the integrity of an (unsigned) license can be preserved
186 by signing it in the <wsse:Security> header.

```

187 <S:Envelope xmlns:S="...">
188     <S:Header>
189         <wsse:Security xmlns:wsse="..."
190             <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-
191 ef375268">
192             ...
193         </r:license>
194         ...
195         <wsse:SecurityTokenReference wsu:Id="Str1">
196             <wsse:Reference
197                 URI="#SecurityToken-ef375268"
198             />
199         </wsse:SecurityTokenReference>
200         ...
201         <ds:Signature>
202             <ds:SignedInfo>
203                 ...
204                 <Reference URI="#Str1">
205                     <Transforms>
206                         <ds:Transform
207                             Algorithm="http://schemas.xmlsoap.org/2003/06/STR-
208 Transform">
209                             <ds:CanonicalizationMethod
210                                 Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
211 20010315"/>
212                         </ds:Transform>

```

```

213         </ds:Transforms>
214         <ds:DigestMethod
215             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
216         />
217         <ds:DigestValue>...</ds:DigestValue>
218     </ds:Reference>
219 </ds:SignedInfo>
220 <ds:SignatureValue>...</ds:SignatureValue>
221 <ds:KeyInfo>...</ds:KeyInfo>
222 </ds:Signature>
223 </wsse:Security>
224 </S:Header>
225 <S:Body>
226     ...
227 </S:Body>
228 </S:Envelope>

```

229 Note: since licenses allow the use of the wsu:Id attribute, it is usually not necessary to use the
230 STR-Transform because the license can be referred to directly in the ds:SignedInfo as shown in
231 the following example:

```

232 <S:Envelope xmlns:S="...">
233   <S:Header>
234     <wsse:Security xmlns:wsse="...">
235       <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-
236 ef375268">
237         ...
238       </r:license>
239       ...
240       <ds:Signature>
241         <ds:SignedInfo>
242           ...
243           <ds:Reference URI="#SecurityToken-ef375268">
244             <ds:DigestMethod
245                 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
246             />
247             <ds:DigestValue>...</ds:DigestValue>
248           </ds:Reference>
249         </ds:SignedInfo>
250         <ds:SignatureValue>...</ds:SignatureValue>
251         <ds:KeyInfo>...</ds:KeyInfo>
252       </ds:Signature>
253     </wsse:Security>
254   </S:Header>
255   <S:Body>
256     ...
257 </S:Body>
258 </S:Envelope>

```

259 3.5 Authentication

260 The Web Services Security: SOAP Message Security [WS-Security] specification does not dictate
261 how claim confirmation must be performed. As well, the REL allows for multiple types of
262 confirmation. This profile of WS-Security REQUIRES that message senders and receivers
263 support claim confirmation for <r:keyHolder> principals. It is RECOMMENDED that an XML

264 Signature be used to establish the relationship between the message sender and the claims. This
265 is especially RECOMMENDED whenever the SOAP message exchange is conducted over an
266 unprotected transport.

267 The following table enumerates the mandatory principals to be supported by claim confirmation
268 and summarizes their associated processing models. It should be noted that this table is not all-
269 encompassing, and it is envisioned that future specifications may expand this table over time.

Principal	RECOMMENDED Processing Rules
<r:keyHolder>	The message sender adds (to the security header) an XML Signature that can be verified with the key information specified in the <r:keyHolder> of the referenced license.

270 **Table 3. Processing Rules for Claim Confirmation**

271 Note that the high-level processing model described in the following sections does not
272 differentiate between message author and message sender as would be necessary to guard
273 against replay attacks. The high-level processing model also does not take into account
274 requirements for authentication of receiver by sender or for message or token confidentiality.
275 These concerns must be addressed by means other than those described in the high-level
276 processing model. If confidentiality of the token in the message is important, then use the
277 approach defined by [WS-Security] to encrypt the token.

278 **3.5.1 <r:keyHolder> Principal**

279 The following sections describe the <r:keyHolder> method of establishing the correspondence
280 between a SOAP message sender and the claims within a license.

281 **Sender**

282 The message sender MUST include within the <wsse:Security> header element a <r:license>
283 containing at least one <r:grant> to an <r:keyHolder> identifying the key to be used to confirm the
284 claims. If the message sender includes an <r:license> containing more than one <r:grant> to an
285 <r:keyHolder>, then all of those <r:keyHolder> elements MUST be equal.

