Web Services Security
Rights Expression Language (REL)
Token Profile 1.1

Committee Draft: 14 June 2005

Document Location:
http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.1.pdf

Errata Location:
http://www.oasis-open.org/committees/wss

Technical Committee:
Web Services Security (WSS)

Chairs:
Kelvin Lawrence, IBM
Chris Kaler, Microsoft

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

Abstract:
This document describes how to use ISO/IEC 21000-5 Rights Expressions with the Web Services Security (WSS) specification.

Status:
The status of this document is Committee Draft. Please send comments to the editors. If you are on the wss@lists.oasis-open.org list for committee members, send comments there. If you are not on that list, subscribe to the wss-comment@lists.oasis-open.org list
and send comments there. To subscribe, send an email message to wss-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For patent disclosure information that may be essential to the implementation of this specification, and any offers of licensing terms, refer to the Intellectual Property Rights section of the OASIS Web Services Security Technical Committee (WSS TC) web page at http://www.oasis-open.org/committees/wss/ipr.php. General OASIS IPR information can be found at http://www.oasis-open.org/who/intellectualproperty.shtml.
OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

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

Copyright © The Organization for the Advancement of Structured Information Standards [OASIS] 2002-2005. All Rights Reserved.

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

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

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction (Informative)</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Notations and Terminology (Normative)</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Notational Conventions</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Namespaces</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Terminology</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Usage (Normative)</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>Token Types</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>Processing Model</td>
<td>8</td>
</tr>
<tr>
<td>3.3</td>
<td>Attaching Security Tokens</td>
<td>8</td>
</tr>
<tr>
<td>3.4</td>
<td>Identifying and Referencing Security Tokens</td>
<td>8</td>
</tr>
<tr>
<td>3.5</td>
<td>Authentication</td>
<td>11</td>
</tr>
<tr>
<td>3.5.1</td>
<td>&lt;r:keyHolder&gt; Principal</td>
<td>12</td>
</tr>
<tr>
<td>3.6</td>
<td>Confidentiality</td>
<td>14</td>
</tr>
<tr>
<td>3.6.1</td>
<td>&lt;r:keyHolder&gt; Principal</td>
<td>15</td>
</tr>
<tr>
<td>3.7</td>
<td>Error Codes</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Types of Licenses (Informative)</td>
<td>17</td>
</tr>
<tr>
<td>4.1</td>
<td>Attribute Licenses</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Sender Authorization</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Issuer Authorization</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Threat Model and Countermeasures (Informative)</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Eavesdropping</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Replay</td>
<td>21</td>
</tr>
<tr>
<td>5.3</td>
<td>Message Insertion</td>
<td>22</td>
</tr>
<tr>
<td>5.4</td>
<td>Message Deletion</td>
<td>22</td>
</tr>
<tr>
<td>5.5</td>
<td>Message Modification</td>
<td>22</td>
</tr>
<tr>
<td>5.6</td>
<td>Man-in-the-Middle</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>References</td>
<td>23</td>
</tr>
<tr>
<td>94</td>
<td>Appendix A: Acknowledgements</td>
<td>24</td>
</tr>
<tr>
<td>95</td>
<td>Appendix B: Revision History</td>
<td>26</td>
</tr>
</tbody>
</table>
1 Introduction (Informative)

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

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

2.1 Notational Conventions

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

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

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

2.2 Namespaces

The following namespaces are used in this document:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td><a href="http://www.w3.org/2001/12/soap-envelope">http://www.w3.org/2001/12/soap-envelope</a></td>
</tr>
<tr>
<td>ds</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a></td>
</tr>
<tr>
<td>xenc</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#">http://www.w3.org/2001/04/xmlenc#</a></td>
</tr>
<tr>
<td>wsse</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a></td>
</tr>
<tr>
<td>wsu</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
</tr>
<tr>
<td>r</td>
<td>urn:mpeg:mpeg21:2003:01-REL-R-NS</td>
</tr>
</tbody>
</table>
### 2.3 Terminology


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

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

---

**Table 1 Namespace Prefixes**

| sx   | urn:mpeg:mpeg21:2003:01-REL-SX-NS |
3 Usage (Normative)

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

3.1 Token Types

When a URI value is used to indicate a license according to this profile, its value MUST be http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license.

