WS-SecurityPolicy 1.3

OASIS Standard

2 February 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc
(Authoritative)
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.pdf
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html

Previous Version:

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cs-01.doc
(Authoritative)
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cs-01.pdf
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cs-01.html

Latest Version:

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.doc
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.pdf
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.html

Technical Committee:

OASIS Web Services Secure Exchange TC

Chair(s):

Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):

Anthony Nadalin, IBM
Marc Goodner, Microsoft
Martin Gudgin, Microsoft
Abbie Barbir, Nortel
Hans Granqvist, VeriSign

Related work:

N/A

Declared XML Namespace(s):

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802

Abstract:

This document indicates the policy assertions for use with [WS-Policy] which apply to WSS:
SOAP Message Security [WSS10, WSS11], [WS-Trust] and [WS-SecureConversation]

Status:

This document was last revised or approved by the WS-SX TC on the above date. The level of
approval is also listed above. Check the current location noted above for possible later revisions
of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/ws-sx.

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

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/ws-sx.

Notices

Copyright © OASIS® 1993–2009. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

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 section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, 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 OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

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' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on 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 implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

WS-Policy defines a framework for allowing web services to express their constraints and requirements. Such constraints and requirements are expressed as policy assertions. This document defines a set of security policy assertions for use with the [WS-Policy] framework with respect to security features provided in WSS: SOAP Message Security [WSS10, WSS11], [WS-Trust] and [WS-SecureConversation]. Within this specification the use of the namespace prefix wsp refers to the WS-Policy 1.5 namespace. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

Sections 11, 12 and all examples and all Appendices are non-normative.

1.1 Example

Table 1 shows an "Effective Policy" example, including binding assertions and associated property assertions, token assertions and integrity and confidentiality assertions. This example has a scope of [Endpoint Policy Subject], but for brevity the attachment mechanism is not shown.

Table 1: Example security policy.

```
(01) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
(02)   <sp:SymmetricBinding>
(03)     <wsp:Policy>
(04)       <sp:ProtectionToken>
(05)         <wsp:Policy>
(06)           <sp:Kerberos sp:IncludeToken=".../IncludeToken/Once" />
(07)             <wsp:Policy>
(08)               <sp:WSSKerberosV5ApReqToken11/>
(09)             <wsp:Policy>
(10)           </sp:Kerberos>
(11)         </wsp:Policy>
(12)       </sp:ProtectionToken>
(13)       <sp:SignBeforeEncrypting />
(14)       <sp:EncryptSignature />
(15)     </wsp:Policy>
(16)   </sp:SymmetricBinding>
(17)   <sp:SignedParts>
(18)     <sp:Body/>
(19)     <sp:Header
           Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
         />
(20)   </sp:SignedParts>
(21)   <sp:EncryptedParts>
(22)     <sp:Body/>
```

```
44   (23)    </sp:EncryptedParts>
45   (24)</wsp:Policy>
```

46

47   Line 1 in Table 1 indicates that this is a policy statement and that all assertions contained by the
48   wsp:Policy element are required to be satisfied. Line 2 indicates the kind of security binding in force. Line
49   3 indicates a nested `wsp:Policy` element which contains assertions that qualify the behavior of the
50   SymmetricBinding assertion. Line 4 indicates a ProtectionToken assertion. Line 5 indicates a nested
51   `wsp:Policy` element which contains assertions indicating the type of token to be used for the
52   ProtectionToken. Lines 6 to 10 indicate that a Kerberos V5 APREQ token is to be used by both parties in
53   a message exchange for protection. Line 13 indicates that signatures are generated over plaintext rather
54   than ciphertext. Line 14 indicates that the signature over the signed messages parts is required to be
55   encrypted. Lines 17-20 indicate which message parts are to be covered by the primary signature; in this
56   case the soap:Body element, indicated by Line 18 and any SOAP headers in the WS-Addressing
57   namespace, indicated by line 19. Lines 21-23 indicate which message parts are to be encrypted; in this
58   case just the soap:Body element, indicated by Line 22.

59 1.2 Namespaces

60   The XML namespace URIs that MUST be used by implementations of this specification are:

```
61   http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
62   http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802
```

63

64   Table 2 lists XML namespaces that are used in this specification. The choice of any namespace prefix is
65   arbitrary and not semantically significant.

66   *Table 2: Prefixes and XML Namespaces used in this specification.*

Prefix	Namespace	Specification(s)
S	http://schemas.xmlsoap.org/soap/envelope/	[SOAP]
S12	http://www.w3.org/2003/05/soap-envelope	[SOAP12]
ds	http://www.w3.org/2000/09/xmldsig#	[XML-Signature]
enc	http://www.w3.org/2001/04/xmlenc#	[XML-Encrypt]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[WSS10]
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WSS10]
wsse11	http://docs.oasis-open.org/wss/oasis-wss-wsecurity-secext-1.1.xsd	[WSS11]
xsd	http://www.w3.org/2001/XMLSchema	[XML-Schema1], [XML-Schema2]
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[WS-Trust]
wst14	http://docs.oasis-open.org/ws-sx/ws-trust/200802	[WS-Trust]
wsc	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512	[WS-SecureConversation]

wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	This specification
sp13	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802	This specification
wsp	http://www.w3.org/ns/ws-policy	[WS-Policy]

1.3 Schema Files

A normative copy of the XML Schemas [XML-Schema1, XML-Schema2] description for this specification can be retrieved from the following address:

```
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy-1.2.xsd
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy-1.3.xsd
```

1.4 Terminology

Policy - A collection of policy alternatives.

Policy Alternative - A collection of policy assertions.

Policy Assertion - An individual requirement, capability, other property, or a behavior.

Initiator - The role sending the initial message in a message exchange.

Recipient - The targeted role to process the initial message in a message exchange.

Security Binding - A set of properties that together provide enough information to secure a given message exchange.

Security Binding Property - A particular aspect of securing an exchange of messages.

Security Binding Assertion - A policy assertion that identifies the type of security binding being used to secure an exchange of messages.

Security Binding Property Assertion - A policy assertion that specifies a particular value for a particular aspect of securing an exchange of message.

Assertion Parameter - An element of variability within a policy assertion.

Token Assertion -Describes a token requirement. Token assertions defined within a security binding are used to satisfy protection requirements.

Supporting Token - A token used to provide additional claims.

1.4.1 Notational Conventions

The key words "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].

This specification uses the following syntax to define outlines for assertions:

- The syntax appears as an XML instance, but values in italics indicate data types instead of literal values.
- Characters are appended to elements and attributes to indicate cardinality:
  - o "?" (0 or 1)
  - o "*" (0 or more)
  - o "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

103 • The characters "[" and "]" are used to call out references and property names.

104 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be
105 added at the indicated extension points but MUST NOT contradict the semantics of the parent
106 and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver
107 SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated
108 below.

109 • XML namespace prefixes (see Table 2) are used to indicate the namespace of the element being
110 defined.

111

112 Elements and Attributes defined by this specification are referred to in the text of this document using
113 XPath 1.0 expressions. Extensibility points are referred to using an extended version of this syntax:

114 • An element extensibility point is referred to using {any} in place of the element name. This
115 indicates that any element name can be used, from any namespace other than the namespace of
116 this specification.

117 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
118 indicates that any attribute name can be used, from any namespace other than the namespace of
119 this specification.

120 Extensibility points in the exemplar MAY NOT be described in the corresponding text.

121 In this document reference is made to the `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires`
122 elements in a utility schema (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
123 1.0.xsd). The `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires` elements were added to the
124 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp
125 element could reference it (as is done here).

126

127 WS-SecurityPolicy is designed to work with the general Web Services framework including WSDL service
128 descriptions, UDDI businessServices and bindingTemplates and SOAP message structure and message
129 processing model, and WS-SecurityPolicy SHOULD be applicable to any version of SOAP. The current
130 SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit
131 the applicability of this specification to a single version of SOAP.

1.5 Normative References

133 134	[RFC2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.
135		http://www.ietf.org/rfc/rfc2119.txt
136	[SOAP]	W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.
137		http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
138 139	[SOAP12]	W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003.
140		http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
141 142	[SOAPNorm]	W3C Working Group Note, "SOAP Version 1.2 Message Normalization", 8 October 2003.
143		http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/
144 145 146	[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
147		http://www.ietf.org/rfc/rfc3986.txt

148

149	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
150		
151		http://www.ietf.org/rfc/rfc2068.txt
152	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
153		http://www.ietf.org/rfc/rfc2246.txt
154	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
155		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
156	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
157		
158		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
159	[WS-Policy]	W3C Recommendation, "Web Services Policy 1.5 - Framework", 04 September 2007.
160		
161		http://www.w3.org/TR/2007/REC-ws-policy-20070904/
162		W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
163		
164		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
165	[WS-PolicyAttachment]	W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04 September 2007.
166		
167		http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
168		W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
169		
170		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
171		
172	[WS-Trust]	OASIS Standard, "WS-Trust 1.4", February 2009
173		http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc
174		
175		OASIS Standard, "WS-Trust 1.3", March 2007
176		http://docs.oasis-open.org/ws-sx/ws-trust/200512
177	[WS-SecureConversation]	OASIS Standard, "WS-SecureConversation 1.4", February 2009
178		http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc
179		
180	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
181		
182		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
183		
184	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
185		
186		http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
187		
188	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004
189		
190		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
191		
192	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
193		
194		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
195		

196 197	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
198 199		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
200 201	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
202 203		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
204 205	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
206 207		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
208 209	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
210		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
211 212	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
213 214		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
215 216	[WSS:RELTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
217		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
218 219	[WSS:RELTokenProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile 1.1", February 2006
220 221		http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
222 223	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1", February 2006
224 225		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
226 227	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
228		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
229 230	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002.
231 232 233 234		http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ W3C Recommendation, D. Eastlake et al. XML Signature Syntax and Processing (Second Edition). 10 June 2008. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
235 236	[XPATH]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999.
237		http://www.w3.org/TR/1999/REC-xpath-19991116
238 239	[XPath 2.0 Filter]	W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November 2002.
240		http://www.w3.org/TR/2002/REC-xmldsig-filter2-20021108/
241 242	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition", 28 October 2004.
243		http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

244	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second
245		Edition", 28 October 2004.
246		http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.6 Non-Normative References

248   None.

2 Security Policy Model

This specification defines policy assertions for the security properties for Web services. These assertions are primarily designed to represent the security characteristics defined in the WSS: SOAP Message Security [WSS10] [WSS11], [WS-Trust] and [WS-SecureConversation] specifications, but they can also be used for describing security requirements at a more general or transport-independent level.

The primary goal of this specification is to define an initial set of patterns or sets of assertions that represent common ways to describe how messages are secured on a communication path. The intent is to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging transport security, but to be specific enough to ensure interoperability based on assertion matching.

It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for selecting policy alternatives and the attachment mechanism for associating policy assertions with web service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters or attributes. This enables first-level, QName based assertion matching without security domain-specific knowledge to be done at the framework level. The first level matching is intended to provide a narrowed set of policy alternatives that are shared by the two parties attempting to establish a secure communication path. Parameters defined by this specification represent additional information for engaging behaviors that do not need to participate in matching. When multiple security policy assertions of the same type with parameters present occur in the same policy alternative the parameters should be treated as a union. Note that a service may choose to accept messages that do not match its policy.

In general, assertions defined in this specification allow additional attributes, based on schemas, to be added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not match based on these attributes. Attributes specified on the assertion element that are not defined in this specification or in WS-Policy are to be treated as informational properties.

2.1 Security Assertion Model

The goal to provide richer semantics for combinations of security constraints and requirements and enable first-level QName matching, is enabled by the assertions defined in this specification being separated into simple patterns: what parts of a message are being secured (Protection Assertions), general aspects or pre-conditions of the security (Conditional Assertions), the security mechanism (Security Binding Assertions) that is used to provide the security, the token types and usage patterns (Supporting Token Assertions) used to provide additional claims, and token referencing and trust options (WSS and Trust Assertions).

To indicate the scope of protection, assertions identify message parts that are to be protected in a specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

The general aspects of security includes the relationships between or characteristics of the environment in which security is being applied, such as the tokens being used, which are for integrity or confidentiality protection and which are supporting, the applicable algorithms to use, etc.

291 The security binding assertion is a logical grouping which defines how the general aspects are used to
292 protect the indicated parts. For example, that an asymmetric token is used with a digital signature to
293 provide integrity protection, and that parts are encrypted with a symmetric key which is then encrypted
294 using the public key of the recipient. At its simplest form, the security binding restricts what can be placed
295 in the `wsse:Security` header and the associated processing rules.

296

297 The intent of representing characteristics as assertions is so that QName matching will be sufficient to
298 find common alternatives and so that many aspects of security can be factored out and re-used. For
299 example, it may be common that the mechanism is constant for an endpoint, but that the parts protected
300 vary by message action.

301

302 Assertions defined by this specification MUST NOT include the wsp:Ignorable attribute in its attributes
303 with a value of true.

2.2 Nested Policy Assertions

305 Assertions MAY be used to further qualify a specific aspect of another assertion. For example, an
306 assertion describing the set of algorithms to use MAY qualify the specific behavior of a security binding. If
307 the schema outline below for an assertion type requires a nested policy expression but the assertion does
308 not further qualify one or more aspects of the behavior indicated by the assertion type (i.e., no assertions
309 are needed in the nested policy expression), the assertion MUST include an empty <wsp:Policy/>
310 element. For further information consult the section Policy Assertion Nesting of [WS-Policy].

2.3 Security Binding Abstraction

312 As previously indicated, individual assertions are designed to be used in multiple combinations. The
313 binding represents common usage patterns for security mechanisms. These Security Binding assertions
314 are used to determine how the security is performed and what to expect in the `wsse:Security` header.
315 Bindings are described textually and enforced programmatically. This specification defines several
316 bindings but others can be defined and agreed to for interoperability if participating parties support it.

317

318 A binding defines the following security characteristics:

319 • The minimum set of tokens that will be used and how they are bound to messages. Note that
320   services might accept messages containing more tokens than those specified in policy.
321 • Any necessary key transport mechanisms
322 • Any REQUIRED message elements (e.g. timestamps) in the `wsse:Security` header.
323 • The content and ordering of elements in the `wsse:Security` header. Elements not specified in
324   the binding are not allowed.
325 • Various parameters, including those describing the algorithms to be used for canonicalization,
326   signing and encryption.

327

328 Together the above pieces of information, along with the assertions describing conditions and scope,
329 provide enough information to secure messages between an initiator and a recipient. A policy consumer
330 has enough information to construct messages that conform to the service's policy and to process
331 messages returned by the service. Note that a service MAY choose to reject messages despite them
332 conforming to its policy, for example because a client certificate has been revoked. Note also that a
333 service MAY choose to accept messages that do not conform to its policy.

334

335   The following list identifies the bindings defined in this specification. The bindings are identified primarily
336   by the style of encryption used to protect the message exchange. A later section of this document
337   provides details on the assertions for these bindings.

338   •   TransportBinding (Section 7.3)

339   •   SymmetricBinding (Section 7.4)

340   •   AsymmetricBinding (Section 7.5)

3 Policy Considerations

The following sections discuss details of WS-Policy and WS-PolicyAttachment relevant to this specification.

3.1 Nested Policy

This specification makes extensive use of nested policy assertions as described in the Policy Assertion Nesting section of WS-Policy.

3.2 Policy Subjects

WS-PolicyAttachment defines various attachment points for policy. This section defines properties that are referenced later in this document describing the RECOMMENDED or REQUIRED attachment points for various assertions. In addition, Appendix A groups the various assertions according to policy subject.

Note: This specification does not define any assertions that have a scope of [Service Policy Subject].

[Message Policy Subject]

This property identifies a Message Policy Subject [WS-PolicyAttachment]. WS-PolicyAttachment defines seven WSDL [WSDL 1.1] policy attachment points with Message Policy Subject:

wsdl:message

> A policy expression containing one or more assertions with Message Policy Subject MUST NOT be attached to a wsdl:message.

wsdl:portType/wsdl:operation/wsdl:input, ./wsdl:output, or ./wsdl:fault

> A policy expression containing one or more assertions with Message Policy Subject MUST NOT be attached to a descendant of wsdl:portType.

wsdl:binding/wsdl:operation/wsdl:input, ./wsdl:output, or ./wsdl:fault

> A policy expression containing one or more of the assertions with Message Policy Subject MUST be attached to a descendant of wsdl:binding.

[Operation Policy Subject]

A token assertion with Operation Policy Subject indicates usage of the token on a per-operation basis:

wsdl:portType/wsdl:operation

> A policy expression containing one or more token assertions MUST NOT be attached to a wsdl:portType/wsdl:operation.

wsdl:binding/wsdl:operation

> A policy expression containing one or more token assertions MUST be attached to a wsdl:binding/wsdl:operation.

[Endpoint Policy Subject]

A token assertion instance with Endpoint Policy Subject indicates usage of the token for the entire set of messages described for the endpoint:

wsdl:portType

380   A policy expression containing one or more assertions with Endpoint Policy Subject MUST NOT
381   be attached to a wsdl:portType.

382 wsdl:binding

383   A policy expression containing one or more of the assertions with Endpoint Policy Subject
384   SHOULD be attached to a wsdl:binding.

385 wsdl:port

386   A policy expression containing one or more of the assertions with Endpoint Policy Subject MAY
387   be attached to a wsdl:port

388 **4 Protection Assertions**

389 The following assertions are used to identify *what* is being protected and the level of protection provided.
390 These assertions SHOULD apply to [Message Policy Subject]. These assertions MAY apply to [Endpoint
391 Policy Subject] or [Operation Policy Subject]. Where they apply to [Operation Policy Subject] they apply to
392 all messages of that operation. Where they apply to [Endpoint Policy Subject] they apply to all operations
393 of that endpoint.

