pt, Bold, Not Italic, Complex Script Font: Not Bold, Italic, Centered

Web Services Security XrML-Based Rights Expression Token Profile

Deleted: Binding

Working Draft 03, 30 January 2003

Document identifier:

WSS-REL-03

Location:

TBD

Editors:

Phillip Hallam-Baker, VeriSign
Chris Kaler, Microsoft
Ronald Monzillo, Sun
Anthony Nadalin, IBM

Contributors:

TBD – Revise this list to include WSS TC contributors

Thomas DeMartini, ContentGuard
Maryann Hondo, IBM
Guillermo Lao, ContentGuard
Anthony Nadalin, IBM
TJ Pannu, ContentGuard
John Shewchuck, Microsoft
Phillip Hallam-Baker, Verisign
Chris Kaler, Microsoft
Hiroshi Maruyama, IBM
Nataraj Nagaratnam, IBM
Hemma Prefullchandra, VeriSign
Xin Wang, ContentGuard

Abstract:

This document describes how to use eXtensible rights Markup Language (XrML)–based Rights Expression Language (REL) licenses with the WS-Security specification.

Deleted: This document describes how to use eXtensible rights Markup Language (XrML) based Rights Language (RL) licenses with the WS-Security specification.¶

Status:

This is an interim draft. Please send comments to the editors.

Committee members should send comments on this specification to the mailto:wss@lists.oasis-open.org list. Others should subscribe to and send comments to the wss-comment@lists.oasis-open.org list. To subscribe, visit http://lists.oasis-open.org/ob/adm.pl.

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

Deleted: Page Break

Table of Contents

Deleted: 4
Field Code Changed
Deleted: 4
Field Code Changed
Field Code Changed
Deleted: 4
Deleted: 4
Field Code Changed
Field Code Changed
Deleted: 5
Field Code Changed
Deleted: 5
Field Code Changed
Deleted: 5
Deleted: 6
Field Code Changed
Field Code Changed
Deleted: 7
Field Code Changed
Deleted: 7
Field Code Changed
Deleted: 7
Field Code Changed
Deleted: 7
Field Code Changed
Deleted: 8
Field Code Changed
Deleted: 9
Field Code Changed
Deleted: 9
Deleted: 10
Field Code Changed
Field Code Changed
Deleted: 11
Field Code Changed
Deleted: 12
Field Code Changed
Deleted: 13

1 Introduction

The 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 eXtensible rights Markup Language (XrML)–based Rights Expression Language (REL) licenses with respect to the **Error! Hyperlink reference not valid.** specification.

Deleted:)

Field Code Changed

Deleted: WS-Security

Deleted: Note that Section 1 is non-normative.¶

Formatted: Bullets and Numbering

2 Notations and Terminology

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 RFC2119.

Namespace URIs (of the general form "some-URI") represent some application-dependent or context-dependent URI as defined in RFC2396.

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.

This specification is designed to work with the general XrML2 license structure and processing model, and should be applicable to any XrML2-based rights expression language. The current XrML 2.1 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 an XrML2-based rights expression language.

2.2 Namespaces

Deleted: Readers are presumed to be familiar with the terms in the Internet Security Glossary.¶

Formatted: Bullets and Numbering

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

```
        http://schemas.xmlsoap.org/ws/2002/xx/secext
        http://schemas.xmlsoap.org/ws/2002/xx/utility
```

The following namespaces are used in this document:

Formatted Table

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://schemas.xmlsoap.org/ws/2002/xx/secext
wsu	http://schemas.xmlsoap.org/ws/2002/xx/utility
r	http://www.xrml.org/schema/2002/05/xrml2core

sx	http://www.xrml.org/schema/2002/05/xrml2sx

Table 1 Namespace Prefixes

Formatted: Keep with next

Formatted: Left

Field Code Changed

2.3 Terminology

This specification employs the terminology defined in the WS-Security Core Specification.