3.2 Processing Model

The processing model for WS-Security with licenses is no different from that of WS-Security with other token formats as described in Web Services Security: SOAP Message Security [WS-Security]. At the token level, a processor of licenses MUST conform to the required validation and processing rules defined in ISO/IEC 21000-5 [REL].

3.3 Attaching Security Tokens

Licenses are attached to SOAP messages using WS-Security by placing the license element inside the \(<\text{wsse:Security}>\) header. The following example illustrates a SOAP message with a license.

```
<\text{S:Envelope xmlns:S="..."}>
  <\text{S:Header}>
    <\text{wsse:Security xmlns:wsse="...">}
      <\text{r:license xmlns:r="...">}
      ...
    </r:license>
    ...
  </\text{wsse:Security}>
  <\text{S:Body}>
    ...
  </\text{S:Body}>
</\text{S:Envelope}>
```

3.4 Identifying and Referencing Security Tokens

The Web Services Security: SOAP Message Security [WS-Security] specification defines the \textit{wsu:id} attribute as the common mechanism for identifying security tokens (the specification describes the reasons for this). Licenses have an additional identification mechanism available: their licenseId attribute, the value of which is a URI. The following example shows a license that uses both mechanisms:
Licenses can be referenced either according to their location or their licenseId. Location references are dependent on location and can be either local or remote. LicenseId references are not dependent on location.

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

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

Implementations compliant with this profile SHOULD set the `/wsse:SecurityTokenReference/wsse:Reference/@ValueType` attribute to `http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license` when using `wsse:SecurityTokenReference` to refer to a license by licenseId. This is OPTIONAL when referring to a license by location.

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

<table>
<thead>
<tr>
<th>By Location</th>
<th>Local</th>
<th>Remote</th>
<th>By licenseId</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>&lt;wsse:Reference</code></td>
<td><code>&lt;wsse:Reference</code></td>
<td><code>&lt;wsse:Reference</code></td>
</tr>
<tr>
<td></td>
<td><code>&gt;</code></td>
<td><code>&gt;</code></td>
<td><code>&gt;</code></td>
</tr>
</tbody>
</table>

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

```
<S:Envelope xmlns:S="..." xmlns:r="..." xmlns:wsu="..."
  licenseId="urn:foo:SecurityToken:ef375268"
  wsu:Id="SecurityToken-ef375268">
  ...
</S:Envelope>
```
The following example shows a signature over a local license using a location reference to that license. The example demonstrates how the integrity of an (unsigned) license can be preserved by signing it in the <wsse:Security> header.

```xml
<S:Envelope xmlns:S="...">
    <S:Header>
        <wsse:Security xmlns:wsse="">
            <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-ef375268">
                ... 
            </r:license>
            ... 
            <ds:Signature>
                <ds:SignedInfo>
                    ... 
                </ds:SignedInfo>
                <ds:SignatureValue>...</ds:SignatureValue>
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference>
                        <wsse:Reference
                            URI="#SecurityToken-ef375268"/>
                    </wsse:SecurityTokenReference>
                    ... 
                </ds:KeyInfo>
                <wsse:Security/>
            </S:Body>
            ... 
        </S:Body>
    </S:Envelope>
```
Note: since licenses allow the use of the wsu:Id attribute, it is usually not necessary to use the STR-Transform because the license can be referred to directly in the ds:SignedInfo as shown in the following example:

```
<S:Envelope xmlns:S="...">
  <S:Header>
    <wsse:Security xmlns:wsse="...">
      <r:license xmlns:r="..." xmlns:wsu="..." wsu:Id="SecurityToken-ef375268"/>
    </r:license>
    ...
    <ds:Signature>
      <ds:SignedInfo>
        ...
        <ds:Reference URI="#SecurityToken-ef375268">
          <ds:DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>...</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>...</ds:SignatureValue>
      <ds:KeyInfo>...</ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

### 3.5 Authentication

The Web Services Security: SOAP Message Security [WS-Security] specification does not dictate how claim confirmation must be performed. As well, the REL allows for multiple types of confirmation. This profile of WS-Security REQUIRES that message senders and receivers support claim confirmation for `<r:keyHolder>` principals. It is RECOMMENDED that an XML
Signature be used to establish the relationship between the message sender and the claims. This is especially RECOMMENDED whenever the SOAP message exchange is conducted over an unprotected transport.

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

<table>
<thead>
<tr>
<th>Principal</th>
<th>RECOMMENDED Processing Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;r:keyHolder&gt;</td>
<td>The message sender adds (to the security header) an XML Signature that can be verified with the key information specified in the &lt;r:keyHolder&gt; of the referenced license.</td>
</tr>
</tbody>
</table>

Table 3. Processing Rules for Claim Confirmation

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

3.5.1 <r:keyHolder> Principal

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

Sender

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

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

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

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

Example

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