394 Note that when assertions defined in this section are present in a policy, the order of those assertions in
395 that policy has no effect on the order of signature and encryption operations (see Section 6.3).

396 **4.1 Integrity Assertions**

397 Two mechanisms are defined for specifying the set of message parts to integrity protect. One uses
398 QNames to specify either message headers or the message body while the other uses XPath
399 expressions to identify any part of the message.

400 **4.1.1 SignedParts Assertion**

401 The SignedParts assertion is used to specify the parts of the message outside of security headers that
402 require integrity protection. This assertion can be satisfied using WSS: SOAP Message Security
403 mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the
404 message over a secure transport protocol like HTTPS. The binding specific token properties detail the
405 exact mechanism by which the protection is provided.

406

407 There MAY be multiple SignedParts assertions present. Multiple SignedParts assertions present within a
408 policy alternative are equivalent to a single SignedParts assertion containing the union of all specified
409 message parts. Note that this assertion does not require that a given part appear in a message, just that if
410 such a part appears, it requires integrity protection.

411 **Syntax**

```
412    <sp:SignedParts xmlns:sp="..." ... >
413      <sp:Body />?
414      <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
415      <sp:Attachments>
416        <sp13:ContentSignatureTransform /> ?
417        <sp13:AttachmentCompleteSignatureTransform /> ?
418      </sp:Attachments> ?
419      ...
420    </sp:SignedParts>
```

421

422 The following describes the attributes and elements listed in the schema outlined above:

423 /sp:SignedParts

424   This assertion specifies the parts of the message that need integrity protection. If no child
425   elements are specified, all message headers targeted at the UltimateReceiver role [SOAP12] or
426   actor [SOAP11] and the body of the message MUST be integrity protected.

427 /sp:SignedParts/sp:Body

428   Presence of this OPTIONAL empty element indicates that the entire body, that is the soap:Body
429   element, it's attributes and content, of the message needs to be integrity protected.

430 /sp:SignedParts/sp:Header

431   Presence of this OPTIONAL element indicates a specific SOAP header, its attributes and content
432   (or set of such headers) needs to be protected. There may be multiple sp:Header elements within
433   a single sp:SignedParts element. If multiple SOAP headers with the same local name but
434   different namespace names are to be integrity protected multiple sp:Header elements are
435   needed, either as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts
436   assertions.
437   This element only applies to SOAP header elements targeted to the same actor/role as the
438   Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
439   SOAP Header elements targeted to a different actor/role, that may be accomplished using the
440   sp:SignedElements assertion.

441 /sp:SignedParts/sp:Header/@Name

442   This OPTIONAL attribute indicates the local name of the SOAP header to be integrity protected. If
443   this attribute is not specified, all SOAP headers whose namespace matches the Namespace
444   attribute are to be protected.

445 /sp:SignedParts/sp:Header/@Namespace

446   This REQUIRED attribute indicates the namespace of the SOAP header(s) to be integrity
447   protected.

448 /sp:SignedParts/sp:Attachments

449   Presence of this OPTIONAL element indicates that all SwA (SOAP Messages with Attachments)
450   attachments [SwA] are to be integrity protected. When SOAP Message Security is used to
451   accomplish this, all message parts other than the part containing the primary SOAP envelope are
452   to be integrity protected as outlined in WSS: SOAP Message Security [WSS:SwAProfile1.1].

453 /sp:SignedParts/sp:Attachments/sp13:ContentSignatureTransform

454   Presence of this OPTIONAL empty element indicates that the
455   AttachmentContentSignatureTransform must be used as part of attachment protection.

456 /sp:SignedParts/sp:Attachments/sp13:AttachmentCompleteSignatureTransform

457   Presence of this OPTIONAL empty element indicates that the
458   AttachmentCompleteSignatureTransform must be used as part of attachment protection.

459   This is the default if neither sp13:ContentSignatureTransform or
460   sp13:AttachmentCompleteSignatureTransform are specified.

461 ## 4.1.2 SignedElements Assertion

462 The SignedElements assertion is used to specify arbitrary elements in the message that require integrity
463 protection. This assertion can be satisfied using WSS: SOAP Message Security mechanisms or by
464 mechanisms out of scope of SOAP message security, for example by sending the message over a
465 secure transport protocol like HTTPS. The binding specific token properties detail the exact mechanism
466 by which the protection is provided.

467

468 There MAY be multiple SignedElements assertions present. Multiple SignedElements assertions present
469 within a policy alternative are equivalent to a single SignedElements assertion containing the union of all
470 specified XPath expressions.

471 **Syntax**

472 ```
473 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
474   <sp:XPath>xs:string</sp:XPath>+
475   <sp13:Xpath2 Filter="xs:string">xs:string</sp13:Xpath2>+
476   ...
      </sp:SignedElements>
```

477 The following describes the attributes and elements listed in the schema outlined above:

478 /sp:SignedElements

479     This assertion specifies the parts of the message that need integrity protection.

480 /sp:SignedElements/@XPathVersion

481     This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
482     attribute is provided, then XPath 1.0 is assumed.

483 /sp:SignedElements/sp:XPath

484     This element contains a string specifying an XPath expression that identifies the nodes to be
485     integrity protected. The XPath expression is evaluated against the S:Envelope element node of
486     the message. Multiple instances of this element MAY appear within this assertion and SHOULD
487     be treated as separate references in a signature when message security is used.

488 /sp:SignedElements/sp:XPath2

489     This element contains a string specifying an XPath 2 expression that identifies the nodes to be
490     integrity protected. The XPath expression is evaluated against the S:Envelope element node of
491     the message. Multiple instances of this element MAY appear within this assertion and SHOULD
492     be treated as separate references in a signature when message security is used.

493 /sp:SignedElements/sp:XPath2@Filter

494     This REQUIRED attribute contains a string to specify an [XPath Filter 2.0] transform to apply.

## 495 4.2 Confidentiality Assertions

496 Two mechanisms are defined for specifying the set of message parts to confidentiality protect. One uses
497 QNames to specify either message headers or the message body while the other uses XPath
498 expressions to identify any part of the message.

## 499 4.2.1 EncryptedParts Assertion

500 The EncryptedParts assertion is used to specify the parts of the message that require confidentiality. This
501 assertion can be satisfied with WSS: SOAP Message Security mechanisms or by mechanisms out of
502 scope of SOAP message security, for example by sending the message over a secure transport protocol
503 like HTTPS.  The binding specific token properties detail the exact mechanism by which the protection is
504 provided.

505

506 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present
507 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all
508 specified message parts. Note that this assertion does not require that a given part appear in a message,
509 just that if such a part appears, it requires confidentiality protection.

510 **Syntax**

```
511    <sp:EncryptedParts xmlns:sp="..." ... >
512      <sp:Body/>?
513      <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
514      <sp:Attachments />?
515      ...
516    </sp:EncryptedParts>
```

517

518 The following describes the attributes and elements listed in the schema outlined above:

519 /sp:EncryptedParts

520 This assertion specifies the parts of the message that need confidentiality protection. The single
521 child element of this assertion specifies the set of message parts using an extensible dialect.

522 If no child elements are specified, the body of the message MUST be confidentiality protected.

523 /sp:EncryptedParts/sp:Body

524 Presence of this OPTIONAL empty element indicates that the entire body of the message needs
525 to be confidentiality protected. In the case where mechanisms from WSS: SOAP Message
526 Security are used to satisfy this assertion, then the soap:Body element is encrypted using the
527 #Content encryption type.

528 /sp:EncryptedParts/sp:Header

529 Presence of this OPTIONAL element indicates that a specific SOAP header (or set of such
530 headers) needs to be protected. There may be multiple sp:Header elements within a single Parts
531 element. Each header or set of headers MUST be encrypted. Such encryption will encrypt such
532 elements using WSS 1.1 Encrypted Headers. As such, if WSS 1.1 Encrypted Headers are not
533 supported by a service, then this element cannot be used to specify headers that require
534 encryption using message level security. If multiple SOAP headers with the same local name but
535 different namespace names are to be encrypted then multiple sp:Header elements are needed,
536 either as part of a single sp:EncryptedParts assertion or as part of separate sp:EncryptedParts
537 assertions.

538 /sp:EncryptedParts/sp:Header/@Name

539 This OPTIONAL attribute indicates the local name of the SOAP header to be confidentiality
540 protected. If this attribute is not specified, all SOAP headers whose namespace matches the
541 Namespace attribute are to be protected.

542 /sp:EncryptedParts/sp:Header/@Namespace

543 This REQUIRED attribute indicates the namespace of the SOAP header(s) to be confidentiality
544 protected.

545 /sp:EncryptedParts/sp:Attachments

546 Presence of this OPTIONAL empty element indicates that all SwA (SOAP Messages with
547 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message
548 Security is used to accomplish this, all message parts other than the part containing the primary
549 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security
550 [WSS:SwAProfile1.1].

551 4.2.2 EncryptedElements Assertion

552 The EncryptedElements assertion is used to specify arbitrary elements in the message that require
553 confidentiality protection. This assertion can be satisfied using WSS: SOAP Message Security
554 mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the
555 message over a secure transport protocol like HTTPS. The binding specific token properties detail the
556 exact mechanism by which the protection is provided.

557

558 There MAY be multiple EncryptedElements assertions present. Multiple EncryptedElements assertions
559 present within a policy alternative are equivalent to a single EncryptedElements assertion containing the
560 union of all specified XPath expressions.

561 **Syntax**

```
562    <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
563      <sp:XPath>xs:string</sp:XPath>+
564      ...
565    </sp:EncryptedElements>
```

566   The following describes the attributes and elements listed in the schema outlined above:

567   /sp:EncryptedElements

568   This assertion specifies the parts of the message that need confidentiality protection. Any such
569   elements are subject to #Element encryption.

570   /sp:EncryptedElements/@XPathVersion

571   This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
572   attribute is provided, then XPath 1.0 is assumed.

573   /sp:EncryptedElements/sp:XPath

574   This element contains a string specifying an XPath expression that identifies the nodes to be
575   confidentiality protected. The XPath expression is evaluated against the S:Envelope element
576   node of the message. Multiple instances of this element MAY appear within this assertion and
577   SHOULD be treated as separate references.

578   4.2.3 ContentEncryptedElements Assertion

579   The ContentEncryptedElements assertion is used to specify arbitrary elements in the message that
580   require confidentiality protection of their content. This assertion can be satisfied using WSS: SOAP
581   Message Security mechanisms or by mechanisms out of scope of SOAP message security, for example
582   by sending the message over a secure transport protocol like HTTPS. The binding specific token
583   properties detail the exact mechanism by which the protection is provided.

584

585   There MAY be multiple ContentEncryptedElements assertions present. Multiple
586   ContentEncryptedElements assertions present within a policy alternative are equivalent to a single
587   ContentEncryptedElements assertion containing the union of all specified XPath expressions.

588   **Syntax**

```
589    <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>
590      <sp:XPath>xs:string</sp:XPath>+
591      ...
592    </sp:ContentEncryptedElements>
```

593   The following describes the attributes and elements listed in the schema outlined above:

594   /sp:ContentEncryptedElements

595   This assertion specifies the parts of the message that need confidentiality protection. Any such
596   elements are subject to #Content encryption.

597   /sp:ContentEncryptedElements/@XPathVersion

598   This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
599   attribute is provided, then XPath 1.0 is assumed.

600   /sp:ContentEncryptedElements/sp:XPath

601   This element contains a string specifying an XPath expression that identifies the nodes to be
602   confidentiality protected. The XPath expression is evaluated against the S:Envelope element
603   node of the message. Multiple instances of this element MAY appear within this assertion and
604   SHOULD be treated as separate references.

605   4.3 Required Elements Assertion

606   A mechanism is defined for specifying, using XPath expressions, the set of header elements that a
607   message MUST contain.

608

609 Note: Specifications are expected to provide domain specific assertions that specify which headers are
610 expected in a message. This assertion is provided for cases where such domain specific assertions have
611 not been defined.

4.3.1 RequiredElements Assertion

613 The RequiredElements assertion is used to specify header elements that the message MUST contain.
614 This assertion specifies no security requirements.

615

616 There MAY be multiple RequiredElements assertions present. Multiple RequiredElements assertions
617 present within a policy alternative are equivalent to a single RequiredElements assertion containing the
618 union of all specified XPath expressions.

619 **Syntax**

```
620  <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
621    <sp:XPath>xs:string</sp:XPath> +
622    ...
623  </sp:RequiredElements>
```

624

625 The following describes the attributes and elements listed in the schema outlined above:

626 /sp:RequiredElements

627   This assertion specifies the headers elements that MUST appear in a message.

628 /sp:RequiredElements/@XPathVersion

629   This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
630   attribute is provided, then XPath 1.0 is assumed.

631 /sp:RequiredElements/sp:XPath

632   This element contains a string specifying an XPath expression that identifies the header elements
633   that a message MUST contain. The XPath expression is evaluated against the
634   S:Envelope/S:Header element node of the message. Multiple instances of this element MAY
635   appear within this assertion and SHOULD be treated as a combined XPath expression.

4.3.2 RequiredParts Assertion

637 RequiredParts is a QName based alternative to the RequiredElements assertion (which is based on
638 XPATH) for specifying header elements that MUST be present in the message. This assertion specifies
639 no security requirements.

640

641 There MAY be multiple RequiredParts assertions present. Multiple RequiredParts assertions present
642 within a policy alternative are equivalent to a single RequiredParts assertion containing the union of all
643 specified Header elements.

644 **Syntax**