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

[TBS]

3 Usage

This section describes the profile (specific mechanisms and procedures) for the XrML binding of WS-Security.

Identification: urn:oasis:names:tc:WSS:1.0:profiles:WSS-REL-token

Contact information: TBD

Description: Given below.

Updates: None.

3.1 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 WS-Security.

At the token level, a processor of XrML-based REL security tokens MUST conform to the required validation and processing rules defined in the respective REL specification.

3.2 Attaching Security Tokens

REL licenses are attached to SOAP messages using WS-Security by placing the license element inside the <wsse:Security> header. The following example illustrates a SOAP message with an license token.

```
<S:Envelope xmlns:S="...">
    <S:Header>
        <wsse:Security xmlns:wsse="...">
            <r:license xmlns:xrml="...">
                ...
            </r:license>
            ...
        </wsse:Security>
    </S:Header>
    <S:Body>
        ...
    </S:Body>
</S:Envelope>


```

3.3 Identifying and Referencing Security Tokens

The WS-Security specification defines the *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:

```
<r:license xmlns:r="..." xmlns:wsu="..."
  licenseId="urn:foo:SecurityToken:ef375268"
```

Deleted: bindings

Deleted: XrML-binding

Formatted: Font: Verdana

Formatted: Font: Verdana

Formatted: Font: Verdana, Font color: Auto

Formatted: Font: Verdana

Formatted: Font: Verdana

Deleted: The processing model for WS-Security with XrML licenses is no different from that of WS-Security with other token formats as described in WS-Security. ¶

Formatted: Bullets and Numbering

Deleted: XrML

Deleted: XrML

Deleted: The WS-Security specification defines the *wsu:Id* attribute as the common mechanism for referencing security tokens by "Id" (the specification describes the reasons for this). Since the XrML specification does not allow attribute extensibility on the <r:license> element, this specification defines a separate mechanism for referencing licenses. The XrML specification allows licenses to be named using a URI with the *licenseId* attribute. Consequently, this specification defines the global namespace qualifier attribute *xmltok:RefType* for use with the <wsse:Reference> element (used within a <wsse:SecurityTokenReference> element). Using this attribute, references can specify the type of token desired thereby allowing the token-specific matching rules to be processed. Specifically, when the *xmktok:RefType* attribute's value is "r:license", then the *URI* attribute refers to an <r:license> element whose *licenseId* attribute is specified by the URI attribute.¶

Formatted: Code,c, Pattern: Clear

```
  wsu:Id="SecurityToken-ef375268">
  ...
</r:license>
```

Licenses can be referenced either according to their licenseId or their location. LicenseId references are not dependent on location. Location references are dependent on location and can be either local or remote.

References may occur in three different contexts:

Formatted: Bullets and Numbering

? The reference may be contained inside the <ds:KeyInfo> element within an XML signature. The reference in this case points to the license that contains the key that was used to sign the digest of the <ds:SignedInfo>. The receiver may use this reference to verify the integrity of the <ds:SignedInfo>.

? The reference may also occur within an element other than the <ds:Signature> element. This may be useful to indicate where a service can find other licenses for additional security-related processing.

? The license may be referenced from within the <ds:SignedInfo> element of an XML signature. To ensure the integrity of the license, a signing authority may sign the license and place the resulting signature within a <ds:Signature> element. In this case, the <ds:SignedInfo> element of the <ds:Signature> contains a <ds:Reference> element that points to the license.

The following few sections demonstrate how to reference licenses from these contexts.

License Referenced from the <ds:KeyInfo> Element of an XML Signature

A license can be referenced from within the <ds:KeyInfo> element of a <ds:Signature> element. WS-Security specifies that this is accomplished using the <wsse:SecurityTokenReference> element.

Implementations compliant with this profile SHOULD set the /wsse:SecurityTokenReference/wsse:Reference/@ValueType attribute to r:license when using wsse:SecurityTokenReference to refer to a license by licenseId. This is not necessary when referring to a license by location.