```xml
<S:Envelope xmlns:S="..."/>
  <S:Header>
    <wsse:Security xmlns:wsse="..."/>
      <r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
        <r:grant>
          <r:keyHolder>
            <r:info>
              <ds:KeyValue>...</ds:KeyValue>
            </r:info>
          </r:keyHolder>
          <r:possessProperty/>
          <sx:commonName xmlns:sx="...">John Doe</sx:commonName>
        </r:grant>
        <r:issuer>
          <ds:Signature>...</ds:Signature>
        </r:issuer>
      </r:license>
    <ds:Signature>
      <ds:SignedInfo>...
        <ds:Reference URI="#MsgBody">
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>...</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>...</ds:SignatureValue>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="urn:foo:SecurityToken:ef375268"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf#license"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </S:Header>
</S:Envelope>
```
3.6 Confidentiality

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

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

<table>
<thead>
<tr>
<th>Principal</th>
<th>RECOMMENDED Processing Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;r:keyHolder&gt;</td>
<td>The message sender adds (to the security header) either 1) an <a href="">xenc:ReferenceList</a> that points to one or more <a href="">xenc:EncryptedData</a> elements that can be decrypted with a key which can be determined from information specified in the &lt;r:keyHolder&gt; of the referenced license or 2) an <a href="">xenc:EncryptedKey</a> that can be decrypted with a key determined from information specified in the &lt;r:keyHolder&gt; of the referenced license.</td>
</tr>
</tbody>
</table>

Table 4. Processing Rules for Confidentiality

Note that this section deals only with Confidentiality. Details of authentication of the sender by the receiver must be addressed by means other than those described in this section (see the previous section).
3.6.1 <r:keyHolder> Principal

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

Sender

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

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

Receiver

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

Example

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

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

```xml
<S:Envelope xmlns:S="...">
    <S:Header>
        <wsse:Security xmlns:wsse="...">
            <r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
                <r:grant>
                    <r:keyHolder>
                        <r:info>
                            <ds:KeyValue>...</ds:KeyValue>
                        </r:info>
                    </r:keyHolder>
                </r:grant>
            </r:license>
        </wsse:Security>
    </S:Header>
</S:Envelope>
```
3.7 Error Codes

It is RECOMMENDED that the error codes defined in the Web Services Security: SOAP Message Security [WS-Security] specification are used. However, implementations MAY use custom errors, defined in private namespaces if they desire. Care should be taken not to introduce security vulnerabilities in the errors returned.
4 Types of Licenses (Informative)

4.1 Attribute Licenses

In addition to key information, licenses can carry information about attributes of those keys. Examples of such information on a client are e-mail address or common name. A service's key, on the other hand, might be associated with a DNS name and common name.

The following is an example client attribute license.