```
645  <sp:RequiredParts XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
646    <sp:Header  Name ="..."  Namespace= "..." /> +
647  </sp:RequiredParts>
```

648

649 The following describes the attributes and elements listed in the schema outlined above:

650 /sp:RequiredParts/sp:Header

651   This assertion specifies the headers elements that MUST be present in the message.

652 /sp:RequiredParts/sp:Header/@Name

653   This REQUIRED attribute indicates the local name of the SOAPHeader that needs to be present
654   in the message.

655   /sp:RequiredParts/sp:Header/@Namespace

656   This REQUIRED attribute indicates the namespace of the SOAP header that needs to be present
657   in the message.

5 Token Assertions

Token assertions specify the type of tokens to use to protect or bind tokens and claims to the message. These assertions do not recommend usage of a Policy Subject. Assertions which contain them SHOULD recommend a policy attachment point. With the exception of transport token assertions, the token assertions defined in this section are not specific to any particular security binding.

5.1 Token Inclusion

Any token assertion MAY also carry an OPTIONAL `sp:IncludeToken` attribute. The schema type of this attribute is `xs:anyURI`. This attribute indicates whether the token SHOULD be included, that is written, in the message or whether cryptographic operations utilize an external reference mechanism to refer to the key represented by the token. This attribute is defined as a global attribute in the WS-SecurityPolicy namespace and is intended to be used by any specification that defines token assertions.

5.1.1 Token Inclusion Values

The following table describes the set of valid token inclusion mechanisms supported by this specification:

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never	The token MUST NOT be included in any messages sent between the initiator and the recipient; rather, an external reference to the token SHOULD be used.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Once	The token MUST be included in only one message sent from the initiator to the recipient. References to the token MAY use an internal reference mechanism. Subsequent related messages sent between the recipient and the initiator MAY refer to the token using an external reference mechanism.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient	The token MUST be included in all messages sent from initiator to the recipient. The token MUST NOT be included in messages sent from the recipient to the initiator.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToInitiator	The token MUST be included in all messages sent from the recipient to the initiator. The token MUST NOT be included in messages sent from the initiator to the recipient.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Always	The token MUST be included in all messages sent between the initiator and the recipient. This is the default behavior.

Note: In examples, the namespace URI is replaced with "..." for brevity. For example, .../IncludeToken/Never is actually http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never. Other token inclusion URI values MAY be defined but are out-of-scope of this specification.

The default behavior characteristics defined by this specification if this attribute is not specified on a token assertion are .../IncludeToken/Always.

5.1.2 Token Inclusion and Token References

A token assertion MAY carry a sp:IncludeToken attribute that requires that the token be included in the
message. The Web Services Security specifications [WSS10, WSS11] define mechanisms for how tokens
are included in a message.

Several Token assertions (see Section 5.3) support mechanisms for referencing tokens in addition to
Direct References, for example external URI references or references using a Thumbprint.

Certain combination of sp:IncludeToken value and token reference assertions can result in a token
appearing in a message more than once. For example, if a token assertion carries a sp:IncludeToken
attribute with a value of '.../Always' and that token assertion also contains a nested
sp:RequireEmbeddedTokenReference (see Section 5.3.3) assertion, then the token would be included
twice in the message. While such combinations are not in error, they are probably best avoided for
efficiency reasons.

If a token assertion contains multiple reference assertions, then references to that token are REQUIRED
to contain all the specified reference types. For example, if a token assertion contains nested
sp:RequireIssuerSerialReference and sp:RequireThumbprintReference assertions then references to that
token contain both reference forms. Again, while such combinations are not in error, they are probably
best avoided for efficiency reasons.

5.2 Token Issuer and Required Claims

5.2.1 Token Issuer

Any token assertion MAY also carry an OPTIONAL sp:Issuer element. The schema type of this element is
wsa:EndpointReferenceType. This element indicates the token issuing authority by pointing to the issuer
endpoint address. This element is defined as a global element in the WS-SecurityPolicy namespace and
is intended to be used by any specification that defines token assertions.

5.2.2 Token Issuer Name

Any token assertion MAY also carry an OPTIONAL sp:IssuerName element. The schema type of this
element is xs:anyURI. This element indicated the token issuing authority by pointing to the issuer by using
its logical name. This element is defined as a global element in the WS-SecurityPolicy namespace and is
intended to be used by any specification that defines token assertions.

It is out of scope of this specification how the relationship between the issuer's logical name and the
physical manifestation of the issuer in the security token is defined.
While both sp:Issuer and sp:IssuerName elements are OPTIONAL they are also mutually exclusive and
cannot be specified both at the same time.

5.2.3 Required Claims

Any token assertion MAY also carry an OPTIONAL wst:Claims element. The element content is defined in
the WS-Trust namespace. This specification does not further define or limit the content of this element or
the wst:Claims/@Dialect attribute as it is out of scope of this document.

This element indicates the REQUIRED claims that the security token must contain in order to satisfy the
requirements of the token assertion.

Individual token assertions MAY further limit what claims MAY be specified for that specific token
assertion.

5.2.4 Processing Rules and Token Matching

The sender is free to compose the requirements expressed by token assertions inside the receiver's policy to as many tokens as it sees fit. As long as the union of all tokens in the received message contains the REQUIRED set of claims from REQUIRED token issuers the message is valid according to the receiver's policy.

For example if the receiver's policy contains two token assertions, one requires IssuedToken from issuer A with claims C1 and C2 and the second requires IssuedToken from issuer B with claims C3 and C4, the sender can satisfy such requirements with any of the following security token decomposition:

1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and T2 is issued by issuer B and contains claims C3 and C4.
2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is also issued by issuer A and contains claim C2 and T3 is issued by issuer B and contains claims C3 and C4.
3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2, T2 is issued by issuer B and contains claim C3 and T3 is also issued by issuer B and contains claim C4.
4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is also issued by issuer A and contains claim C2, T3 is issued by issuer B and contains claim C3 and T4 is also issued by issuer B and contains claim C4.

5.3 Token Properties

5.3.1 [Derived Keys] Property

This boolean property specifies whether derived keys SHOULD be used as defined in WS-SecureConversation. If the value is 'true', derived keys MUST be used. If the value is 'false', derived keys MUST NOT be used. The value of this property applies to a specific token. The value of this property is populated by assertions specific to the token. The default value for this property is 'false'.

See the [Explicit Derived Keys] and [Implied Derived Key] properties below for information on how particular forms of derived keys are specified.

Where the key material associated with a token is asymmetric, this property applies to the use of symmetric keys encrypted with the key material associated with the token.

5.3.2 [Explicit Derived Keys] Property

This boolean property specifies whether Explicit Derived Keys (see Section 7 of [WS-SecureConversation]) are allowed. If the value is 'true' then Explicit Derived Keys MAY be used. If the value is 'false' then Explicit Derived Keys MUST NOT be used.

5.3.3 [Implied Derived Keys] Property

This boolean property specifies whether Implied Derived Keys (see Section 7.3 of [WS-SecureConversation]) are allowed. If the value is 'true' then Implied Derived Keys MAY be used. If the value is 'false' then Implied Derived Keys MUST NOT be used.

5.4 Token Assertion Types

The following sections describe the token assertions defined as part of this specification.

5.4.1 UsernameToken Assertion

This element represents a requirement to include a username token.

763   There are cases where encrypting the UsernameToken is reasonable. For example:

764   1.   When transport security is not used.

765   2.   When a plaintext password is used.

766   3.   When a weak password hash is used.

767   4.   When the username needs to be protected, e.g. for privacy reasons.

768   When the UsernameToken is to be encrypted it SHOULD be listed as a

769   SignedEncryptedSupportingToken (Section 8.5), EndorsingEncryptedSupportingToken (Section 8.6) or

770   SignedEndorsingEncryptedSupportingToken (Section 8.7).

771

772   **Syntax**

773
```
<sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  (
    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
    <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
  <wst:Claims Dialect="..."> ... </wst:Claims> ?
  <wsp:Policy xmlns:wsp="...">
    ((
      <sp:NoPassword ... /> |
      <sp:HashPassword ... />
    ) |
    (
      <sp13:Created .../> ?
      <sp13:Nonce .../> ?
    )) ?
    (
      <sp:RequireDerivedKeys /> |
      <sp:RequireImpliedDerivedKeys ... /> |
      <sp:RequireExplicitDerivedKeys ... />
    ) ?
    (
      <sp:WssUsernameToken10 ... /> |
      <sp:WssUsernameToken11 ... />
    ) ?
    ...
  </wsp:Policy>
  ...
</sp:UsernameToken>
```

801

802   The following describes the attributes and elements listed in the schema outlined above:

803   /sp:UsernameToken

804   This identifies a UsernameToken assertion.

805   /sp:UsernameToken/@sp:IncludeToken

806   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

807   /sp:UsernameToken/sp:Issuer

808   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
809   of the sp:UsernameToken.

810   /sp:UsernameToken/sp:IssuerName

811   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:UsernameToken
812   issuer.

813   /sp:UsernameToken/wst:Claims

814   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
815   order to satisfy the token assertion requirements.

816 /sp:UsernameToken/wsp:Policy

817   This REQUIRED element identifies additional requirements for use of the sp:UsernameToken
818   assertion.

819 /sp:UsernameToken/wsp:Policy/sp:NoPassword

820   This OPTIONAL element is a policy assertion that indicates that the wsse:Password element
821   MUST NOT be present in the Username token.

822 /sp:UsernameToken/wsp:Policy/sp:HashPassword

823   This OPTIONAL element is a policy assertion that indicates that the wsse:Password element
824   MUST be present in the Username token and that the content of the wsse:Password element
825   MUST contain a hash of the timestamp, nonce and password as defined in [WSS: Username
826   Token Profile].

827 /sp13:UsernameToken/wsp:Policy/sp13:Created

828   This OPTIONAL element is a policy assertion that MUST only be used with the default clear text
829   password case, and, if present, indicates that the wsse:Created element MUST be present in the
830   Username token.

831 /sp13:UsernameToken/wsp:Policy/sp13:Nonce

832   This OPTIONAL element is a policy assertion that MUST only be used with the default clear text
833   password case, and, if present, that indicates that the wsse:Nonce element MUST be present in
834   the Username token.

835 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

836   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
837   and [Implied Derived Keys] properties for this token to 'true'.

838 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

839   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
840   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
841   'false'.

842 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

843   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
844   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
845   'false'.

846 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

847   This OPTIONAL element is a policy assertion that indicates that a Username token should be
848   used as defined in [WSS:UsernameTokenProfile1.0].

849 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

850   This OPTIONAL element is a policy assertion that indicates that a Username token should be
851   used as defined in [WSS:UsernameTokenProfile1.1].

852 5.4.2 ICreatessuedToken Assertion

853 This element represents a requirement for an issued token, which is one issued by some token issuer
854 using the mechanisms defined in WS-Trust. This assertion is used in 3rd party scenarios. For example,
855 the initiator may need to request a SAML token from a given token issuer in order to secure messages
856 sent to the recipient.

857 **Syntax**

```
858    <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
859      (
860      <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
861      <sp:IssuerName>xs:anyURI</sp:IssuerName>
862      ) ?
863      <wst:Claims Dialect="..."> ... </wst:Claims> ?
864      <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
865        ...
866      </sp:RequestSecurityTokenTemplate>
867      <wsp:Policy xmlns:wsp="...">
868        (
869          <sp:RequireDerivedKeys ... /> |
870          <sp:RequireImpliedDerivedKeys ... /> |
871          <sp:RequireExplicitDerivedKeys ... />
872        ) ?
873        <sp:RequireExternalReference ... /> ?
874        <sp:RequireInternalReference ... /> ?
875        ...
876      </wsp:Policy>
877      ...
878    </sp:IssuedToken>
```

879   The following describes the attributes and elements listed in the schema outlined above:

880   /sp:IssuedToken

881   This identifies an IssuedToken assertion.

882   /sp:IssuedToken/@sp:IncludeToken

883   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

884   /sp:IssuedToken/sp:Issuer

885   This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
886   for the issued token.

887   /sp:IssuedToken/sp:IssuerName

888   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:IssuedToken
889   issuer.

890   /sp:IssuedToken/wst:Claims

891   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
892   order to satisfy the token assertion requirements.

893   /sp:IssuedToken/sp:RequestSecurityTokenTemplate

894   This REQUIRED element contains elements which MUST be copied into the
895   wst:SecondaryParameters of the RST request sent to the specified issuer. Note: the initiator is
896   NOT REQUIRED to understand the contents of this element.

897   See Appendix B for details of the content of this element.

898   /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

899   This OPTIONAL attribute contains a WS-Trust specification namespace URI identifying the
900   version of WS-Trust referenced by the contents of this element. For example, when using Trust
901   1.3 the URI http://docs.oasis-open.org/ws-sx/ws-trust/200512 should be used and when using
902   Trust 1.4 the URI http://docs.oasis-open.org/ws-sx/ws-trust/200802 should be used.

903   /sp:IssuedToken/wsp:Policy

904   This REQUIRED element identifies additional requirements for use of the sp:IssuedToken
905   assertion.

906   /sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys

907   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
908   and [Implied Derived Keys]   properties for this token to 'true'.

909 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

910   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
911   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
912   'false'.

913 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

914   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
915   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
916   'false'.

917 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

918   This OPTIONAL element is a policy assertion that indicates whether an internal reference is
919   REQUIRED when referencing this token.
920   Note: This reference will be supplied by the issuer of the token.

921 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

922   This OPTIONAL element is a policy assertion that indicates whether an external reference is
923   REQUIRED when referencing this token.
924   Note: This reference will be supplied by the issuer of the token.

925 Note: The IssuedToken MAY or MAY NOT be associated with key material and such key material may be
926 symmetric or asymmetric. The Binding assertion will imply the type of key associated with this token.
927 Services MAY also include information in the `sp:RequestSecurityTokenTemplate` element to
928 explicitly define the expected key type. See Appendix B for details of the
929 `sp:RequestSecurityTokenTemplate` element.

5.4.3 X509Token Assertion

931 This element represents a requirement for a binary security token carrying an X509 token.

932 **Syntax**

```
933    <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
934      (
935        <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
936        <sp:IssuerName>xs:anyURI</sp:IssuerName>
937      ) ?
938      <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```
939      <wsp:Policy xmlns:wsp="...">
940        (
941          <sp:RequireDerivedKeys ... /> |
942          <sp:RequireExplicitDerivedKeys ... /> |
943          <sp:RequireImpliedDerivedKeys ... />
944        ) ?
945        <sp:RequireKeyIdentifierReference ... /> ?
946        <sp:RequireIssuerSerialReference ... /> ?
947        <sp:RequireEmbeddedTokenReference ... /> ?
948        <sp:RequireThumbprintReference ... /> ?
949        (
950          <sp:WssX509V3Token10 ... /> |
951          <sp:WssX509Pkcs7Token10 ... /> |
952          <sp:WssX509PkiPathV1Token10 ... /> |
953          <sp:WssX509V1Token11 ... /> |
954          <sp:WssX509V3Token11 ... /> |
955          <sp:WssX509Pkcs7Token11 ... /> |
956          <sp:WssX509PkiPathV1Token11 ... />
957        ) ?
958        ...
959      </wsp:Policy>
960      ...
961    </sp:X509Token>
```

963   The following describes the attributes and elements listed in the schema outlined above:

964   /sp:X509Token

965   This identifies an X509Token assertion.

966   /sp:X509Token/@sp:IncludeToken

967   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

968   /sp:X509Token/sp:Issuer

969   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
970   of the sp:X509Token.

971   /sp:X509Token/sp:IssuerName

972   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:X509Token
973   issuer.

974   /sp:X509Token/wst:Claims

975   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
976   order to satisfy the token assertion requirements.

977   /sp:X509Token/wsp:Policy

978   This REQUIRED element identifies additional requirements for use of the sp:X509Token
979   assertion.

980   /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

981   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
982   and [Implied Derived Keys] properties for this token to 'true'.

983   /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

984   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
985   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
986   'false'.

987   /sp:X509Token/wsp:Policy/sp:RequireImpliedDerivedKeys

988
989
990
This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

991 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

992
993
This OPTIONAL element is a policy assertion that indicates that a key identifier reference is REQUIRED when referencing this token.

994 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

995
996
This OPTIONAL element is a policy assertion that indicates that an issuer serial reference is REQUIRED when referencing this token.

997 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

998
999
This OPTIONAL element is a policy assertion that indicates that an embedded token reference is REQUIRED when referencing this token.

1000 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1001
1002
This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is REQUIRED when referencing this token.

1003 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1004
1005
This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should be used as defined in [WSS:X509TokenProfile1.0].

1006 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1007
1008
This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be used as defined in [WSS:X509TokenProfile1.0].

1009 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1010
1011
This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1 token should be used as defined in [WSS:X509TokenProfile1.0].

1012 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1013
1014
This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should be used as defined in [WSS:X509TokenProfile1.1].

1015 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1016
1017
This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should be used as defined in [WSS:X509TokenProfile1.1].

1018 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1019
1020
This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be used as defined in [WSS:X509TokenProfile1.1].

1021 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1022
1023
This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1 token should be used as defined in [WSS:X509TokenProfile1.1].

5.4.4 KerberosToken Assertion

1025 This element represents a requirement for a Kerberos token [WSS:KerberosToken1.1].

1026 **Syntax**

```
<sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  (
    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
    <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
```

```
1032        <wst:Claims Dialect="..."> ... </wst:Claims> ?
1033        <wsp:Policy xmlns:wsp="...">
1034          (
1035            <sp:RequireDerivedKeys ... /> |
1036            <sp:RequireImpliedDerivedKeys ... /> |
1037            <sp:RequireExplicitDerivedKeys ... />
1038          ) ?
1039          <sp:RequireKeyIdentifierReference ... /> ?
1040          (
1041            <sp:WssKerberosV5ApReqToken11 ... /> |
1042            <sp:WssGssKerberosV5ApReqToken11 ... />
1043          ) ?

1045          ...
1046        </wsp:Policy>
1047        ...
1048      </sp:KerberosToken>
```

1050   The following describes the attributes and elements listed in the schema outlined above:

1051   /sp:KerberosToken

1052   This identifies a KerberosV5ApReqToken assertion.

1053   /sp:KerberosToken/@sp:IncludeToken

1054   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1055   /sp:KerberosToken/sp:Issuer

1056   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1057   of the sp:KerberosToken.

1058   /sp:KerberosToken/sp:IssuerName

1059   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken
1060   issuer.

1061   /sp:KerberosToken/wst:Claims

1062   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1063   order to satisfy the token assertion requirements.

1064   /sp:KerberosToken/wsp:Policy

1065   This REQUIRED element identifies additional requirements for use of the sp:KerberosToken
1066   assertion.

1067   /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1068   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1069   and [Implied Derived Keys] properties for this token to 'true'.

1070   /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1071   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1072   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1073   'false'.

1074   /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1075   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1076   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1077   'false'.

1078   /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

| 1079 | This OPTIONAL element is a policy assertion that indicates that a key identifier reference is |
| 1080 | REQUIRED when referencing this token. |

1081 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

1082 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ
1083 token should be used as defined in [WSS:KerberosTokenProfile1.1].

1084 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1085 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-
1086 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

5.4.5 SpnegoContextToken Assertion

1088 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
1089 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1090 **Syntax**

```
1091 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1092   (
1093   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1094   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1095   ) ?
1096   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1097   <wsp:Policy xmlns:wsp="...">
1098     (
1099       <sp:RequireDerivedKeys ... /> |
1100       <sp:RequireImpliedDerivedKeys ... /> |
1101       <sp:RequireExplicitDerivedKeys ... />
1102     ) ?
1103     <sp:MustNotSendCancel ... /> ?
1104     <sp:MustNotSendAmend ... /> ?
1105     <sp:MustNotSendRenew ... /> ?
1106     ...
1107   </wsp:Policy>
1108   ...
1109 </sp:SpnegoContextToken>
```

1110

1111 The following describes the attributes and elements listed in the schema outlined above:

1112 /sp:SpnegoContextToken

1113 This identifies a SpnegoContextToken assertion.

1114 /sp:SpnegoContextToken/@sp:IncludeToken

1115 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1116 /sp:SpnegoContextToken/sp:Issuer

1117 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
1118 for the Spnego Context Token.

1119 /sp:SpnegoContextToken/sp:IssuerName

1120 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
1121 sp:SpnegoContextToken issuer.

1122 /sp:SpnegoContextToken/wst:Claims

1123 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1124 order to satisfy the token assertion requirements.

1125 /sp:SpnegoContextToken/wsp:Policy

1126   This REQUIRED element identifies additional requirements for use of the
1127   sp:SpnegoContextToken assertion.

1128 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1129   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1130   and [Implied Derived Keys] properties for this token to 'true'.

1131 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1132   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1133   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1134   'false'.

1135 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1136   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1137   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1138   'false'.

1139 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1140   This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
1141   token does not support SCT/Cancel RST messages. If this assertion is missing it means that
1142   SCT/Cancel RST messages are supported by the STS.

1143 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1144   This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
1145   token does not support SCT/Amend RST messages. If this assertion is missing it means that
1146   SCT/Amend RST messages are supported by the STS.

1147 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1148   This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
1149   token does not support SCT/Renew RST messages. If this assertion is missing it means that
1150   SCT/Renew RST messages are supported by the STS.

1151 5.4.6 SecurityContextToken Assertion

1152 This element represents a requirement for a SecurityContextToken token.

1153 **Syntax**