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

By licenseId		<wsse:SecurityTokenReference>
		  <wsse:Reference
		    URI="urn:foo:SecurityToken:ef375268"
		    ValueType="r:license"
		  />
		</wsse:SecurityTokenReference>

By Location	Local	<wsse:SecurityTokenReference>
		  <wsse:Reference
		    URI="#SecurityToken-ef375268"
		  />
		</wsse:SecurityTokenReference>

	Remote	<wsse:SecurityTokenReference>
		  <wsse:Reference
		    URI="http://www.foo.com/ef375268.xml"
		  />
		</wsse:SecurityTokenReference>

Formatted: Code,c, Pattern: Clear

Formatted: Code,c, Pattern: Clear

Formatted: Code,c, Pattern: Clear

Table 2. <wsse:SecurityTokenReference>

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="...">
  <S:Header>
    <wsse:Security xmlns:wsse="...">
      <r:license xmlns:r="..."
licenseId="urn:foo:SecurityToken:ef375268">
        ...
      </r:license>
      ...
      <ds:Signature>
        <ds:SignedInfo>
          ...
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
              URI="urn:foo:SecurityToken:ef375268"
              ValueType="r:license"
            />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

Formatted: Code,c, Pattern: Clear

License Referenced from Elements Other Than <ds:Signature>

A license can be referenced from elements other than <ds:Signature>. WS-Security specifies that this is accomplished using the <wsse:SecurityTokenReference> element. (For details on the use of the <wsse:SecurityTokenReference> element to refer to licenses, please see Table 2 in 0).

The following example demonstrates how a <wsse:SecurityTokenReference> can be used to refer to a license from directly within the <wsse:Security> header element (just one such element that is an element other than a <ds:Signature>). In this case, we choose to show a location reference to a remote license.

```
<S:Envelope xmlns:S="...">
  <S:Header>
    <wsse:Security xmlns:wsse="...">
      ...
      <wsse:SecurityTokenReference>
        <wsse:Reference
          URI="http://www.foo.com/ef375268.xml"
        />
      </wsse:SecurityTokenReference>
      ...
    </wsse:Security>
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

Formatted: Code,c, Pattern: Clear

License Referenced from the <ds:SignedInfo> Element of an XML Signature

A license can be referenced from within the <ds:SignedInfo> element of a <ds:Signature> element. DIGSIG specifies that this is accomplished using the <ds:Reference> element. The following table demonstrates the use of the <ds:Reference> element to refer to licenses.

By licenseId		<ds:Reference URI=”urn:foo:SecurityToken:ef375268”>
		  <ds:DigestMethod
		    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
		  />
		  <ds:DigestValue>...</ds:DigestValue>
		</ds:Reference>

By Location	Local	<ds:Reference URI="#SecurityToken-ef375268”>
		  <ds:DigestMethod
		    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
		  />
		  <ds:DigestValue>...</ds:DigestValue>
		</ds:Reference>

	Remote	<ds:Reference URI=”http://www.foo.com/ef375268.xml”>
		  <ds:DigestMethod
		    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
		  />
		  <ds:DigestValue>...</ds:DigestValue>
		</ds:Reference>

Formatted: Code,c, Pattern: Clear

Formatted: Code,c, Pattern: Clear

Formatted: Code,c, Pattern: Clear

Table 3. <ds:Reference>

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.

```
<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:Header>
```

Formatted: Code,c, Pattern: Clear

```
   <S:Body>
      ...
   </S:Body>
</S:Envelope>
```

3.4 Authentication

Deleted: Proof-of-Possession of Security Tokens

Deleted: As previously stated, the WS-Security specification does not dictate how subject confirmation must be performed. As well, the XrML specification allows for multiple types of confirmation. If a secure transport is not used, it is strongly RECOMMENDED that a key-based confirmation mechanism be used.¶
Any processor of XrML security tokens MUST conform to th ... [1]

The WS-Security specification does not dictate how claim confirmation must be performed. As well, XrML-based RELs allow for multiple types of confirmation. The REL profile of WS-Security requires that message senders and receivers support claim confirmation for <r:keyHolder> principals. It is strongly 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.

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 REL license.

Table 4. 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.

<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 security token.

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.

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 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.

Formatted: Code,c, Pattern: Clear