286 In order for the receiver to perform claim confirmation, the sender MUST demonstrate knowledge
287 of the confirmation key. The sender MAY accomplish this by using the confirmation key to sign
288 content from within the message and by including the resulting <ds:Signature> element in the
289 <wsse:Security> header element. <ds:Signature> elements produced for this purpose MUST
290 conform to the canonicalization and token inclusion rules defined in the core WS-Security
291 specification and this profile specification.

292 Licenses that contain at least one <r:grant> to an <r:keyHolder> SHOULD contain an <r:issuer>
293 with a <ds:Signature> element that identifies the license issuer to the relying party and protects
294 the integrity of the confirmation key established by the license issuer.

295 Receiver

296 If the receiver determines that the sender has demonstrated knowledge of a confirmation key as
297 specified in an <r:keyHolder>, then the claims (found in the licenses) pertaining to that
298 <r:keyHolder> MAY be attributed to the sender. If one of these claims is an identity and if the
299 conditions of that claim are satisfied, then any elements of the message whose integrity is
300 protected by the confirmation key MAY be considered to have been authored by that identity.

301 Example

302 The following example illustrates how a license security token having an <r:keyHolder> principal
303 can be used with a <ds:Signature> to establish that John Doe is requesting a stock report on
304 FOO.

```
305 <S:Envelope xmlns:S="...">
306   <S:Header>
307     <wsse:Security xmlns:wsse="...">
308       <r:license xmlns:r="..."
309 licenseId="urn:foo:SecurityToken:ef375268">
310         <r:grant>
311           <r:keyHolder>
312             <r:info>
313               <ds:KeyValue>...</ds:KeyValue>
314             </r:info>
315           </r:keyHolder>
316           <r:possessProperty/>
317           <sx:commonName xmlns:sx="...">John Doe</sx:commonName>
318         </r:grant>
319         <r:issuer>
320           <ds:Signature>...</ds:Signature>
321         </r:issuer>
322       </r:license>
323     </S:Header>
324   <ds:Signature>
325     <ds:SignedInfo>
326       ...
327       <ds:Reference URI="#MsgBody">
328         <ds:DigestMethod
329           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
330         />
331         <ds:DigestValue>...</ds:DigestValue>
332       </ds:Reference>
333     </ds:SignedInfo>
334     <ds:SignatureValue>...</ds:SignatureValue>
335     <ds:KeyInfo>
336       <wsse:SecurityTokenReference>
337         <wsse:Reference
338           URI="urn:foo:SecurityToken:ef375268"
339           ValueType="http://docs.oasis-open.org/wss/2004/##/oasis-
340 #####-wss-REL-token-profile-1.0#license"
341         />
342       </wsse:SecurityTokenReference>
343     </ds:KeyInfo>
344   </ds:Signature>
```

346
347
348
349
350
351
352
353
354
355
356
357

```
</ds:Signature>

</wsse:Security>
</S:Header>

<S:Body @wsu:Id="MsgBody" xmlns:wsu="...">
  <ReportRequest>
    <TickerSymbol>FOO</TickerSymbol>
  </ReportRequest>
</S:Body>

</S:Envelope>
```

358 3.6 Confidentiality

359 This section details how licenses may be used to protect the confidentiality of a SOAP message
360 within WS-Security. The Web Services Security: SOAP Message Security [WS-Security]
361 specification does not dictate how confidentiality must be performed. As well, the REL allows for
362 multiple types of confidentiality. This profile of WS-Security REQUIRES that message senders
363 and receivers support confidentiality for <r:keyHolder> principals. It is RECOMMENDED that
364 XML Encryption be used to ensure confidentiality. This is especially RECOMMENDED whenever
365 the SOAP message exchange is conducted over an unprotected transport.

366 The following table enumerates the mandatory principals to be supported for confidentiality and
367 summarizes their associated processing models. It should be noted that this table is not all-
368 encompassing, and it is envisioned that future specifications may expand this table over time.