```
1154    <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1155    (
1156       <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1157       <sp:IssuerName>xs:anyURI</sp:IssuerName>
1158    ) ?
1159    <wst:Claims Dialect="..."> ... </wst:Claims> ?
1160    <wsp:Policy xmlns:wsp="...">
1161       (
1162         <sp:RequireDerivedKeys ... /> |
1163         <sp:RequireImpliedDerivedKeys ... /> |
1164         <sp:RequireExplicitDerivedKeys ... />
1165       ) ?
1166       <sp:RequireExternalUriReference ... /> ?
1167       <sp:SC13SecurityContextToken... /> ?
1168       ...
1169    </wsp:Policy>
1170    ...
1171    </sp:SecurityContextToken>
```

1172

1173 The following describes the attributes and elements listed in the schema outlined above:

1174 /sp:SecurityContextToken

1175   This identifies a SecurityContextToken assertion.

1176   /sp:SecurityContextToken/@sp:IncludeToken

1177   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1178   /sp:SecurityContextToken/sp:Issuer

1179   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1180   of the sp:SecurityContextToken.

1181   /sp:SecurityContextToken/sp:IssuerName

1182   This OPTIONAL element, of type xs:anyURI, contains the logical name of the
1183   sp:SecurityContextToken issuer.

1184   /sp:SecurityContextToken/wst:Claims

1185   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1186   order to satisfy the token assertion requirements.

1187   /sp:SecurityContextToken/wsp:Policy

1188   This REQUIRED element identifies additional requirements for use of the
1189   sp:SecurityContextToken assertion.

1190   /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1191   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1192   and [Implied Derived Keys] properties for this token to 'true'.

1193   /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1194   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1195   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1196   'false'.

1197   /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1198   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1199   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1200   'false'.

1201   /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1202   This OPTIONAL element is a policy assertion that indicates that an external URI reference is
1203   REQUIRED when referencing this token.

1204   /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1205   This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
1206   be used as defined in [WS-SecureConversation].

1207

1208   Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
1209   both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
1210   a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
1211   sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

5.4.7 SecureConversationToken Assertion

1213   This element represents a requirement for a Security Context Token retrieved from the indicated issuer
1214   address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
1215   service endpoint address.

1216

1217 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1218 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD
1219 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1220 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1221 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1222 below.

1223

1224

1225 If the bootstrap policy assertion is used to indicate the security binding and policy in effect when
1226 requesting a secure conversation token from the target service, then subsequent Amend, Renew and
1227 Cancel messages MUST comply with the following rules.

Amending Context

1229 To amend an existing secure conversation token, a requestor uses the context amending mechanism as
1230 described by the WS-SecureConversation specification. The message exchange MUST be secured
1231 using the existing (to be amended) SCT in accordance with the target service (outer) policy, combined
1232 with endorsing supporting tokens carrying the new claims to be associated with the amended context with
1233 the inclusion mode set to:

1234 `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient`

1235 See the EndorsingSupportingTokens Assertion section for more details on the usage of the endorsing
1236 supporting tokens.

Renewing Context

1238 To renew an existing secure conversation token, a requestor uses the context renewal mechanism as
1239 described by the WS-SecureConversation specification. The message exchange MUST be secured
1240 according to the requirements of the bootstrap policy assertion, combined with the existing (to be
1241 renewed) SCT used as an endorsing supporting token with the inclusion mode set to:

1242 `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient`

1243 See the EndorsingSupportingTokens Assertion section for more details on the usage of endorsing
1244 support tokens.

Canceling Context

1246 To cancel an existing secure conversation token, a requestor uses the context cancelling mechanism as
1247 described by the WS-SecureConversation specification. The message exchange MUST be secured
1248 using the existing (to be cancelled) SCT in accordance with the target service (outer) policy.

Handling Policy Alternatives

1250 If there are policy alternatives present in either the bootstrap policy assertion or the target service (outer)
1251 policy assertion, the following rules MUST be followed.

1252 • The policy alternative used as a basis for the context renewal MUST be the same as the policy
1253   alternative which was previously used for the context issuance.

1254 • If the target service (outer) policy has policy alternatives and SecureConversationToken assertion
1255   appears in multiple alternatives as follows:
1256   Policy
1257   Policy-alternative-1
1258   SecureConversationToken-assertion-1
1259   Policy-alternative-2
1260   SecureConversationToken-assertion-2

1261 The policy alternative used as basis for context amend and cancel MUST be the same as the policy
1262 alternative that was used to obtain the context. This means that Policy-alternative-1 above cannot be
1263 used to amend and cancel SecureConversationToken-assertion-2 and vice-versa.

1264 • If the target service (outer) policy has policy alternatives that are outside the
1265   SecureConversationToken assertion as follows:
1266   Policy
1267   SecureConversationToken-assertion-1
1268   Policy-alternative-1
1269   Policy-alternative-2

1270 Any policy alternative can be used to amend or cancel the context. This means that either Policy-
1271 alternative-1 or Policy-alternative-2 can be used to amend or cancel SecureConversationToken-
1272 assertion-1.

1273

1274 **Syntax**

```
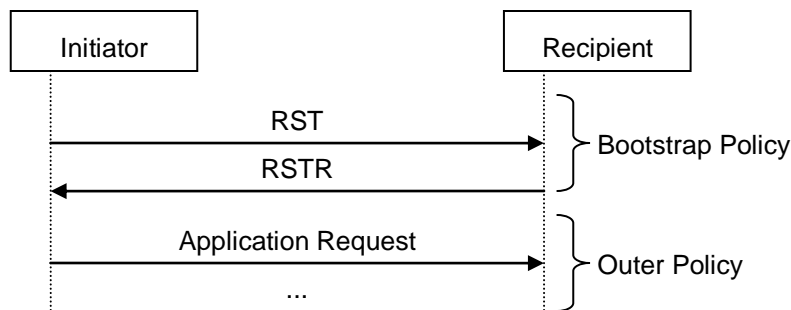1275 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1276   (
1277   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1278   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1279   ) ?
1280   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1281   <wsp:Policy xmlns:wsp="...">
1282     (
1283       <sp:RequireDerivedKeys ... /> |
1284       <sp:RequireImpliedDerivedKeys ... /> |
1285       <sp:RequireExplicitDerivedKeys ... />
1286     ) ?
1287     <sp:RequireExternalUriReference ... /> ?
1288     <sp:SC13SecurityContextToken ... /> ?
1289     <sp:MustNotSendCancel ... /> ?
1290     <sp:MustNotSendAmend ... /> ?
1291     <sp:MustNotSendRenew ... /> ?
1292     <sp:BootstrapPolicy ... >
1293       <wsp:Policy> ... </wsp:Policy>
1294     </sp:BootstrapPolicy> ?
1295   </wsp:Policy>
1296   ...
1297 </sp:SecureConversationToken>
```

1298

1299 The following describes the attributes and elements listed in the schema outlined above:

1300 /sp:SecureConversationToken

1301   This identifies a SecureConversationToken assertion.

1302 /sp:SecureConversationToken/@sp:IncludeToken

1303   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1304 /sp:SecureConversationToken/sp:Issuer

1305   This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
1306   for the Security Context Token.

1307 /sp:SecureConversationToken/sp:IssuerName

1308   This OPTIONAL element, of type xs:anyURI, contains the logical name of the
1309   sp:SecureConversationToken issuer.

1310 /sp:SpnegoContextToken/wst:Claims

1311   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1312   order to satisfy the token assertion requirements.

1313 /sp:SecureConversationToken/wsp:Policy

1314   This REQUIRED element identifies additional requirements for use of the
1315   sp:SecureConversationToken assertion.

1316 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1317   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1318   and [Implied Derived Keys] properties for this token to 'true'.

1319 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1320   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1321   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1322   'false'.

1323 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1324   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1325   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1326   'false'.

1327 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1328   This OPTIONAL element is a policy assertion that indicates that an external URI reference is
1329   REQUIRED when referencing this token.

1330 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken

1331   This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
1332   be used as obtained using the protocol defined in [WS-SecureConversation].

1333 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

1334   This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1335   conversation token does not support SCT/Cancel RST messages. If this assertion is missing it
1336   means that SCT/Cancel RST messages are supported by the STS.

1337 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

1338   This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1339   conversation token does not support SCT/Amend RST messages. If this assertion is missing it
1340   means that SCT/Amend RST messages are supported by the STS.

1341 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1342   This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1343   conversation token does not support SCT/Renew RST messages. If this assertion is missing it
1344   means that SCT/Renew RST messages are supported by the STS.

1345 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1346   This OPTIONAL element is a policy assertion that contains the policy indicating the requirements
1347   for obtaining the Security Context Token.

1348 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1349   This element contains the security binding requirements for obtaining the Security Context Token.
1350   It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
1351   protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
1352   are to be protected.

1353 **Example**

```
1354  <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1355    <sp:SymmetricBinding>
1356      <wsp:Policy>
1357        <sp:ProtectionToken>
1358          <wsp:Policy>
1359            <sp:SecureConversationToken>
1360              <sp:Issuer>
1361                <wsa:Address>http://example.org/sts</wsa:Address>
1362              </sp:Issuer>
1363              <wsp:Policy>
1364                <sp:SC13SecurityContextToken />
1365                <sp:BootstrapPolicy>
1366                  <wsp:Policy>
1367                    <sp:AsymmetricBinding>
1368                      <wsp:Policy>
1369                        <sp:InitiatorToken>
1370                          ...
1371                        </sp:InitiatorToken>
1372                        <sp:RecipientToken>
1373                          ...
1374                        </sp:RecipientToken>
1375                      </wsp:Policy>
1376                    </sp:AsymmetricBinding>
1377                    <sp:SignedParts>
1378                      ...
1379                    </sp:SignedParts>
1380                    ...
1381                  </wsp:Policy>
1382                </sp:BootstrapPolicy>
1383              </wsp:Policy>
1384            </sp:SecureConversationToken>
1385          </wsp:Policy>
1386        </sp:ProtectionToken>
1387        ...
1388      </wsp:Policy>
1389    </sp:SymmetricBinding>
1390    <sp:SignedParts>
1391    ...
1392    </sp:SignedParts>
1393    ...
1394  </wsp:Policy>
```

5.4.8 SamlToken Assertion

1396 This element represents a requirement for a SAML token.

1397 **Syntax**

```
1398  <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1399    (
1400      <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1401      <sp:IssuerName>xs:anyURI</sp:IssuerName>
1402    ) ?
1403    <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```
     <wsp:Policy xmlns:wsp="...">
       (
         <sp:RequireDerivedKeys ... /> |
         <sp:RequireImpliedDerivedKeys ... /> |
         <sp:RequireExplicitDerivedKeys ... />
       ) ?
       <sp:RequireKeyIdentifierReference ... /> ?
       (
         <sp:WssSamlV11Token10 ... /> |
         <sp:WssSamlV11Token11 ... /> |
         <sp:WssSamlV20Token11 ... />
       ) ?
       ...
     </wsp:Policy>
     ...
   </sp:SamlToken>
```

The following describes the attributes and elements listed in the schema outlined above:

/sp:SamlToken

> This identifies a SamlToken assertion.

/sp:SamlToken/@sp:IncludeToken

> This OPTIONAL attribute identifies the token inclusion value for this token assertion.

/sp:SamlToken/sp:Issuer

> This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer of the sp:SamlToken.

/sp:SamlToken/sp:IssuerName

> This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken issuer.

/sp:SamlToken/wst:Claims

> This OPTIONAL element identifies the REQUIRED claims that a security token must contain in order to satisfy the token assertion requirements.

/sp:SamlToken/wsp:Policy

> This REQUIRED element identifies additional requirements for use of the sp:SamlToken assertion.

/sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

> This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

/sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

> This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.

/sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys

> This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

/sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

> This OPTIONAL element is a policy assertion that indicates that a key identifier reference is REQUIRED when referencing this token.

1452 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10

1453   This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1454   should be used as defined in [WSS:SAMLTokenProfile1.0].

1455 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11

1456   This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1457   should be used as defined in [WSS:SAMLTokenProfile1.1].

1458 /sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11

1459   This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token
1460   should be used as defined in [WSS:SAMLTokenProfile1.1].

1461

1462 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1463 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1464 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion
1465 SHOULD be used instead.

5.4.9 RelToken Assertion

1467 This element represents a requirement for a REL token.

1468 **Syntax**

```
1469  <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1470    (
1471      <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1472      <sp:IssuerName>xs:anyURI</sp:IssuerName>
1473    ) ?
1474    <wst:Claims Dialect="..."> ... </wst:Claims> ?
1475    <wsp:Policy xmlns:wsp="...">
1476      (
1477        <sp:RequireDerivedKeys ... /> |
1478        <sp:RequireImpliedDerivedKeys ... /> |
1479        <sp:RequireExplicitDerivedKeys ... />
1480      ) ?
1481      <sp:RequireKeyIdentifierReference ... /> ?
1482      (
1483        <sp:WssRelV10Token10 ... /> |
1484        <sp:WssRelV20Token10 ... /> |
1485        <sp:WssRelV10Token11 ... /> |
1486        <sp:WssRelV20Token11 ... />
1487      ) ?
1488      ...
1489    </wsp:Policy>
1490    ...
1491  </sp:RelToken>
```

1492

1493 The following describes the attributes and elements listed in the schema outlined above:

1494 /sp:RelToken

1495   This identifies a RelToken assertion.

1496 /sp:RelToken/@sp:IncludeToken

1497   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1498 /sp:RelToken/sp:Issuer

1499   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1500   of the sp:RelToken.

1501 /sp:RelToken/sp:IssuerName

1502   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken
1503   issuer.

1504 /sp:RelToken/wst:Claims

1505   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1506   order to satisfy the token assertion requirements.

1507 /sp:RelToken/wsp:Policy

1508   This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

1509 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1510   This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1511   and [Implied Derived Keys] property for this token to 'true'.

1512 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1513   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1514   Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1515   'false'.

1516 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1517   This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1518   Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1519   'false'.

1520 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1521   This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1522   REQUIRED when referencing this token.

1523 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1524   This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
1525   be used as defined in [WSS:RELTokenProfile1.0].

1526 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1527   This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
1528   be used as defined in [WSS:RELTokenProfile1.0].

1529 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1530   This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
1531   be used as defined in [WSS:RELTokenProfile1.1].

1532 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1533   This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
1534   be used as defined in [WSS:RELTokenProfile1.1].

1535

1536 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
1537 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1538 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion
1539 SHOULD be used instead.

5.4.10 HttpsToken Assertion

1541 This element represents a requirement for a transport binding to support the use of HTTPS.

1542 **Syntax**

```
1543    <sp:HttpsToken xmlns:sp="..." ... >
1544      (
1545        <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1546        <sp:IssuerName>xs:anyURI</sp:IssuerName>
1547      ) ?
1548      <wst:Claims Dialect="..."> ... </wst:Claims> ?
1549      <wsp:Policy xmlns:wsp="...">
1550        (
1551          <sp:HttpBasicAuthentication /> |
1552          <sp:HttpDigestAuthentication /> |
1553          <sp:RequireClientCertificate /> |
1554          ...
1555        )?
1556        ...
1557      </wsp:Policy>
1558      ...
1559    </sp:HttpsToken>
```

1560   The following describes the attributes and elements listed in the schema outlined above:

1561   /sp:HttpsToken

1562   This identifies an Https assertion stating that use of the HTTPS protocol specification is
1563   supported.

1564   /sp:HttpsToken/sp:Issuer

1565   This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1566   of the sp:HttpsToken.

1567   /sp:HttpsToken/sp:IssuerName

1568   This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken
1569   issuer.

1570   /sp:HttpsToken/wst:Claims

1571   This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1572   order to satisfy the token assertion requirements.

1573   /sp:HttpsToken/wsp:Policy

1574   This REQUIRED element identifies additional requirements for use of the sp:HttpsToken
1575   assertion.

1576   /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1577   This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic
1578   Authentication [RFC2068] to authenticate to the service.

1579   /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1580   This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP
1581   Digest Authentication [RFC2068] to authenticate to the service.

1582   /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1583   This OPTIONAL element is a policy assertion that indicates that the client MUST provide a
1584   certificate when negotiating the HTTPS session.

5.4.11 KeyValueToken Assertion

1586   This element represents a requirement for a KeyValue token. The next section defines the KeyValue
1587   security token abstraction for purposes of this token assertion.
1588

1589 This document defines requirements for KeyValue token when used in combination with RSA
1590 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by
1591 introducing new nested assertions besides *sp:RsaKeyValue*.

1592 **Syntax**

```
1593  <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1594    <wsp:Policy xmlns:wsp="...">
1595      <sp:RsaKeyValue ... /> ?
1596         ...
1597    </wsp:Policy>
1598    ...
1599  </sp:KeyValueToken>
```

1600 The following describes the attributes listed in the schema outlined above:

1601 /sp:KeyValueToken

1602   This identifies a RsaToken assertion.

1603 /sp:KeyValueToken/@sp:IncludeToken

1604   This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1605 /sp:KeyValueToken/wsp:Policy

1606   This REQUIRED element identifies additional requirements for use of the sp:KeyValueToken
1607   assertion.

1608 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1609   This OPTIONAL element is a policy assertion that indicates that the ds:RSAKeyValue element
1610   must be present in the KeyValue token. This indicates that an RSA key pair must be used.

5.4.11.1 KeyValue Token

1612 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key
1613 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this
1614 section is to define the KeyValue token abstraction that represents such key pair referencing mechanism.
1615
1616 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be
1617 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*
1618 element in combination with RSA cryptographic algorithm.
1619
1620 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the
1621 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1622  <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1623    <ds:KeyValue>
1624      <ds:RSAKeyValue>
1625        <ds:Modulus>ds:CryptoBinary</ds:Modulus>
1626        <ds:Exponent>ds:CryptoBinary</ds:Exponent>
1627      </ds:RSAKeyValue>
1628    <ds:KeyValue>
1629  </ds:KeyInfo>
```

1630
1631 When the KeyValue token is used the corresponding public key value appears directly in the signature or
1632 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValue token
1633 manifestation outside the *ds:KeyInfo* element.

```
1634  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1635    <SignedInfo>
1636      <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1637  c14n#" />
1638      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1639      <Reference URI="#_1">
1640        <Transforms>
```

```
1641            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1642          </Transforms>
1643          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1644          <DigestValue>...</DigestValue>
1645        </Reference>
1646      </SignedInfo>
1647      <SignatureValue>...</SignatureValue>
1648      <KeyInfo>
1649        <KeyValue>
1650          <RSAKeyValue>
1651            <Modulus>...</Modulus>
1652            <Exponent>...</Exponent>
1653          </RSAKeyValue>
1654        </KeyValue>
1655      </KeyInfo>
1656    </Signature>
```

1657

1658 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no
1659 identifier can be associated with the token, the KeyValue token cannot be referenced by using
1660 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the KeyValue
1661 token can be used whenever a security token can be used as illustrated on the following example:

```
1662    <t:RequestSecurityToken xmlns:t="...">
1663      <t:RequestType>...</t:RequestType>
1664      ...
1665      <t:UseKey>
1666        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1667          <KeyValue>
1668            <RSAKeyValue>
1669              <Modulus>...</Modulus>
1670              <Exponent>...</Exponent>
1671            </RSAKeyValue>
1672          </KeyValue>
1673        </KeyInfo>
1674      </t:UseKey>
1675    </t:RequestSecurityToken>
```

6 Security Binding Properties

1676

1677 This section defines the various properties or conditions of a security binding, their semantics, values and
1678 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are
1679 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that
1680 populates a value of a property appears in a policy, that property is set to the value indicated by the
1681 assertion. The security binding then uses the value of the property to control its behavior. The properties
1682 listed here are common to the various security bindings described in Section 7. Assertions that define
1683 values for these properties are defined in Section 7. The following properties are used by the security
1684 binding assertions.