```
<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>
</license>
```

The following is an example service attribute license.

```
<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>
</license>
```
Additional examples of and processing rules for the use of attribute licenses can be found in the above sections on Authentication and Confidentiality.

### 4.2 Sender Authorization

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

```xml
<r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
  <r:grant>
    <r:keyHolder>
      <r:info>
        <ds:KeyValue>FDFEWF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
    <sx:rightUri definition='...'/>
    <x:someResource/>
    <x:someCondition/>
  </r:grant>
  <r:issuer>
    <ds:Signature>...</ds:Signature>
  </r:issuer>
</r:license>
```

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

### 4.3 Issuer Authorization

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

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

```xml
<r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
  <r:grant>
    <r:forAll varName='K'/>
    <r:forAll varName='P'/>
    <r:keyHolder>
      <r:info>
        <ds:KeyValue>FDFEWF...</ds:KeyValue>
      </r:info>
    </r:keyHolder>
  </r:grant>
</r:license>
```
The following license is an example issuer authorization license for authorizing an issuer to issue sender authorization licenses.

```
<r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
  <r:grant>
    <r:forAll varName='K'/>
    <r:forAll varName='P'/>
    <r:keyHolder>
      <r:info>
        <ds:KeyValue>FDFEWEFF…</ds:KeyValue>
      </r:info>
      <r:issue/>
      <r:grant>
        <r:forAll varName='K'/>
        <r:resource varRef='P'/>
        <r:keyHolder varRef='K'/>
        <sx:rightUri definition='...'/>
        <r:resource varRef='R'/>
        <r:issue/>
        <r:grant>
          <r:forAll varName='P'/>
          <r:resource varRef='P'/>
          <r:keyHolder varRef='K'/>
          <r:posseesProperty/>
          <r:propertyAbstract varRef='P'/>
          <r:issue/>
        </r:grant>
      </r:grant>
    </r:keyHolder>
  </r:grant>
  <ds:Signature>...<ds:Signature>
</r:issuer>
</r:license>
```

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