Principal	RECOMMENDED Processing Rules
<r:keyHolder>	The message sender adds (to the security header) either 1) an <xenc:ReferenceList> that points to one or more <xenc:EncryptedData> elements that can be decrypted with a key which can be determined from information specified in the <r:keyHolder> of the referenced license or 2) an <xenc:EncryptedKey> that can be decrypted with a key determined from information specified in the <r:keyHolder> of the referenced license.

369 **Table 4. Processing Rules for Confidentiality**

370 Note that this section deals only with Confidentiality. Details of authentication of the sender by
371 the receiver must be addressed by means other than those described in this section (see the
372 previous section).

373 3.6.1 <r:keyHolder> Principal

374 The following sections describe the <r:keyHolder> method of establishing confidentiality using a
375 license.

376 Sender

377 The message sender MUST include within the <wsse:Security> header element a <r:license>
378 containing at least one <r:grant> to an <r:keyHolder> identifying the key used to encrypt some
379 data or key. If the message sender includes an <r:license> containing more than one <r:grant> to
380 an <r:keyHolder>, then all of those <r:keyHolder> elements MUST be equal.

381 In order for the receiver to know when to decrypt the data or key, the sender MUST indicate the
382 encryption in the message. The sender MAY accomplish this by placing an
383 <xenc:EncryptedData> or <xenc:EncryptedKey> in the appropriate place in the message and by
384 including the resulting <xenc:ReferenceList> or <xenc:EncryptedKey> element in the
385 <wsse:Security> header element. <xenc:ReferenceList> or <xenc:EncryptedKey> elements
386 produced for this purpose MUST conform to the rules defined in the core WS-Security
387 specification and this profile specification.

388 Receiver

389 If the receiver determines that he has knowledge of a decryption key as specified in an
390 <r:keyHolder>, then he MAY decrypt the associated data or key. In the case of decrypting a key,
391 he may then recursively decrypt any data or key that that key can decrypt.

392

393 Example

394 The following example illustrates how a license containing a <r:keyHolder> principal can be used
395 with XML encryption schema elements to protect the confidentiality of a message using a
396 separate encryption key given in the <xenc:EncryptedKey> in the security header.

397 In this example, the r:license element provides information about the recipient's RSA public key
398 (i.e., KeyValue in keyHolder) used to encrypt the symmetric key carried in the EncryptedKey
399 element. The recipient uses this information to determine the correct private key to use in
400 decrypting the symmetric key. The symmetric key is then used to decrypt the EncryptedData child
401 of the Body element.

402

```
403 <S:Envelope xmlns:S="...">  
404   <S:Header>  
405     <wsse:Security xmlns:wsse="...">  
406       <r:license xmlns:r="..."  
407         licenseId="urn:foo:SecurityToken:ef375268">  
408         <r:grant>  
409           <r:keyHolder>  
410             <r:info>  
411               <ds:KeyValue>...</ds:KeyValue>  
412             </r:info>  
413           </r:keyHolder>
```

```

414     <r:possessProperty/>
415     <sx:commonName xmlns:sx="...">SOME COMPANY</sx:commonName>
416 </r:grant>
417 <r:issuer>
418     <ds:Signature>...</ds:Signature>
419 </r:issuer>
420 </r:license>
421 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
422     <xenc:EncryptionMethod
423         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
424     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
425         <wsse:SecurityTokenReference>
426             <wsse:Reference URI="urn:foo:SecurityToken:ef375268"/>
427         </wsse:SecurityTokenReference>
428     </KeyInfo>
429     <xenc:CipherData>
430         <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
431     </xenc:CipherData>
432     <xenc:ReferenceList>
433         <xenc:DataReference URI="#enc"/>
434     </xenc:ReferenceList>
435 </xenc:EncryptedKey>
436 </wsse:Security>
437 </S:Header>
438 <S:Body wsu:Id="body"
439     xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
440     <xenc:EncryptedData Id="enc"
441         Type="http://www.w3.org/2001/04/xmlenc#Content"
442         xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
443         <xenc:EncryptionMethod
444             Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
445         <xenc:CipherData>
446             <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
447         </xenc:CipherData>
448         </xenc:EncryptedData>
449     </S:Body>
450 </S:Envelope>

```

451 3.7 Error Codes

452 It is RECOMMENDED that the error codes defined in the Web Services Security:
453 SOAP Message Security [WS-Security] specification are used. However,
454 implementations MAY use custom errors, defined in private namespaces if they
455 desire. Care should be taken not to introduce security vulnerabilities in the errors
456 returned.