6.1 [Algorithm Suite] Property

1685

1686 This property specifies the algorithm suite REQUIRED for performing cryptographic operations with
1687 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and
1688 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This
1689 property defines the set of available algorithms. The value of this property is typically referenced by a
1690 security binding and is used to specify the algorithms used for all message level cryptographic operations
1691 performed under the security binding.

1692 Note: In some cases, this property MAY be referenced under a context other than a security binding and
1693 used to control the algorithms used under that context. For example, supporting token assertions define
1694 such a context. In such contexts, the specified algorithms still apply to message level cryptographic
1695 operations.

1696 An algorithm suite defines values for each of the following operations and properties:

1697 - [Sym Sig]   Symmetric Key Signature
1698 - [Asym Sig]   Signature with an asymmetric key
1699 - [Dig]   Digest
1700 - [Enc]   Encryption
1701 - [Sym KW]   Symmetric Key Wrap
1702 - [Asym KW]   Asymmetric Key Wrap
1703 - [Comp Key]   Computed key
1704 - [Enc KD]   Encryption key derivation
1705 - [Sig KD]   Signature key derivation
1706 - [Min SKL]   Minimum symmetric key length
1707 - [Max SKL]   Maximum symmetric key length
1708 - [Min AKL]   Minimum asymmetric key length
1709 - [Max AKL]   Maximum asymmetric key length

1710

1711 The following table provides abbreviations for the algorithm URI used in the table below:

Abbreviation	Algorithm URI
HmacSha1	http://www.w3.org/2000/09/xmldsig#hmac-sha1
RsaSha1	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Sha1	http://www.w3.org/2000/09/xmldsig#sha1
Sha256	http://www.w3.org/2001/04/xmlenc#sha256

Sha512	http://www.w3.org/2001/04/xmlenc#sha512
Aes128	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Aes192	http://www.w3.org/2001/04/xmlenc#aes192-cbc
Aes256	http://www.w3.org/2001/04/xmlenc#aes256-cbc
TripleDes	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
KwAes128	http://www.w3.org/2001/04/xmlenc#kw-aes128
KwAes192	http://www.w3.org/2001/04/xmlenc#kw-aes192
KwAes256	http://www.w3.org/2001/04/xmlenc#kw-aes256
KwTripleDes	http://www.w3.org/2001/04/xmlenc#kw-tripledes
KwRsaOaep	http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
KwRsa15	http://www.w3.org/2001/04/xmlenc#rsa-1_5
PSha1	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
PSha1L128	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
PSha1L192	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
PSha1L256	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
XPath	http://www.w3.org/TR/1999/REC-xpath-19991116
XPath20	http://www.w3.org/2002/06/xmldsig-filter2
C14N	http://www.w3.org/TR/2001/REC-xml-c14n-20010315
C14N11	http://www.w3.org/2006/12/xml-c14n11
ExC14N	http://www.w3.org/2001/10/xml-exc-c14n#
SNT	http://www.w3.org/TR/soap12-n11n
STRT10	http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-soap-message-security-1.0#STR-Transform
AbsXPath	http://docs.oasis-open.org/...TBD.../AbsXPath

1712

1713 The tables below show all the base algorithm suites defined by this specification. This table defines
1714 values for properties which are common for all suites:

Property	Algorithm / Value
[Sym Sig]	HmacSha1
[Asym Sig]	RsaSha1
[Comp Key]	PSha1
[Max SKL]	256
[Min AKL]	1024
[Max AKL]	4096

1715 This table defines additional properties whose values can be specified along with the default value for that
1716 property.

Property	Algorithm / Value
[C14n Algorithm]	ExC14N
[Soap Norm]	None
[STR Trans]	None
[XPath]	None

1717 This table defines values for the remaining components for each algorithm suite.

Algorithm Suite	[Dig]	[Enc]	[Sym KW]	[Asym KW]	[Enc KD]	[Sig KD]	[Min SKL]
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128

Algorithm Suite	[Dig]	[Enc]	[Sym KW]	[Asym KW]	[Enc KD]	[Sig KD]	[Min SKL]
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192
Basic256Sha256	Sha256	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192Sha256	Sha256	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128Sha256	Sha256	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDesSha256	Sha256	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Sha256Rsa15	Sha256	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Sha256Rsa15	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Sha256Rsa15	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesSha256Rsa15	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

6.2 [Timestamp] Property

This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security` header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be present. The default value for this property is 'false'.

6.3 [Protection Order] Property

This property indicates the order in which integrity and confidentiality are applied to the message, in cases where both integrity and confidentiality are REQUIRED:

EncryptBeforeSigning	Signature MUST computed over ciphertext. Encryption key and signing key MUST be derived from the same source key unless distinct keys are provided, see Section 7.5 on the AsymmetricBinding.
SignBeforeEncrypting	Signature MUST be computed over plaintext. The resulting signature SHOULD be encrypted. Supporting signatures MUST be over the plain text signature.

The default value for this property is 'SignBeforeEncrypting'.

6.4 [Signature Protection] Property

This boolean property specifies whether the signature MUST be encrypted. If the value is 'true', the primary signature MUST be encrypted and any signature confirmation elements MUST also be encrypted. The primary signature element is NOT REQUIRED to be encrypted if the value is 'true' when there is nothing in the message that is covered by this signature that is encrypted. If the value is 'false', the primary signature MUST NOT be encrypted and any signature confirmation elements MUST NOT be encrypted. The default value for this property is 'false'.

6.5 [Token Protection] Property

This boolean property specifies whether signatures MUST cover the token used to generate that signature. If the value is 'true', then each token used to generate a signature MUST be covered by that signature. If the value is 'false', then the token MUST NOT be covered by the signature. Note that in cases where derived keys are used the 'main' token, and NOT the derived key token, is covered by the signature. It is RECOMMENDED that assertions that define values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

6.6 [Entire Header and Body Signatures] Property

This boolean property specifies whether signature digests over the SOAP body and SOAP headers MUST only cover the entire body and entire header elements. If the value is 'true', then each digest over the SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In addition each digest over a SOAP header MUST be over an actual header element and not a descendant of a header element. This restriction does not specifically apply to the wsse:Security header. However signature digests over child elements of the wsse:Security header MUST be over the entire child element and not a descendent of that element. If the value is 'false', then signature digests MAY be over a descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to 'true' mitigates against some possible re-writing attacks. It is RECOMENDDED that assertions that define values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

6.7 [Security Header Layout] Property

This property indicates which layout rules to apply when adding items to the security header. The following table shows which rules are defined by this specification.

Strict	Items are added to the security header following the numbered layout rules described below according to a general principle of 'declare before use'.
Lax	Items are added to the security header in any order that conforms to WSS: SOAP Message Security
LaxTimestampFirst	As Lax except that the first item in the security header MUST be a wsu:Timestamp. Note that the [Timestamp] property MUST also be set to 'true' in this case.
LaxTimestampLast	As Lax except that the last item in the security header MUST be a wsu:Timestamp. Note that the [Timestamp] property MUST also be set to 'true' in this case.

6.7.1 Strict Layout Rules for WSS 1.0

1. Tokens that are included in the message MUST be declared before use. For example:
   a. A local signing token MUST occur before the signature that uses it.
   b. A local token serving as the source token for a derived key token MUST occur before that derived key token.
   c. A local encryption token MUST occur before the reference list that points to xenc:EncryptedData elements that use it.
   d. If the same token is used for both signing and encryption, then it SHOULD appear before the ds:Signature and xenc:ReferenceList elements in the security header that are generated using the token.
2. Signed elements inside the security header MUST occur before the signature that signs them. For example:
   a. A timestamp MUST occur before the signature that signs it.

1769   b. A Username token (usually in encrypted form) MUST occur before the signature that
1770   signs it.

1771   c. A primary signature MUST occur before the supporting token signature that signs the
1772   primary signature's signature value element.

1773   3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1774   has the same order requirements as the source plain text element, unless requirement 4
1775   indicates otherwise. For example, an encrypted primary signature MUST occur before any
1776   supporting token signature per 2.c above and an encrypted token has the same ordering
1777   requirements as the unencrypted token.

1778 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1779 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1780 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1781 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1782 Layout Rules for WSS 1.1

1783   1. Tokens that are included in the message MUST be declared before use. For example:

1784   a. A local signing token MUST occur before the signature that uses it.

1785   b. A local token serving as the source token for a derived key token MUST occur before that
1786   derived key token.

1787   c. A local encryption token MUST occur before the reference list that points to
1788   xenc:EncryptedData elements that use it.

1789   d. If the same token is used for both signing and encryption, then it SHOULD appear before
1790   the ds:Signature and xenc:ReferenceList elements in the security header that are
1791   generated using the token.

1792   2. Signed elements inside the security header MUST occur before the signature that signs them.
1793   For example:

1794   a. A timestamp MUST occur before the signature that signs it.

1795   b. A Username token (usually in encrypted form) MUST occur before the signature that
1796   signs it.

1797   c. A primary signature MUST occur before the supporting token signature that signs the
1798   primary signature's signature value element.

1799   d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.

1800   3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1801   has the same order requirements as the source plain text element, unless requirement 4
1802   indicates otherwise. For example, an encrypted primary signature MUST occur before any
1803   supporting token signature per 2.c above and an encrypted token has the same ordering
1804   requirements as the unencrypted token.

1805   4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1806   MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1807   xenc:EncryptedData elements in the security header that are referenced from the reference list.
1808   However, the xenc:ReferenceList is NOT REQUIRED to appear before independently encrypted
1809   tokens such as the xenc:EncryptedKey token as defined in WSS.

1810   5. An xenc:EncryptedKey element without an internal reference list [WSS: SOAP Message Security
1811   1.1] MUST obey rule 1 above.

7 Security Binding Assertions

1812

1813 The appropriate representation of the different facets of security mechanisms requires distilling the
1814 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy
1815 scope of assertions defined in this section is the policy scope of their containing element.

7.1 AlgorithmSuite Assertion

1816

1817 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]
1818 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

1819 **Syntax**

```
1820    <sp:AlgorithmSuite xmlns:sp="..." ... >
1821      <wsp:Policy xmlns:wsp="...">
1822       (<sp:Basic256 ... /> |
1823        <sp:Basic192 ... /> |
1824        <sp:Basic128 ... /> |
1825        <sp:TripleDes ... /> |
1826        <sp:Basic256Rsa15 ... /> |
1827        <sp:Basic192Rsa15 ... /> |
1828        <sp:Basic128Rsa15 ... /> |
1829        <sp:TripleDesRsa15 ... /> |
1830        <sp:Basic256Sha256 ... /> |
1831        <sp:Basic192Sha256 ... /> |
1832        <sp:Basic128Sha256 ... /> |
1833        <sp:TripleDesSha256 ... /> |
1834        <sp:Basic256Sha256Rsa15 ... /> |
1835        <sp:Basic192Sha256Rsa15 ... /> |
1836        <sp:Basic128Sha256Rsa15 ... /> |
1837        <sp:TripleDesSha256Rsa15 ... /> |
1838        ...)
1839        <sp:InclusiveC14N ... /> ?
1840        <sp:InclusiveC14N11 ... /> ?
1841       <sp:SOAPNormalization10 ... /> ?
1842        <sp:STRTransform10 ... /> ?
1843       (<sp:XPath10 ... /> |
1844        <sp:XPathFilter20 ... /> |
1845        <sp:AbsXPath ... /> |
1846        ...)?
1847        ...
1848      </wsp:Policy>
1849      ...
1850    </sp:AlgorithmSuite>
```

1851

1852 The following describes the attributes and elements listed in the schema outlined above:

1853 /sp:AlgorithmSuite

1854   This identifies an AlgorithmSuite assertion.

1855 /sp:AlgorithmSuite/wsp:Policy

1856   This REQUIRED element contains one or more policy assertions that indicate the specific
1857   algorithm suite to use.

1858 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1859   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1860   set to 'Basic256'.

1861   /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

1862   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1863   set to 'Basic192'.

1864   /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1865   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1866   set to 'Basic128'.

1867   /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1868   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1869   set to 'TripleDes'.

1870   /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1871   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1872   set to 'Basic256Rsa15'.

1873   /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1874   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1875   set to 'Basic192Rsa15'.

1876   /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1877   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1878   set to 'Basic128Rsa15'.

1879   /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1880   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1881   set to 'TripleDesRsa15'.

1882   /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1883   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1884   set to 'Basic256Sha256'.

1885   /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1886   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1887   set to 'Basic192Sha256'.

1888   /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1889   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1890   set to 'Basic128Sha256'.

1891   /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1892   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1893   set to 'TripleDesSha256'.

1894   /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1895   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1896   set to 'Basic256Sha256Rsa15'.

1897   /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1898   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1899   set to 'Basic192Sha256Rsa15'.

1900   /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1901   This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1902   set to 'Basic128Sha256Rsa15'.

1903   /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1904 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1905 set to 'TripleDesSha256Rsa15'.

1906 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1907 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an
1908 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]
1909 property is 'ExC14N'.

1910 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N11
1911
1912 This optional element is a policy assertion that indicates that the
1913 [C14N] property of an algorithm suite is set to 'C14N11'. Note: as
1914 indicated in Section 6.1 the default value of the [C14N] property is
1915 'ExC14N'.

1916 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1917 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set
1918 to 'SNT'.

1919 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1920 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is
1921 set to 'STRT10'.

1922 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1923 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1924 'XPath'.

1925 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1926 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1927 'XPath20'.

1928 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1929 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1930 'AbsXPath' (see AbsoluteLocationPath in [XPATH]).

1931

7.2 Layout Assertion

1933 This assertion indicates a requirement for a particular security header layout as defined under the
1934 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1935 containing assertion.

1936 **Syntax**

```
1937    <sp:Layout xmlns:sp="..." ... >
1938      <wsp:Policy xmlns:wsp="...">
1939        <sp:Strict ... /> |
1940        <sp:Lax ... /> |
1941        <sp:LaxTsFirst ... /> |
1942        <sp:LaxTsLast ... /> |
1943        ...
1944      </wsp:Policy>
1945      ...
1946    </sp:Layout>
```

1947

1948 The following describes the attributes and elements listed in the schema outlined above:

1949 /sp:Layout

1950   This identifies a Layout assertion.

1951 /sp:Layout/wsp:Policy

1952   This REQUIRED element contains one or more policy assertions that indicate the specific security
1953   header layout to use.

1954 /sp:Layout/wsp:Policy/sp:Strict

1955   This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1956   property is set to 'Strict'.

1957 /sp:Layout/wsp:Policy/sp:Lax

1958   This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1959   property is set to 'Lax'.

1960 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1961   This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1962   property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1963   'true' by the presence of an sp:IncludeTimestamp assertion.

1964 /sp:Layout/wsp:Policy/sp:LaxTsLast

1965   This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1966   property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
1967   'true' by the presence of an sp:IncludeTimestamp assertion.

1968 7.3 TransportBinding Assertion

1969 The TransportBinding assertion is used in scenarios in which message protection and security correlation
1970 is provided by means other than WSS: SOAP Message Security, for example by a secure transport like
1971 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
1972 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
1973 MUST apply to [Endpoint Policy Subject].

1974 **Syntax**

```
1975      <sp:TransportBinding xmlns:sp="..." ... >
1976        <wsp:Policy xmlns:wsp="...">
1977          <sp:TransportToken ... >
1978            <wsp:Policy> ... </wsp:Policy>
1979            ...
1980          </sp:TransportToken>
1981          <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1982          <sp:Layout ... > ... </sp:Layout> ?
1983          <sp:IncludeTimestamp ... /> ?
1984          ...
1985        </wsp:Policy>
1986        ...
1987      </sp:TransportBinding>
```

1988

1989 The following describes the attributes and elements listed in the schema outlined above:

1990 /sp:TransportBinding

1991   This identifies a TransportBinding assertion.

1992 /sp:TransportBinding/wsp:Policy

1993   This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
1994   assertion.

1995 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1996   This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.
1997   The specified token populates the [Transport Token] property and indicates how the transport is
1998   secured.

1999   /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

2000   This indicates a nested policy that identifies the type of Transport Token to use.

2001   /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

2002   This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2003   Suite] property. See Section 6.1 for more details.

2004   /sp:TransportBinding/wsp:Policy/sp:Layout

2005   This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2006   Header Layout] property. See Section 6.7 for more details.