```
<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>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
              URI="urn:foo:SecurityToken:ef375268"
              ValueType="r:license"
            />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>

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

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

</S:Envelope>
```

Formatted: Complex Script Font: Courier New, 9 pt

3.5 Error Codes

It is RECOMMENDED thatthe error codes defined in the 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.

Deleted: When using XrML licenses, i

Deleted: to use

3.6 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 XrML-based REL 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.

Deleted: The use of XrML licenses with WS-Security introduces no new threats beyond those identified for XrML or WS-Security with other types of security tokens.¶

The use of XrML-based REL licenses with WS-Security introduces no new threats beyond those identified for the XrML-based 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 XrML-based REL licenses ownership is verified by 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 license tokens, they SHOULD be signed natively and/or using the mechanisms outlined in WS-Security. This allows readers of the tokens to be certain that the tokens 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.

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.

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.

Message Insertion

The XrML-based REL token 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.

Message Deletion

The XrML-based REL token profile of WS-Security is not vulnerable to message deletion attacks. 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.

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, XrML-based REL licenses appearing in <wsse:Security> header elements MUST 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 XrML-based REL licenses and/or license references from modification while in transport, but signatures are required to extend such protection through intermediaries.

Man-in-the-Middle

The XrML-based REL token 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.

Formatted: Normal

4 Acknowledgements

This specification was developed as a result of joint work of many individuals from the WSS TC including:

TBD

5 References

[DIGSIG] Informational RFC 2828, "Internet Security Glossary," May 2000.

Field Code Changed

[KEYWORDS] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

Field Code Changed

[MPEG-REL] Text of ISO/IEC Final Committee Draft 21000-5 Rights Expression Language, December 2002.

[SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

Field Code Changed

W3C Working Draft, Nilo Mitra (Editor), SOAP Version 1.2 Part 0: Primer, June 2002.

Field Code Changed

W3C Working Draft, SOAP Version 1.2 Part 1: Messaging Framework, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen (Editors), June 2002.

Field Code Changed

W3C Working Draft, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen (Editors), SOAP Version 1.2 Part 2: Adjuncts, June 2002.

Field Code Changed

[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

Field Code Changed

[WS-Security] TBS – point to the OASIS core draft

[XML-ns] W3C Recommendation, "Namespaces in XML," 14 January 1999.

Field Code Changed

[XML Signature] W3C Recommendation, "XML Signature Syntax and Processing," 12 February 2002.

Field Code Changed

[XML Token] Contribution to the WSS TC**,** Chris Kaler (Editor), WS-Security Profile for XML-based Tokens, August 2002.

[XrML] ContentGuard, eXtensible rights Markup Language Core 2.1 Specification, 20 May 2002.

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

Appendix B: Notices

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 © OASIS Open 2002. *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.

Page 11: [1] Deleted **Phillip Hallam-Baker** **30-Jan-03 3:39 PM**

As previously stated, the WS-Security specification does not dictate how subject confirmation must be performed. As well, the XrML specification allows for multiple types of confirmation. If a secure transport is not used, it is strongly RECOMMENDED that a key-based confirmation mechanism be used.

Any processor of XrML security tokens MUST conform to the required validation and processing rules defined in the XrML specification.

The following table illustrates how several different confirmation mechanisms are processed:

Mechanism	RECOMMENDED Processing Rules
<r:keyHolder>	If the entity represented by r:keyHolder is the sender of the message, it SHOULD include in the wsse:Security header a ds:Signature that can be verified with the key information in the referenced security token. Otherwise, the sender may chose to either provide no proof of possession for that entity to that entity's key, vouch for the authenticity itself, or provide some other proof of possession.