457

4 Types of Licenses (Informative)

458

4.1 Attribute Licenses

459

In addition to key information, licenses can carry information about attributes of those keys.

460

Examples of such information on a client are e-mail address or common name. A service's key,

461

on the other hand, might be associated with a DNS name and common name.

462

The following is an example client attribute license.

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

```

<r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
  <r:inventory>
    <r:keyHolder licensePartId="client">
      <r:info>
        <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
  </r:inventory>
  <r:grant>
    <r:keyHolder licensePartIdRef="client"/>
    <r:possessProperty/>
    <sx:commonName>John Doe</sx:commonName>
  </r:grant>
  <r:grant>
    <r:keyHolder licensePartIdRef="client"/>
    <r:possessProperty/>
    <sx:emailName>jd@foo.com</sx:emailName>
  </r:grant>
  <r:issuer>
    <ds:Signature>...</ds:Signature>
  </r:issuer>
</r:license>

```

485

The following is an example service attribute license.

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

```

<r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
  <r:inventory>
    <r:keyHolder licensePartId="service">
      <r:info>
        <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
  </r:inventory>
  <r:grant>
    <r:keyHolder licensePartIdRef="service"/>
    <r:possessProperty/>
    <sx:commonName>MyService Company</sx:commonName>
  </r:grant>
  <r:grant>
    <r:keyHolder licensePartIdRef="service"/>
    <r:possessProperty/>
    <sx:dnsName>www.myservice.com</sx:dnsName>
  </r:grant>
  <r:issuer>
    <ds:Signature>...</ds:Signature>
  </r:issuer>
</r:license>

```

508 Additional examples of and processing rules for the use of attribute licenses can be found in the
509 above sections on Authentication and Confidentiality.

510 4.2 Sender Authorization

511 Licenses may be used by a sender as proof of authorization to perform a certain action on a
512 particular resource. This WS-Security specification does not describe how authorization must be
513 performed. In the context of web services, a sender can send to a receiver an authorization
514 license in the security header as proof of authorization to call the sender. Typically, this
515 authorization license is signed by a trusted authority and conforms to the syntax pattern specified
516 below.

```
517 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">  
518   <r:grant>  
519     <r:keyHolder>  
520       <r:info>  
521         <ds:KeyValue>FDFEWEFF...</ds:KeyValue>  
522       </r:info>  
523     </r:keyHolder>  
524     <sx:rightUri definition='...'/>  
525     <x:someResource/>  
526     <x:someCondition/>  
527   </r:grant>  
528   <r:issuer>  
529     <ds:Signature>...</ds:Signature>  
530   </r:issuer>  
531 </r:license>
```

532 The above license contains an authorization grant authorizing the keyholder (sender's public
533 key), the right to exercise the right identified in the <sx:rightUri> element. The resource in the
534 license typically corresponds to the semantics of the URI given in the definition attribute of the
535 <sx:rightUri> element. The entire license along with the <ds:Signature> element in the <r:issuer>
536 certifies the fact that the principal (<keyholder>) is granted the authorization to exercise the right
537 in the <sx:rightUri> element over the specified resource. The integrity of the license is usually
538 protected with a digital signature contained within the <ds:Signature>.

539 4.3 Issuer Authorization

540 To enunciate that a particular issuer is allowed to issue particular types of licenses, one can use
541 the kind of license described here. Issuer authorization licenses can accompany other licenses in
542 the security header such as those used for authentication, sender authorization, or other issuer
543 authorizations. These issuer authorization licenses might help complete the authorization proof
544 that is required for authorizing or authenticating a particular sender.