2007   /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

2008   This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2009   to 'true'.

2010 7.4 SymmetricBinding Assertion

2011   The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
2012   defined in WSS: SOAP Message Security. This binding has two binding specific token properties;
2013   [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
2014   binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
2015   initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
2016   initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
2017   properties and is used as the basis for both encryption and signature in both directions. This assertion
2018   SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

2019   **Syntax**

```
2020    <sp:SymmetricBinding xmlns:sp="..." ... >
2021      <wsp:Policy xmlns:wsp="...">
2022        (
2023          <sp:EncryptionToken ... >
2024            <wsp:Policy> ... </wsp:Policy>
2025          </sp:EncryptionToken>
2026          <sp:SignatureToken ... >
2027            <wsp:Policy> ... </wsp:Policy>
2028          </sp:SignatureToken>
2029        ) | (
2030          <sp:ProtectionToken ... >
2031            <wsp:Policy> ... </wsp:Policy>
2032          </sp:ProtectionToken>
2033        )
2034        <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2035        <sp:Layout ... > ... </sp:Layout> ?
2036        <sp:IncludeTimestamp ... /> ?
2037        <sp:EncryptBeforeSigning ... /> ?
2038        <sp:EncryptSignature ... /> ?
2039        <sp:ProtectTokens ... /> ?
2040        <sp:OnlySignEntireHeadersAndBody ... /> ?
2041        ...
2042      </wsp:Policy>
2043      ...
2044    </sp:SymmetricBinding>
```

2045

2046   The following describes the attributes and elements listed in the schema outlined above:

2047   /sp:SymmetricBinding

2048   This identifies a SymmetricBinding assertion.

2049   /sp:SymmetricBinding/wsp:Policy

2050   This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
2051   assertion.

2052   /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken

2053   This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption
2054   Token. The specified token populates the [Encryption Token] property and is used for encryption.
2055   It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

2056   /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy

2057   The policy contained here MUST identify exactly one token to use for encryption.

2058   /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken

2059   This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.
2060   The specified token populates the [Signature Token] property and is used for the message
2061   signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
2062   specified.

2063   /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy

2064   The policy contained here MUST identify exactly one token to use for signatures.

2065   /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken

2066   This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.
2067   The specified token populates the [Encryption Token] and [Signature Token properties] and is
2068   used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
2069   assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
2070   specified.

2071   /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy

2072   The policy contained here MUST identify exactly one token to use for protection.

2073   /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite

2074   This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2075   Suite] property. See Section 6.1 for more details.

2076   /sp:SymmetricBinding/wsp:Policy/sp:Layout

2077   This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2078   Header Layout] property. See Section 6.7 for more details.

2079   /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp

2080   This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2081   to 'true'.

2082   /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

2083   This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
2084   set to 'EncryptBeforeSigning'.

2085   /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature

2086   This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
2087   property is set to 'true'.

2088   /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens

2089   This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
2090   set to 'true'.

2091   /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

2092   This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
2093   Signatures] property is set to 'true'.

7.5 AsymmetricBinding Assertion

2095   The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means
2096   defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly
2097   used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and
2098   signature. However it is also common practice to use distinct keys for encryption and signature, because
2099   of their different lifecycles.

2100

2101   This binding enables either of these practices by means of four binding specific token properties: [Initiator
2102   Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption
2103   Token].

2104

2105   If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator
2106   Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient
2107   Encryption Token] will both refer to the same token.

2108

2109   If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator
2110   Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient
2111   Encryption Token] will refer to different tokens.

2112

2113   If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the
2114   message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response
2115   message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the
2116   response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for
2117   the message encryption from initiator to the recipient. Note that in each case, the token is associated with
2118   the party (initiator or recipient) who knows the secret.

2119   This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy
2120   Subject].

2121   **Syntax**

```
2122   <sp:AsymmetricBinding xmlns:sp="..." ... >
2123     <wsp:Policy xmlns:wsp="...">
2124       (
2125        <sp:InitiatorToken>
2126         <wsp:Policy> ... </wsp:Policy>
2127        </sp:InitiatorToken>
2128       ) | (
2129        <sp:InitiatorSignatureToken>
2130          <wsp:Policy> ... </wsp:Policy>
2131        </sp:InitiatorSignatureToken>
2132        <sp:InitiatorEncryptionToken>
2133          <wsp:Policy> ... </wsp:Policy>
2134        </sp:InitiatorEncryptionToken>
2135       )
2136       (
2137        <sp:RecipientToken>
2138          <wsp:Policy> ... </wsp:Policy>
2139        </sp:RecipientToken>
2140       ) | (
```

```
2141          <sp:RecipientSignatureToken>
2142            <wsp:Policy> ... </wsp:Policy>
2143          </sp:RecipientSignatureToken>
2144          <sp:RecipientEncryptionToken>
2145            <wsp:Policy> ... </wsp:Policy>
2146          </sp:RecipientEncryptionToken>
2147        )
2148        <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2149        <sp:Layout ... > ... </sp:Layout> ?
2150        <sp:IncludeTimestamp ... /> ?
2151        <sp:EncryptBeforeSigning ... /> ?
2152        <sp:EncryptSignature ... /> ?
2153        <sp:ProtectTokens ... /> ?
2154        <sp:OnlySignEntireHeadersAndBody ... /> ?
2155        ...
2156      </wsp:Policy>
2157      ...
2158    </sp:AsymmetricBinding>
```

2160   The following describes the attributes and elements listed in the schema outlined above:

2161   /sp:AsymmetricBinding

2162   This identifies a AsymmetricBinding assertion.

2163   /sp:AsymmetricBinding/wsp:Policy

2164   This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
2165   assertion.

2166   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2167   This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.
2168   The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
2169   properties and is used for the message signature from initiator to recipient, and encryption from
2170   recipient to initiator.

2171   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2172   The policy contained here MUST identify one or more token assertions.

2173   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2174   This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2175   Signature Token. The specified token populates the [Initiator Signature Token] property and is
2176   used for the message signature from initiator to recipient.

2177   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2178   The policy contained here MUST identify one or more token assertions.

2179   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2180   This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2181   Encryption Token. The specified token populates the [Initiator Encryption Token] property and is
2182   used for the message encryption from recipient to initiator.

2183   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2184   The policy contained here MUST identify one or more token assertions.

2185   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2186   This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.
2187   The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
2188   property and is used for encryption from initiator to recipient, and for the message signature from
2189   recipient to initiator.

2190 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2191   The policy contained here MUST identify one or more token assertions.

2192 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2193   This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
2194   Signature Token. The specified token populates the [Recipient Signature Token] property and is
2195   used for the message signature from recipient to initiator.

2196 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy

2197   The policy contained here MUST identify one or more token assertions.

2198 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken

2199   This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
2200   Encryption Token. The specified token populates the [Recipient Encryption Token] property and
2201   is used for the message encryption from initiator to recipient.

2202 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy

2203   The policy contained here MUST identify one or more token assertions.

2204 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite

2205   This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2206   Suite] property. See Section 6.1 for more details.

2207 /sp:AsymmetricBinding/wsp:Policy/sp:Layout

2208   This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2209   Header Layout] property. See Section 6.7 for more details.

2210 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp

2211   This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2212   to 'true'.

2213 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

2214   This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
2215   set to 'EncryptBeforeSigning'.

2216 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature

2217   This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
2218   property is set to 'true'.

2219 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens

2220   This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
2221   set to 'true'.

2222 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

2223   This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
2224   Signatures] property is set to 'true'.

8 Supporting Tokens

Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the "message signature". In case of Transport Binding the message is signed outside of the message XML by the underlying transport protocol and the signature itself is not part of the message. Additional tokens MAY be specified to augment the claims provided by the token associated with the "message signature" provided by the Security Binding. This section defines seven properties related to supporting token requirements which MAY be referenced by a Security Binding: [Supporting Tokens], [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens], [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties: SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens, SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens, EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy Subject] or [Operation Policy Subject].

Supporting tokens MAY be specified at a different scope than the binding assertion which provides support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while the supporting tokens might be defined at the scope of a message. When assertions that populate this property are defined in overlapping scopes, the sender SHOULD merge the requirements by including all tokens from the outer scope and any additional tokens for a specific message from the inner scope.

In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine which order signature and encryptions occurred in.

Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional constraints defined by the supporting token assertion. For example, if the supporting token assertion specifies message parts that need to be encrypted, the specified tokens need to be capable of encryption.

To illustrate the different ways that supporting tokens MAY be bound to the message, let's consider a message with three components: Header1, Header2, and Body.

2262 Even before any supporting tokens are added, each binding requires that the message is signed using a
2263 token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important
2264 parts of the message including the message timestamp (TS) facilitate replay detection. The signature is
2265 then included as part of the Security header as illustrated below:

2266

2267

2268 Note: if REQUIRED, the initiator may also include in the Security header the token used as the basis for
2269 the message signature (Sig1), not shown in the diagram.

2270 If transport security is used, only the message timestamp (TS) is included in the Security header as
2271 illustrated below. The "message signature" is provided by the underlying transport protocol and is not part
2272 of the message XML.

2273

8.1 SupportingTokens Assertion

2275 Supporting tokens are included in the security header and MAY OPTIONALLY include additional
2276 message parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and
2277 do not require any protection (signature or encryption) to be applied to the message before they are
2278 added. More specifically there is no requirement on "message signature" being present before the
2279 supporting tokens are added. However it is RECOMMENDED to employ underlying protection
2280 mechanism to ensure that the supporting tokens are cryptographically bound to the message during the
2281 transmission.

2282 **Syntax**

```
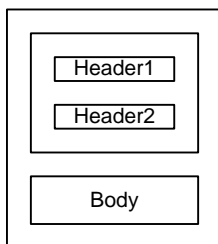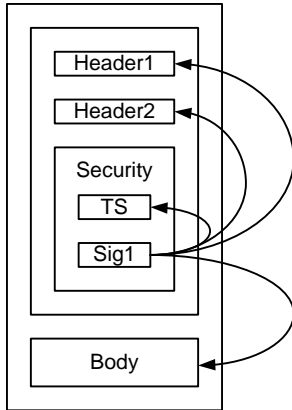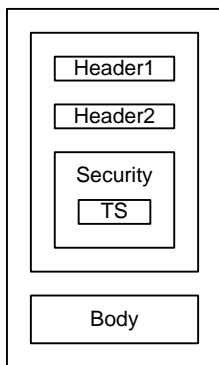2283    <sp:SupportingTokens xmlns:sp="..." ... >
2284      <wsp:Policy xmlns:wsp="...">
2285        [Token Assertion]+
2286        <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2287        (
2288           <sp:SignedParts ... > ... </sp:SignedParts> |
```

```
2289          <sp:SignedElements ... > ... </sp:SignedElements> |
2290          <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2291          <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2292        ) *
2293        ...
2294      </wsp:Policy>
2295      ...
2296    </sp:SupportingTokens>
```

2297

2298 The following describes the attributes and elements listed in the schema outlined above:

2299 /sp:SupportingTokens

2300   This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2301   Tokens] property.

2302 /sp:SupportingTokens/wsp:Policy

2303   This describes additional requirements for satisfying the SupportingTokens assertion.

2304 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2305   The policy MUST identify one or more token assertions.

2306 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2307   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2308   describes the algorithms to use for cryptographic operations performed with the tokens identified
2309   by this policy assertion.

2310 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2311   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2312   and describes additional message parts that MUST be included in the signature generated with
2313   the token identified by this policy assertion.

2314 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2315   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2316   and describes additional message elements that MUST be included in the signature generated
2317   with the token identified by this policy assertion.

2318 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2319   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2320   and describes additional message parts that MUST be encrypted using the token identified by
2321   this policy assertion.

2322 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2323   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2324   and describes additional message elements that MUST be encrypted using the token identified
2325   by this policy assertion.

2326 8.2 SignedSupportingTokens Assertion

2327 Signed tokens are included in the "message signature" as defined above and MAY OPTIONALLY include
2328 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
2329 (Tok2) is signed by the message signature (Sig1):

2330

2331

2332 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2333

2334

2335 **Syntax**

```
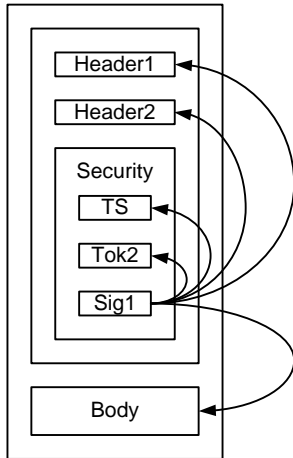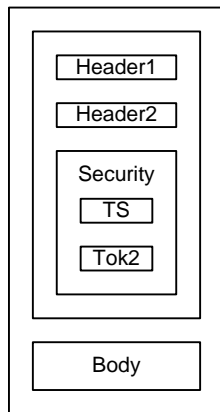2336    <sp:SignedSupportingTokens xmlns:sp="..." ... >
2337      <wsp:Policy xmlns:wsp="...">
2338        [Token Assertion]+
2339        <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2340        (
2341          <sp:SignedParts ... > ... </sp:SignedParts> |
2342          <sp:SignedElements ... > ... </sp:SignedElements> |
2343          <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2344          <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2345        ) *
2346        ...
2347      </wsp:Policy>
2348      ...
2349    </sp:SignedSupportingTokens>
```

2350

2351 The following describes the attributes and elements listed in the schema outlined above:

2352 /sp:SignedSupportingTokens

2353   This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed
2354   Supporting Tokens] property.

2355 /sp:SignedSupportingTokens/wsp:Policy

2356   This describes additional requirements for satisfying the SignedSupportingTokens assertion.

2357　/sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]

2358　　　　The policy MUST identify one or more token assertions.

2359　/sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2360　　　　This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2361　　　　describes the algorithms to use for cryptographic operations performed with the tokens identified
2362　　　　by this policy assertion.

2363　/sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts

2364　　　　This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2365　　　　and describes additional message parts that MUST be included in the signature generated with
2366　　　　the token identified by this policy assertion.

2367　/sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements

2368　　　　This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2369　　　　and describes additional message elements that MUST be included in the signature generated
2370　　　　with the token identified by this policy assertion.

2371　/sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts

2372　　　　This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2373　　　　and describes additional message parts that MUST be encrypted using the token identified by
2374　　　　this policy assertion.

2375　/sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements

2376　　　　This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2377　　　　and describes additional message elements that MUST be encrypted using the token identified
2378　　　　by this policy assertion.

8.3 EndorsingSupportingTokens Assertion

2380　Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
2381　produced from the message signature and MAY OPTIONALLY include additional message parts to sign
2382　and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message
2383　signature (Sig1):

2384

Header1
Header2
Security
TS
Sig1
Sig2
Body

2385

2386　If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
2387　below:

2388

2389

Syntax

```
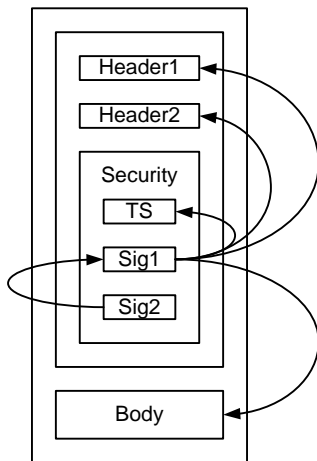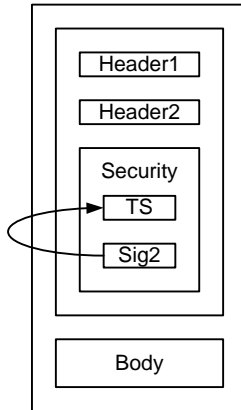<sp:EndorsingSupportingTokens xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    [Token Assertion]+
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
    (
      <sp:SignedParts ... > ... </sp:SignedParts> |
      <sp:SignedElements ... > ... </sp:SignedElements> |
      <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
      <sp:EncryptedElements ... > ... </sp:EncryptedElements>
    ) *
    ...
  </wsp:Policy>
  ...
</sp:EndorsingSupportingTokens>
```

2405

2406   The following describes the attributes and elements listed in the schema outlined above:

2407   /sp:EndorsingSupportingTokens

2408   This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
2409   [Endorsing Supporting Tokens] property.

2410   /sp:EndorsingSupportingTokens/wsp:Policy

2411   This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2412   /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2413   The policy MUST identify one or more token assertions.

2414   /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2415   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2416   describes the algorithms to use for cryptographic operations performed with the tokens identified
2417   by this policy assertion.

2418   /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2419   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2420   and describes additional message parts that MUST be included in the signature generated with
2421   the token identified by this policy assertion.

2422   /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2423   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2424   and describes additional message elements that MUST be included in the signature generated
2425   with the token identified by this policy assertion.

2426 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2427   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2428   and describes additional message parts that MUST be encrypted using the token identified by
2429   this policy assertion.

2430 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2431   This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2432   and describes additional message elements that MUST be encrypted using the token identified
2433   by this policy assertion.

8.4 SignedEndorsingSupportingTokens Assertion

2435 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
2436 and are themselves signed by that message signature, that is both tokens (the token used for the
2437 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY
2438 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
2439 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
2440 message signature (Sig1):

2441

2442

2443 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
2444 SHOULD cover the message timestamp as illustrated below:

2445

2446

Syntax

```
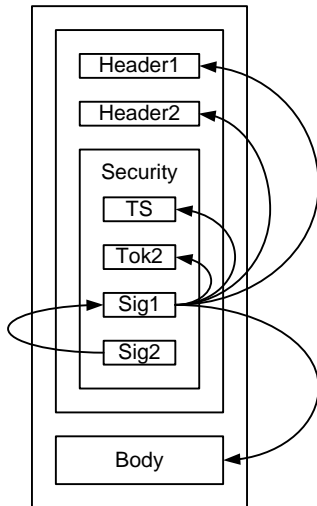<sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    [Token Assertion]+
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
    (
      <sp:SignedParts ... > ... </sp:SignedParts> |
      <sp:SignedElements ... > ... </sp:SignedElements> |
      <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
      <sp:EncryptedElements ... > ... </sp:EncryptedElements>
    ) *
    ...
  </wsp:Policy>
  ...
</sp:SignedEndorsingSupportingTokens>
```

The following describes the attributes and elements listed in the schema outlined above:

/sp:SignedEndorsingSupportingTokens

    This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the [Signed Endorsing Supporting Tokens] property.