```
<r:license xmlns:r="..." licenseId="urn:foo:SecurityToken:ef375268">
  <r:grant>
    <r:forAll varName='I'/>
    <r:keyHolder>
      <r:info>
        <ds:KeyValue>FDFEWEFF…</ds:KeyValue>
      </r:info>
      <r:issue/>
      <r:grant>
        <r:forAll varName='K'/>
        <r:forAll varName='P'/>
        <r:keyHolder varRef='I'/>
        <r:issue/>
        <r:grant>
          <r:forAll varName='K'/>
          <r:forAll varName='P'/>
          <r:keyHolder varRef='I'/>
          <r:issue/>
          <r:grant>
            <r:forAll varName='P'/>
            <r:resource varRef='P'/>
            <r:keyHolder varRef='K'/>
            <r:posseesProperty/>
            <r:propertyAbstract varRef='P'/>
            <r:issue/>
          </r:grant>
        </r:grant>
      </r:grant>
    </r:keyHolder>
  </r:grant>
  <ds:Signature>...<ds:Signature>
</r:issuer>
</r:license>
```
</license>
5 Threat Model and Countermeasures

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

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

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

5.1 Eavesdropping

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

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

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

5.2 Replay

The reliance on authority protected (e.g. signed) licenses to <r:keyHolder> principals precludes all but the key holder from binding the licenses to a SOAP message. Although this mechanism
effectively restricts message authorship to the holder of the confirmation key, it does not preclude
the capture and resubmission of the message by other parties.

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

5.3 Message Insertion

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

5.4 Message Deletion

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

5.5 Message Modification

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

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

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

5.6 Man-in-the-Middle

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


Appendix A: Acknowledgements

Contributors:

Maneesh Sahu Actional Corporation
Gene Thurston AmberPoint
Frank Siebenlist Argonne National Laboratory
Hal Lockhart BEA Systems, Inc.
Corinna Witt BEA Systems, Inc.
Steve Anderson BMC
Richard Levinson Computer Associates
Davanum Srinivas Computer Associates
Thomas DeMartini ContentGuard, Inc.
Guillermo Lao ContentGuard, Inc.
TJ Pannu ContentGuard, Inc.
Xin Wang ContentGuard, Inc.
Merlin Hughes Cybertrust
Sam Wei Documentum
Tim Moses Entrust
Carolina Canales-Valenzuela Ericsson
Dana Kaufman Forum Systems, Inc.
Toshihiro Nishimura Fujitsu Limited
Kefeng Chen GeoTrust
Irving Reid Hewlett-Packard
Kojiro Nakayama Hitachi
Paula Austel IBM
Derek Fu IBM
Maryann Hondo IBM
Kelvin Lawrence IBM
Hiroshi Maruyama 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
Martin Gudgin Microsoft Corporation
Chris Kaler Microsoft Corporation
John Shewchuk Microsoft Corporation
Jeff Hodges Neustar, Inc.
Frederick Hirsch Nokia
Senthil Sengodan Nokia
Abbie Barbir Nortel Networks
Lloyd Burch Novell
Charles Knouse Oblix
Vamsi Motukuru Oracle
<table>
<thead>
<tr>
<th>Name</th>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ramana</td>
<td>Turlapati</td>
<td></td>
<td>Oracle</td>
</tr>
<tr>
<td>Prateek</td>
<td>Mishra</td>
<td></td>
<td>Principal Identity</td>
</tr>
<tr>
<td>Andrew</td>
<td>Nash</td>
<td></td>
<td>Reactivity</td>
</tr>
<tr>
<td>Ben</td>
<td>Hammond</td>
<td></td>
<td>RSA Security</td>
</tr>
<tr>
<td>Rob</td>
<td>Philpott</td>
<td></td>
<td>RSA Security</td>
</tr>
<tr>
<td>Martijn</td>
<td>de Boer</td>
<td></td>
<td>SAP</td>
</tr>
<tr>
<td>Blake</td>
<td>Dournaee</td>
<td></td>
<td>Sarvega</td>
</tr>
<tr>
<td>Coumara</td>
<td>Radja</td>
<td></td>
<td>Sarvega</td>
</tr>
<tr>
<td>Pete</td>
<td>Wenzel</td>
<td></td>
<td>SeeBeyond Technology Corporation</td>
</tr>
<tr>
<td>Ronald</td>
<td>Monzillo</td>
<td></td>
<td>Sun Microsystems</td>
</tr>
<tr>
<td>Jan</td>
<td>Alexander</td>
<td></td>
<td>Systinet</td>
</tr>
<tr>
<td>Symon</td>
<td>Chang</td>
<td></td>
<td>Tibco Software Inc.</td>
</tr>
<tr>
<td>John</td>
<td>Weiland</td>
<td></td>
<td>US Dept of the Navy</td>
</tr>
<tr>
<td>Hans</td>
<td>Granqvist</td>
<td></td>
<td>Verisign</td>
</tr>
<tr>
<td>Phillip</td>
<td>Hallam-Baker</td>
<td></td>
<td>Verisign</td>
</tr>
<tr>
<td>Hemma</td>
<td>Prafullchandra</td>
<td></td>
<td>Verisign</td>
</tr>
</tbody>
</table>
### Appendix B: Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>27-May-2005</td>
<td>Initial draft based on REL Token Profile 1.0, updated for 1.1.</td>
</tr>
<tr>
<td>02</td>
<td>14-Jun-2005</td>
<td>Marked as Committee Draft.</td>
</tr>
</tbody>
</table>