545

546 The following license is an example issuer authorization license for authorizing an issuer to issue
547 a simple attribute license.

```
548 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">  
549   <r:grant>  
550     <r:forAll varName='K' />  
551     <r:forAll varName='P' />  
552     <r:keyHolder>  
553       <r:info>  
554         <ds:KeyValue>FDFEWEFF...</ds:KeyValue>  
555       </r:info>
```

```

556     </r:keyHolder>
557     <r:issue/>
558     <r:grant>
559         <r:keyHolder varRef='K' />
560         <r:possessProperty/>
561         <r:propertyAbstract varRef='P' />
562     </r:grant>
563 </r:grant>
564 <r:issuer>
565     <ds:Signature>...</ds:Signature>
566 </r:issuer>
567 </r:license>

```

568 The following license is an example issuer authorization license for authorizing an issuer to issue
569 sender authorization licenses.

```

570 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
571     <r:grant>
572         <r:forAll varName='K' />
573         <r:forAll varName='R' />
574         <r:keyHolder>
575             <r:info>
576                 <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
577             </r:info>
578         </r:keyHolder>
579         <r:issue/>
580         <r:grant>
581             <r:keyHolder varRef='K' />
582             <sx:rightUri definition='...'/>
583             <r:resource varRef='R' />
584         </r:grant>
585     </r:grant>
586     <r:issuer>
587         <ds:Signature>...</ds:Signature>
588     </r:issuer>
589 </r:license>

```

590 The following license is an example issuer authorization license for authorizing an issuer to issue
591 (to other issuers) issuer authorization licenses allowing those other issuers to issue simple
592 attribute licenses, such as those that can be used for authentication or confidentiality.

```

593 <r:license xmlns:r="..."licenseId="urn:foo:SecurityToken:ef375268">
594     <r:grant>
595         <r:forAll varName='I' />
596         <r:keyHolder>
597             <r:info>
598                 <ds:KeyValue>FDFEWEFF...</ds:KeyValue>
599             </r:info>
600         </r:keyHolder>
601         <r:issue/>
602         <r:grant>
603             <r:forAll varName='K' />
604             <r:forAll varName='P' />
605             <r:keyHolder varRef='I' />
606             <r:issue/>
607             <r:grant>
608                 <r:keyHolder varRef='K' />
609                 <r:possessProperty/>
610                 <r:propertyAbstract varRef='P' />
611             </r:grant>
612         </r:grant>
613     </r:grant>
614     <r:issuer>
615         <ds:Signature>...</ds:Signature>
616     </r:issuer>

```

617

</r:license>

618

619

5 Threat Model and Countermeasures (Informative)

620

621 This section addresses the potential threats that a SOAP message may encounter and the
622 countermeasures that may be taken to thwart such threats. A SOAP message containing licenses
623 may face threats in various contexts. This includes the cases where the message is in transit,
624 being routed through a number of intermediaries, or during the period when the message is in
625 storage.

626 The use of licenses with WS-Security introduces no new threats beyond those identified for the
627 REL or WS-Security with other types of security tokens. Message alteration and eavesdropping
628 can be addressed by using the integrity and confidentiality mechanisms described in WS-
629 Security. Replay attacks can be addressed by using of message timestamps and caching, as well
630 as other application-specific tracking mechanisms. For licenses, ownership is verified by the use
631 of keys; man-in-the-middle attacks are generally mitigated. It is strongly RECOMMENDED that all
632 relevant and immutable message data be signed. It should be noted that transport-level security
633 MAY be used to protect the message and the security token. In order to trust licenses, they
634 SHOULD be signed natively and/or using the mechanisms outlined in WS-Security. This allows
635 readers of the licenses to be certain that the licenses have not been forged or altered in any way.
636 It is strongly RECOMMENDED that the <r:license> elements be signed (either within the token,
637 as part of the message, or both).

638 The following few sections elaborate on the afore-mentioned threats and suggest
639 countermeasures.

640

5.1 Eavesdropping

641 Eavesdropping is a threat to the confidentiality of the message, and is common to all types of
642 network protocols. The routing of SOAP messages through intermediaries increases the potential
643 incidences of eavesdropping. Additional opportunities for eavesdropping exist when SOAP
644 messages are persisted.

645 To provide maximum protection from eavesdropping, licenses, license references, and sensitive
646 message content SHOULD be encrypted such that only the intended audiences can view their
647 content. This removes threats of eavesdropping in transit, but does not remove risks associated
648 with storage or poor handling by the receiver.

649 Transport-layer security MAY be used to protect the message from eavesdropping while in
650 transport, but message content must be encrypted above the transport if it is to be protected from
651 eavesdropping by intermediaries.

652

5.2 Replay

653 The reliance on authority protected (e.g. signed) licenses to <r:keyHolder> principals precludes
654 all but the key holder from binding the licenses to a SOAP message. Although this mechanism

655 effectively restricts message authorship to the holder of the confirmation key, it does not preclude
656 the capture and resubmission of the message by other parties.