/sp:SignedEndorsingSupportingTokens/wsp:Policy

    This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

    The policy MUST identify one or more token assertions.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

    This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and describes the algorithms to use for cryptographic operations performed with the tokens identified by this policy assertion.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

    This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1 and describes additional message parts that MUST be included in the signature generated with the token identified by this policy assertion.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

    This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional message elements that MUST be included in the signature generated with the token identified by this policy assertion.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

    This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

/sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

    This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

8.5 SignedEncryptedSupportingTokens Assertion

Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for encrypting the supporting tokens.

The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the sp:SignedSupportingTokens assertion.

8.6 EncryptedSupportingTokens Assertion

Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in the security header and MUST be encrypted when they appear in the security header. Element encryption SHOULD be used for encrypting these tokens. The encrypted supporting tokens can be added to any SOAP message and do not require the "message signature" being present before the encrypted supporting tokens are added.

The syntax for the sp:EncryptedSupportingTokens differs from the syntax of sp:SupportingTokens only in the name of the assertion itself. All nested policy is as per the sp:SupportingTokens assertion.

The encrypted supporting tokens SHOULD be used only when the sender cannot provide the "message signature" and it is RECOMMENDED that the receiver employs some security mechanisms external to the message to prevent the spoofing attacks. In all other cases it is RECOMMENDED to use signed encrypted supporting tokens instead to ensure that the encrypted tokens are cryptographically bound to the message (See section 8.5).

8.7 EndorsingEncryptedSupportingTokens Assertion

Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for encrypting the supporting tokens.

The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the sp:EndorsingSupportingTokens assertion.

8.8 SignedEndorsingEncryptedSupportingTokens Assertion

Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for encrypting the supporting tokens.

The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the sp:SignedEndorsingSupportingTokens assertion.

8.9 Interaction between [Token Protection] property and supporting token assertions

If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that signature and the following statements hold with respect to the various tokens that sign or are signed;

- The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or the [Signature Token] in the Symmetric binding case, covers that token.
- Endorsing signatures cover the main signature and the endorsing token.

2533   • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the
2534   message signature and once by the endorsing signature.

2535   In addition, signed supporting tokens are covered by the message signature, although this is independent
2536   of [Token Protection].

2537   8.10 Example

2538   Example policy containing supporting token assertions:

```
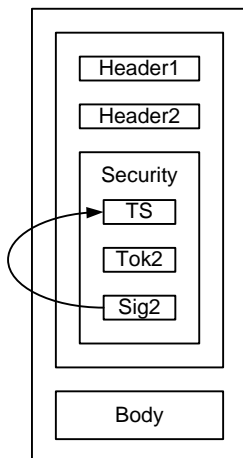2539   <!-- Example Endpoint Policy -->
2540   <wsp:Policy xmlns:wsp="...">
2541     <sp:SymmetricBinding xmlns:sp="...">
2542       <wsp:Policy>
2543         <sp:ProtectionToken>
2544           <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2545             <sp:Issuer>...</sp:Issuer>
2546             <sp:RequestSecurityTokenTemplate>
2547             ...
2548             </sp:RequestSecurityTokenTemplate>
2549           </sp:IssuedToken>
2550         </sp:ProtectionToken>
2551         <sp:AlgorithmSuite>
2552           <wsp:Policy>
2553             <sp:Basic256 />
2554           </wsp:Policy>
2555         </sp:AlgorithmSuite>
2556         ...
2557       </wsp:Policy>
2558     </sp:SymmetricBinding>
2559     ...
2560     <sp:SignedSupportingTokens>
2561       <wsp:Policy>
2562         <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2563       </wsp:Policy>
2564     </sp:SignedSupportingTokens>
2565     <sp:SignedEndorsingSupportingTokens>
2566       <wsp:Policy>
2567         <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2568           <wsp:Policy>
2569             <sp:WssX509v3Token10 />
2570           </wsp:Policy>
2571         </sp:X509Token>
2572       </wsp:Policy>
2573     </sp:SignedEndorsingSupportingTokens>
2574     ...
2575   </wsp:Policy>
```

2576   The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be
2577   included in the security header and covered by the message signature. The
2578   sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the
2579   security header and covered by the message signature. In addition, a signature over the message
2580   signature based on the key material associated with the X509 certificate must be included in the security
2581   header.

9 WSS: SOAP Message Security Options

There are several OPTIONAL aspects to the WSS: SOAP Message Security specification that are independent of the trust and token taxonomies. This section describes another class of properties and associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The assertions defined here MUST apply to [Endpoint Policy Subject].

The properties and assertions dealing with token references defined in this section indicate whether the initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and recipient MAY send a fault if such references are encountered.

Note: This approach is chosen because:

A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used in a single reference.

B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending on which of a series of messages is being secured.

If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a `wsse:InvalidSecurity` fault.

WSS: SOAP Message Security 1.0 Properties

[Direct References]

This property indicates whether the initiator and recipient MUST be able to process direct token references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations MUST be able to process such references.

[Key Identifier References]

This boolean property indicates whether the initiator and recipient MUST be able to process key-specific identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to generate and process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate such references and that the initiator and recipient MAY send a fault if such references are encountered. This property has a default value of 'false'.

[Issuer Serial References]

This boolean property indicates whether the initiator and recipient MUST be able to process references using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate such references and that the initiator and recipient MAY send a fault if such references are encountered. This property has a default value of 'false'.

[External URI References]

This boolean property indicates whether the initiator and recipient MUST be able to process references to tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2624 generate such references and that the initiator and recipient MAY send a fault if such references are
2625 encountered. This property has a default value of 'false'.

2626 **[Embedded Token References]**

2627 This boolean property indicates whether the initiator and recipient MUST be able to process references
2628 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to
2629 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2630 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2631 This property has a default value of 'false'.

2632

2633 **WSS: SOAP Message Security 1.1 Properties**

2634 **[Thumbprint References]**

2635 This boolean property indicates whether the initiator and recipient MUST be able to process references
2636 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to
2637 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2638 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2639 This property has a default value of 'false'.

2640

2641 **[EncryptedKey References]**

2642 This boolean property indicates whether the initiator and recipient MUST be able to process references
2643 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to
2644 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2645 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2646 This property has a default value of 'false'.

2647

2648 **[Signature Confirmation]**

2649 This boolean property specifies whether `wsse11:SignatureConfirmation` elements SHOULD be
2650 used as defined in WSS: Soap Message Security 1.1. If the value is 'true',
2651 `wsse11:SignatureConfirmation` elements MUST be used and signed by the message signature. If
2652 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property
2653 applies to all signatures that are included in the security header. This property has a default value of
2654 'false'. This value of this property does not affect the message parts protected by the message signature
2655 (see the sp:SignedParts and sp:SignedElements assertions)

9.1 Wss10 Assertion

2657 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are
2658 supported.

2659 **Syntax**

```
2660    <sp:Wss10 xmlns:sp="..." ... >
2661      <wsp:Policy xmlns:wsp="...">
2662        <sp:MustSupportRefKeyIdentifier  ... /> ?
2663        <sp:MustSupportRefIssuerSerial   ... /> ?
2664        <sp:MustSupportRefExternalURI  ... /> ?
2665        <sp:MustSupportRefEmbeddedToken  ... /> ?
2666        ...
2667      </wsp:Policy>
2668      ...
2669    </sp:Wss10>
```

2670

2671 The following describes the attributes and elements listed in the schema outlined above:

2672 /sp:Wss10

2673   This identifies a WSS10 assertion.

2674 /sp:Wss10/wsp:Policy

2675   This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2676 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier

2677   This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]
2678   property is set to 'true'.

2679 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial

2680   This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
2681   property is set to 'true'.

2682 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI

2683   This OPTIONAL element is a policy assertion indicates that the [External URI References]
2684   property is set to 'true'.

2685 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken

2686   This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2687   property is set to 'true'.

2688 9.2 Wss11 Assertion

2689 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
2690 supported.

2691 **Syntax**

```
2692    <sp:Wss11 xmlns:sp="..." ... >
2693      <wsp:Policy xmlns:wsp="...">
2694        <sp:MustSupportRefKeyIdentifier  ... /> ?
2695        <sp:MustSupportRefIssuerSerial   ... /> ?
2696        <sp:MustSupportRefExternalURI   ... /> ?
2697        <sp:MustSupportRefEmbeddedToken  ... /> ?
2698        <sp:MustSupportRefThumbprint   ... /> ?
2699        <sp:MustSupportRefEncryptedKey  ... /> ?
2700        <sp:RequireSignatureConfirmation  ... /> ?
2701        ...
2702      </wsp:Policy>
2703    </sp:Wss11>
```

2704

2705 The following describes the attributes and elements listed in the schema outlined above:

2706 /sp:Wss11

2707   This identifies an WSS11 assertion.

2708 /sp:Wss11/wsp:Policy

2709   This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2710 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier

2711   This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]
2712   property is set to 'true'.

2713 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial

2714   This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
2715   property is set to 'true'.

2716   /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI

2717   This OPTIONAL element is a policy assertion indicates that the [External URI References]
2718   property is set to 'true'.

2719   /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken

2720   This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2721   property is set to 'true'.

2722   /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint

2723   This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property
2724   is set to 'true'.

2725   /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey

2726   This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]
2727   property is set to 'true'.

2728   /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation

2729   This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property
2730   is set to 'true'.

10 WS-Trust Options

2732 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically
2733 with client and server challenges and entropy behaviors. These assertions relate to interactions with a
2734 Security Token Service and MAY augment the behaviors defined by the Binding Property Assertions
2735 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2736

2737 **WS-Trust Properties**

2738 **[Client Challenge]**

2739 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a
2740 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of
2741 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of
2742 messages exchanged by the client and service in satisfying the RST. This property has a default value of
2743 'false'.

2744

2745 **[Server Challenge]**

2746 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a
2747 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of
2748 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server MAY
2749 increase the number of messages exchanged by the client and service in order to accommodate the
2750 `wst:SignChallengeResponse` element sent by the client to the server in response to the
2751 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the
2752 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2753

2754 **[Client Entropy]**

2755 This boolean property indicates whether client entropy is REQUIRED to be used as key material for a
2756 requested proof token. A value of 'true' indicates that client entropy is REQUIRED. A value of 'false'
2757 indicates that client entropy is NOT REQUIRED. This property has a default value of 'false'.

2758

2759 **[Server Entropy]**

2760 This boolean property indicates whether server entropy is REQUIRED to be used as key material for a
2761 requested proof token. A value of 'true' indicates that server entropy is REQUIRED. A value of 'false'
2762 indicates that server entropy is NOT REQUIRED. This property has a default value of 'false'.

2763 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy
2764 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm
2765 Suite] property.

2766

2767 **[Issued Tokens]**

2768 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in
2769 WS-Trust. A value of 'true' indicates that the wst:IssuedTokens header is supported. A value of 'false'
2770 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of
2771 'false'.

2772 **[Collection]**

2773 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A
2774 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and
2775 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that
2776 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default
2777 value of 'false'.
2778

2779 **[Scope Policy 1.5]**
2780 This boolean property indicates whether the wsp:AppliesTo element in the [WS-Policy] 1.5 namespace is
2781 supported as described in [WS-Trust]. A value of 'true' indicates that the wsp:AppliesTo element in the
2782 [WS-Policy] 1.5 namespace is supported. A value of 'false' indicates that the wsp:AppliesTo element in
2783 the [WS-Policy] 1.5 namespace is not supported, the [WS-Policy] 1.2 namespace is used instead in this
2784 case. This property has a default value of 'false'.
2785

2786 **[Interactive Challenge]**
2787 This boolean property indicates whether interactive challenges are supported. A value of 'true' indicates
2788 that a wst14:InteractiveChallenge element is supported inside of an RSTR sent by the server to the client.
2789 A value of 'false' indicates that wst14:InteractiveChallenge is not supported. A challenge issued by the
2790 server may increase the number of messages exchanged by the client and service in order to
2791 accommodate the wst14:InteractiveChallengeResponse element sent by the client to the server in
2792 response to the wst14:InteractiveChallenge element. There is an optimization in which a client MAY send
2793 the wst14:InteractiveChallengeResponse element in an initial RST to the server. A final RSTR containing
2794 the issued token will follow subsequent to the server receiving the wst14:InteractiveChallengeResponse
2795 element. This property has a default value of 'false'.
2796

2797 10.1 Trust13 Assertion

2798 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.
2799 **Syntax**

```
2800  <sp:Trust13 xmlns:sp="..." ... >
2801    <wsp:Policy xmlns:wsp="...">
2802      <sp:MustSupportClientChallenge  ... />?
2803      <sp:MustSupportServerChallenge  ... />?
2804      <sp:RequireClientEntropy  ... />?
2805      <sp:RequireServerEntropy  ... />?
2806      <sp:MustSupportIssuedTokens  ... />?
2807      <sp:RequireRequestSecurityTokenCollection />?
2808      <sp:RequireAppliesTo />?
2809      <sp13:ScopePolicy15 />?
2810      <sp13:MustSupportInteractiveChallenge />?
2811      ...
2812    </wsp:Policy>
2813    ...
2814  </sp:Trust13 ... >
```

2815

2816 The following describes the attributes and elements listed in the schema outlined above:
2817 /sp:Trust13
2818   This identifies a Trust13 assertion.
2819 /sp:Trust13/wsp:Policy
2820   This indicates a policy that controls WS-Trust 1.3 options.

2821 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge

2822 This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set
2823 to 'true'.

2824 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge

2825 This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set
2826 to 'true'.

2827 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy

2828 This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to
2829 'true'.

2830 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy

2831 This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to
2832 'true'.

2833 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens

2834 This OPTIONAL element is a policy assertion indicates that the [Issued Tokens] property is set to
2835 'true'.

2836 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2837 This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to
2838 'true'.

2839 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo

2840 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor
2841 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2842 /sp:Trust13/wsp:Policy/sp13:ScopePolicy15

2843 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5]
2844 property is set to 'true'.

2845 /sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge

2846 This optional element is a policy assertion indicates that the [Interactive Challenge]
2847 property is set to 'true'.

11 Guidance on creating new assertions and assertion extensibility

This non-normative appendix provides guidance for designers of new assertions intended for use with this specification.

11.1 General Design Points

- Prefer Distinct Qnames
- Parameterize using nested policy where possible.
- Parameterize using attributes and/or child elements where necessary.

11.2 Detailed Design Guidance

Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per WS-Policy is performed by matching element Qnames. Matching does not take into account attributes that are present on the assertion element. Nor does it take into account child elements except for wsp:Policy elements. If a wsp:Policy element is present, then matching occurs against the assertions nested inside that wsp:Policy element recursively (see Policy Assertion Nesting [WS-Policy]).

When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken into account. In general, multiple assertions with distinct Qnames are preferably to a single assertion that uses attributes and/or content to distinguish different cases. For example, given two possible assertion designs;

```
Design 1

   <A1/>
   <A2/>
   <A3/>

Design 2.

   <A Parameter='1' />
   <A Parameter='2' />
   <A Parameter='3' />
```

then design 1. Would generally be prefered because it allows the policy matching logic to provide more accurate matches between policies.

A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct token assertions, rather than a single sp:Token assertion with, for example, a TokenType attribute. These distinct token assertions make policy matching much more useful as less false positives are generated when performing policy matching.

There are cases where using attributes or child elements as parameters in assertion design is reasonable. Examples include cases when implementations are expected to understand all the values for a given parameter and when encoding the parameter information into the assertion Qname would result in an unmanageable number of assertions. A good example is the sp:IncludeToken attribute that appears

2892   on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2893   attribute and implementations are expected to understand the meaning of all 5 values. If this information
2894   was encoded into the assertion Qnames, each existing token assertion would require five variants, one
2895   for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2896

2897   Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2898   example, the token version assertions defined in Section 5 use such an approach. The overall token type
2899   assertion is parameterized by the nested token version assertions. Policy matching can use these
2900   parameters to find matches between policies where the broad token type is support by both parties but
2901   they might not support the same specific versions.

2902

2903   Note, when designing assertions for new token types such assertions SHOULD allow the
2904   sp:IncludeToken attribute and SHOULD allow nested policy.

2905

12 Security Considerations

2907 It is strongly recommended that policies and assertions be signed to prevent tampering.

2908 It is recommended that policies should not be accepted unless they are signed and have an associated
2909 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely
2910 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that
2911 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2912

2913 It should be noted that the mechanisms described in this document could be secured as part of a SOAP
2914 message using WSS: SOAP Message Security [WSS10, WSS11] or embedded within other objects using
2915 object-specific security mechanisms.

2916

2917 It is recommended that policies not specify two (or more) SignedSupportingTokens or
2918 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are
2919 subject to modification which may be undetectable.

2920

2921 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest
2922 of the policy in order to combat certain XML substitution attacks.

2923

2 February 2009
                                    Page 82 of 114

13 Conformance

An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level requirements defined within this specification. A SOAP Node MUST NOT use the XML namespace identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is compliant with this specification.

This specification references a number of other specifications (see the table above). In order to comply with this specification, an implementation MUST implement the portions of referenced specifications necessary to comply with the required provisions of this specification. Additionally, the implementation of the portions of the referenced specifications that are specifically cited in this specification MUST comply with the rules for those portions as established in the referenced specification.
Additionally normative text within this specification takes precedence over normative outlines (as described in section 1.4.1), which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions. That is, the normative text in this specification further constrains the schemas and/or WSDL that are part of this specification; and this specification contains further constraints on the elements defined in referenced schemas.

This specification defines a number of extensions; compliant services are NOT REQUIRED to implement OPTIONAL features defined in this specification. However, if a service implements an aspect of the specification, it MUST comply with the requirements specified (e.g. related "MUST" statements). If an OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for any other unrecognized/unsupported message. If an OPTIONAL message is supported, then the implementation MUST satisfy all of the MUST and REQUIRED sections of the message.