657 Replay attacks can be addressed by using message timestamps and caching, as well as other
658 application-specific tracking mechanisms.

659 **5.3 Message Insertion**

660 This profile of WS-Security is not vulnerable to message insertion attacks. Higher-level protocols
661 built on top of SOAP and WS-Security should avoid introducing message insertion threats and
662 provide proper countermeasures for any they do introduce.

663 **5.4 Message Deletion**

664 This profile of WS-Security is not vulnerable to message deletion attacks other than denial of
665 service. Higher-level protocols built on top of SOAP and WS-Security should avoid introducing
666 message deletion threats and provide proper countermeasures for any they do introduce.

667 **5.5 Message Modification**

668 Message Modification poses a threat to the integrity of a message. The threat of message
669 modification can be thwarted by signing the relevant and immutable content by the key holder.
670 The receivers SHOULD only trust the integrity of those segments of the message that are signed
671 by the key holder.

672 To ensure that message receivers can have confidence that received licenses have not been
673 forged or altered since their issuance, licenses appearing in <wsse:Security> header elements
674 SHOULD be integrity protected (e.g. signed) by their issuing authority. It is strongly
675 RECOMMENDED that a message sender sign any <r:license> elements that it is confirming and
676 that are not signed by their issuing authority.

677 Transport-layer security MAY be used to protect the message and contained licenses and/or
678 license references from modification while in transport, but signatures are required to extend such
679 protection through intermediaries.

680 **5.6 Man-in-the-Middle**

681 This profile of WS-Security is not vulnerable to man-in-the-middle attacks. Higher-level protocols
682 built on top of SOAP and WS-Security should avoid introducing Man-in-the-Middle threats and
683 provide proper countermeasures for any they do introduce.

684

685

6 References

- 686 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels,"
687 RFC 2119, Harvard University, March 1997,
688 <http://www.ietf.org/rfc/rfc2119.txt>
- 689 **[REL]** ISO/IEC 21000-5:2004, "Information technology -- Multimedia framework
690 (MPEG-21) -- Part 5: Rights Expression Language,"
691 [http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUM
692 BER=36095&ICS1=35&ICS2=40&ICS3=](http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36095&ICS1=35&ICS2=40&ICS3=)
- 693 **[SOAP]** D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.
694 Frystyk Nielsen, S Thatte, D. Winer. Simple Object Access Protocol
695 (SOAP) 1.1, W3C Note 08 May 2000, <http://www.w3.org/TR/SOAP/>
- 696 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers
697 (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox
698 Corporation, August 1998, <http://www.ietf.org/rfc/rfc2396.txt>
- 699 **[WS-Security]** OASIS Standard 200401, "Web Services Security: Soap Message
700 Security 1.0 (WS-Security 2004)," March 2004, [http://docs.oasis-
701 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)
- 702 **[XML-ns]** T. Bray, D. Hollander, A. Layman. Namespaces in XML. W3C
703 Recommendation. January 1999, [http://www.w3.org/TR/1999/REC-xml-
704 names-19990114](http://www.w3.org/TR/1999/REC-xml-names-19990114)
- 705 **[XML Signature]** D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. XML-
706 Signature Syntax and Processing, W3C Recommendation, 12 February
707 2002, <http://www.w3.org/TR/xmlsig-core/>
- 708

Appendix A: Revision History

Rev	Date	What
01	19-Sep-02	Initial draft produced by extracting SAML related content from [XML token]
02	12-Dec-02	Naming changes
03	30-Jan -03	Name changes, merged in comments from Thomas DeMartini
04	13-Nov-03	Updates, merged in comments from Thomas DeMartini
05	08-Jan-04	Contributor list updates, many title page updates, document name updates, namespace updates, switched from QNames to URIs.
06	29-Apr-04	Clarified case when a license contains more than one grant. Updated URIs.
07	24-May-04	Removed XrML from document name, per request from OASIS staff. Updated document to remove references to XrML.
08	18-Jun-04	Updated Contributor List
09	07-Sep-04	Updated contributor list and added link to errata

711 **Appendix B: Notices**

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

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

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

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

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

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

741