A. Assertions and WS-PolicyAttachment

This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

TransportBinding Assertion   (Section 7.3)
SymmetricBinding Assertion   (Section 7.4)
AsymmetricBinding Assertion   (Section 7.5)

A.1.2 Token Assertions

SupportingTokens Assertion   (Section 8.1)
SignedSupportingTokens Assertion   (Section 8.2)
EndorsingSupportingTokens Assertion   (Section 8.3)
SignedEndorsingSupportingTokens Assertion   (Section 8.4)
SignedEncryptedSupportingTokens Assertion   (Section 8.5)
EndorsingEncryptedSupportingTokens Assertion   (Section 8.6)
SignedEndorsingEncryptedSupportingTokens Assertion   (Section 8.7)

A.1.3 WSS: SOAP Message Security 1.0 Assertions

Wss10 Assertion   (Section 9.1)

A.1.4 WSS: SOAP Message Security 1.1 Assertions

Wss11 Assertion   (Section 9.2)

A.1.5 Trust 1.0 Assertions

Trust13 Assertion   (Section 10.1)

A.2 Operation Policy Subject Assertions

A.2.1 Security Binding Assertions

SymmetricBinding Assertion   (Section 7.4)
AsymmetricBinding Assertion   (Section 7.5)

A.2.2 Supporting Token Assertions

SupportingTokens Assertion   (Section 8.1)
SignedSupportingTokens Assertion   (Section 8.2)

2984 A.3 Message Policy Subject Assertions

2985 A.3.1 Supporting Token Assertions

2993 A.3.2 Protection Assertions

3001 A.4 Assertions With Undefined Policy Subject

3002 The assertions listed in this section do not have a defined policy subject because they appear nested
3003 inside some other assertion which does have a defined policy subject. This list is derived from nested
3004 assertions in the specification that have independent sections. It is not a complete list of nested
3005 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
3006 additional nested assertions.

3007 A.4.1 General Assertions

3010 A.4.2 Token Usage Assertions

3011 See the nested assertions under the TransportBinding, SymmetricBinding and AssymetricBinding
3012 assertions.

3013 A.4.3 Token Assertions

B. Issued Token Policy

The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

The issued token security model involves a three-party setup. There's a target Server, a Client, and a trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from STS to Client. Policy MAY be embedded inside an Issued Token assertion, or acquired out-of-band. There MAY be an explicit trust relationship between the Server and the STS. There MUST be a trust relationship between the Client and the STS.

The Issued Token policy assertion includes two parts: 1) client-specific parameters that MUST be understood and processed by the client and 2) STS specific parameters which are to be processed by the STS. The format of the Issued Token policy assertion is illustrated in the figure below.

```
+--------------------------------+
| Issued Token Policy            |
|                                |
| :............................: |
| :  Client Parameters        : |
| :............................: |
|                                |
| :............................: |
| :  STS Parameters           : |
| :............................: |
+--------------------------------+
```

The client-specific parameters of the Issued Token policy assertion along with the remainder of the server policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the RST request sent by the Client to the STS as illustrated in the figure below.

Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to formulate the RST request and will include any security-specific requirements of the STS.

The Client MAY augment or replace the contents of the RST made to the STS based on the Client-specific parameters received from the Issued Token policy assertion contained in the Server policy, from policy it received for the STS, or any other local parameters.

3051   The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The
3052   assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along
3053   to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
3054   request sent by the Client to the STS following the protocol defined in WS-Trust.

3055

3056   Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [WS-
3057   Trust]. All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship
3058   which specifies some or all of the conditions and constraints for issued tokens.

C. Strict Security Header Layout Examples

3059

3060 The following sections describe the security header layout for specific bindings when applying the 'Strict'
3061 layout rules defined in Section 6.7.

C.1 Transport Binding

3062

3063 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

C.1.1 Policy

3064

3065 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport
3066 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username
3067 token attached to the message, and finally an X509 token attached to the message and endorsing the
3068 message signature. No message protection requirements are described since the transport covers all
3069 message parts.

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
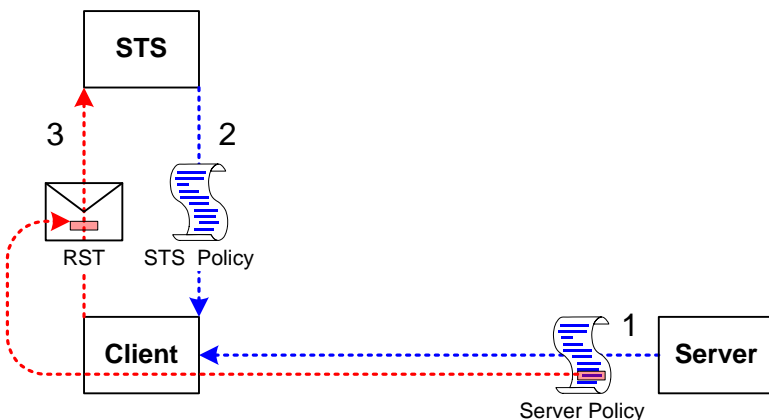        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
    </wsp:Policy>
  </sp:TransportBinding>
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <sp:RequireSignatureConfirmation />
  </sp:Wss11>
</wsp:Policy>
```

3109 This policy is used as the basis for the examples shown in the subsequent section describing the security
3110 header layout for this binding.

3111 C.1.2 Initiator to Recipient Messages

3112 Messages sent from initiator to recipient have the following layout for the security header:

3113   1. A `wsu:Timestamp` element.

3114   2. Any tokens contained in the [Signed Supporting Tokens] property.

3115   3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the
3116   corresponding signature. Each signature MUST cover the `wsu:Timestamp` element from 1
3117   above and SHOULD cover any other unique identifier for the message in order to prevent
3118   replays. If [Token Protection] is 'true', the signature MUST also cover the supporting token. If
3119   [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a
3120   Derived Key Token, based on the supporting token, appears between the supporting token and
3121   the signature.

3122   4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each
3123   signature MUST cover the `wsu:Timestamp` element from 1 above and SHOULD cover at least
3124   some other unique identifier for the message in order to prevent replays. If [Token Protection] is
3125   'true', the signature MUST also cover the supporting token. If [Derived Keys] is 'true' and the
3126   supporting token is associated with a symmetric key, then a Derived Key Token, based on the
3127   supporting token, appears before the signature.

3128 The following diagram illustrates the security header layout for the initiator to recipient message:

3129

3130 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The
3131 arrows on the left from the box labeled Sig$_2$ indicate the parts signed by the supporting token labeled ST$_2$,
3132 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST$_2$.
3133 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering
3134 of the items in the security header follows the most optimal layout for a receiver to process its contents.

3135 *Example:*

3136 Initiator to recipient message

```
3137  <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
3138    <S:Header>
3139      ...
3140      <wsse:Security>
3141        <wsu:Timestamp wsu:Id="timestamp">
3142          <wsu:Created>[datetime]</wsu:Created>
3143          <wsu:Expires>[datetime]</wsu:Expires>
3144        </wsu:Timestamp>
3145        <wsse:UsernameToken wsu:Id='SomeSignedToken' >
3146        ...
3147        </wsse:UsernameToken>
3148        <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
3149        ...
3150        </wsse:BinarySecurityToken>
3151        <ds:Signature>
3152          <ds:SignedInfo>
3153            <ds:References>
3154              <ds:Reference URI="#timestamp" />
3155              <ds:Reference URI="#SomeSignedEndorsingToken" />
3156            </ds:References>
3157          </ds:SignedInfo>
3158          <ds:SignatureValue>...</ds:SignatureValue>
3159          <ds:KeyInfo>
3160            <wsse:SecurityTokenReference>
3161              <wsse:Reference URI="#SomeSignedEndorsingToken" />
3162            </wsse:SecurityTokenReference>
3163          </ds:KeyInfo>
3164        </ds:Signature>
3165        ...
3166      </wsse:Security>
3167      ...
3168    </S:Header>
3169    <S:Body>
3170      ...
3171    </S:Body>
3172  </S:Envelope>
```

C.1.3 Recipient to Initiator Messages

3174 Messages sent from recipient to initiator have the following layout for the security header:

3175 1. A wsu:Timestamp element.

3176 2. If the [Signature Confirmation] property has a value of 'true', then a
3177   wsse11:SignatureConfirmation element for each signature in the corresponding message
3178   sent from initiator to recipient. If there are no signatures in the corresponding message from the
3179   initiator to the recipient, then a wsse11:SignatureConfirmation element with no Value
3180   attribute.

3181 The following diagram illustrates the security header layout for the recipient to initiator message:

3182

3183 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
3184 `wsse11:SignatureConfirmation` element labeled SC$_1$ corresponding to the signature in the initial
3185 message illustrated previously is included. In general, the ordering of the items in the security header
3186 follows the most optimal layout for a receiver to process its contents.

3187 *Example:*

3188 Recipient to initiator message

```
3189  <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3190    <S:Header>
3191      ...
3192      <wsse:Security>
3193        <wsu:Timestamp wsu:Id="timestamp">
3194          <wsu:Created>[datetime]</wsu:Created>
3195          <wsu:Expires>[datetime]</wsu:Expires>
3196        </wsu:Timestamp>
3197        <wsse11:SignatureConfirmation Value="..." />
3198        ...
3199      </wsse:Security>
3200      ...
3201    </S:Header>
3202    <S:Body>
3203      ...
3204    </S:Body>
3205  </S:Envelope>
```

3206 C.2 Symmetric Binding

3207 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

C.2.1 Policy

The following example shows a policy indicating a Symmetric Binding, a symmetric key based IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in the message signature and the supporting signatures, a username token attached to the message, and finally an X509 token attached to the message and endorsing the message signature. Minimum message protection requirements are described as well.

```
<!-- Example Endpoint Policy -->
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
          <sp:Issuer>...</sp:Issuer>
          <sp:RequestSecurityTokenTemplate>
          ...
          </sp:RequestSecurityTokenTemplate>
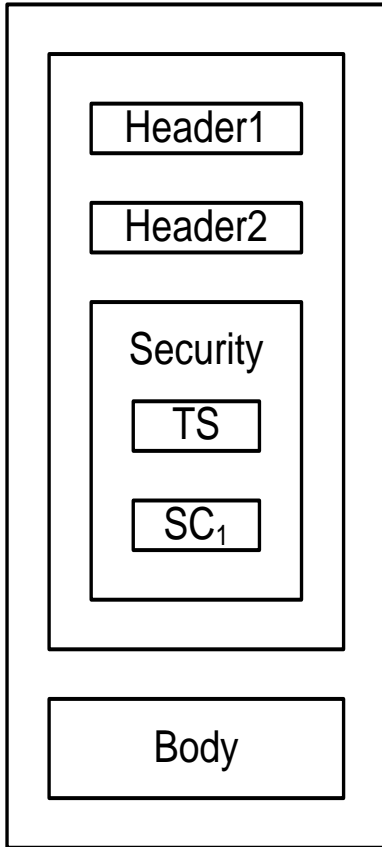        </sp:IssuedToken>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EncryptBeforeSigning />
      <sp:EncryptSignature />
      <sp:ProtectTokens />
      </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <wsp:Policy>
      <sp:RequireSignatureConfirmation />
    </wsp:Policy>
  </sp:Wss11>
</wsp:Policy>
```

```
3264
3265    <!-- Example Message Policy -->
3266    <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3267      <sp:SignedParts>
3268        <sp:Header Name="Header1" Namespace="..." />
3269        <sp:Header Name="Header2" Namespace="..." />
3270        <sp:Body/>
3271      </sp:SignedParts>
3272      <sp:EncryptedParts>
3273        <sp:Header Name="Header2" Namespace="..." />
3274        <sp:Body/>
3275      </sp:EncryptedParts>
3276    </wsp:Policy>
```

3277   This policy is used as the basis for the examples shown in the subsequent section describing the security
3278   header layout for this binding.

C.2.2 Initiator to Recipient Messages

3280   Messages sent from initiator to recipient have the following layout for the security header:

3281   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3282   2. If the `sp:IncludeToken` attribute on the [Encryption Token] is .../IncludeToken/Once or
3283   .../IncludeToken/Always, then the [Encryption Token].

3284   3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3285   Derived Key Token is used for encryption.

3286   4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3287   reference list MUST include a reference to the message signature. If [Protection Order] is
3288   'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3289   specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3290   the token from 3 above MUST be used, otherwise the key in the [Encryption Token].

3291   5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3292   properties whose `sp:IncludeToken` attribute is .../IncludeToken/Once or
3293   .../IncludeToken/Always.

3294   6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3295   attribute on the [Signature Token] is .../IncludeToken/Once or .../IncludeToken/Always, then the
3296   [Signature Token].

3297   7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3298   Derived Key Token is used for signature.

3299   8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3300   whether they are included in the message, and any message parts specified in SignedParts
3301   assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3302   Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3303   the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.

3304   9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3305   Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3306   is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3307   endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3308   endorsing token, appears before the signature.

3309   10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3310   parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3311   in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3312   above.

3313

3314   The following diagram illustrates the security header layout for the initiator to recipient message:

Encrypt Then Sign   Sign Then Encrypt

3315

3316   The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_1.

3317   The dashed arrows on the left from the box labeled Sig_2 indicate the parts signed by the supporting token

3318   labeled ST_2, namely the message signature labeled Sig_1 and the token used as the basis for the

3319   signature labeled ST_2. The arrows on the left from boxes labeled Ref_1 indicate references to parts

3320   encrypted using a key based on the Shared Secret Token labeled ST_1. The dotted arrows inside the box

3321   labeled Security indicate the token that was used as the basis for each cryptographic operation. In

3322   general, the ordering of the items in the security header follows the most optimal layout for a receiver to

3323   process its contents.

3324   *Example:*

3325   Initiator to recipient message using EncryptBeforeSigning:

```
3326    <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3327      xmlns:wsse11="..." xmlns:wsse="..." xmlns:saml="..."
3328      xmlns:xenc="..." xmlns:ds="...">
3329      <S:Header>
3330        <x:Header1 wsu:Id="Header1" >
3331        ...
3332        </x:Header1>
3333
```

```
3334            <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3335              <!-- Plaintext Header2
3336              <x:Header2 wsu:Id="Header2" >
3337              ...
3338              </x:Header2>
3339              -->
3340              ...
3341            </wsse11:EncryptedHeader>
3342            ...
3343            <wsse:Security>
3344              <wsu:Timestamp wsu:Id="Timestamp">
3345                <wsu:Created>...</wsu:Created>
3346                <wsu:Expires>...</wsu:Expires>
3347              </wsu:Timestamp>
3348              <saml:Assertion AssertionId="_SharedSecretToken" ...>
3349              ...
3350              </saml:Assertion>
3351              <xenc:ReferenceList>
3352                <xenc:DataReference URI="#enc_Signature" />
3353                <xenc:DataReference URI="#enc_SomeUsernameToken" />
3354                ...
3355              </xenc:ReferenceList>
3356              <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3357                <!-- Plaintext UsernameToken
3358                <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3359                ...
3360                </wsse:UsernameToken>
3361                -->
3362                ...
3363                <ds:KeyInfo>
3364                  <wsse:SecurityTokenReference>
3365                    <wsse:Reference URI="#_SharedSecretToken" />
3366                  </wsse:SecurityTokenReference>
3367                </ds:KeyInfo>
3368              </xenc:EncryptedData>
3369              <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3370              ...
3371              </wsse:BinarySecurityToken>
3372              <xenc:EncryptedData ID="enc_Signature">
3373                <!-- Plaintext Signature
3374                <ds:Signature Id="Signature">
3375                  <ds:SignedInfo>
3376                    <ds:References>
3377                      <ds:Reference URI="#Timestamp" >...</ds:Reference>
3378                      <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3379                      <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3380                      <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3381                      <ds:Reference URI="#Header1" >...</ds:Reference>
3382                      <ds:Reference URI="#Header2" >...</ds:Reference>
3383                      <ds:Reference URI="#Body" >...</ds:Reference>
3384                    </ds:References>
3385                  </ds:SignedInfo>
3386                  <ds:SignatureValue>...</ds:SignatureValue>
3387                  <ds:KeyInfo>
3388                    <wsse:SecurityTokenReference>
3389                      <wsse:Reference URI="#_SharedSecretToken" />
3390                    </wsse:SecurityTokenReference>
3391                  </ds:KeyInfo>
3392                </ds:Signature>
3393                -->
3394                ...
3395                <ds:KeyInfo>
3396                  <wsse:SecurityTokenReference>
3397                    <wsse:Reference URI="#_SharedSecretToken" />
```

```
3398              </wsse:SecurityTokenReference>
3399            </ds:KeyInfo>
3400          </xenc:EncryptedData>
3401          <ds:Signature>
3402            <ds:SignedInfo>
3403              <ds:References>
3404                <ds:Reference URI="#Signature" >...</ds:Reference>
3405                <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3406              </ds:References>
3407            </ds:SignedInfo>
3408            <ds:SignatureValue>...</ds:SignatureValue>
3409            <ds:KeyInfo>
3410              <wsse:SecurityTokenReference>
3411                <wsse:Reference URI="#SomeSupportingToken" />
3412              </wsse:SecurityTokenReference>
3413            </ds:KeyInfo>
3414          </ds:Signature>
3415          <xenc:ReferenceList>
3416            <xenc:DataReference URI="#enc_Body" />
3417            <xenc:DataReference URI="#enc_Header2" />
3418            ...
3419          </xenc:ReferenceList>
3420        </wsse:Security>
3421      </S:Header>
3422      <S:Body wsu:Id="Body">
3423        <xenc:EncryptedData Id="enc_Body">
3424          ...
3425          <ds:KeyInfo>
3426            <wsse:SecurityTokenReference>
3427              <wsse:Reference URI="#_SharedSecretToken" />
3428            </wsse:SecurityTokenReference>
3429          </ds:KeyInfo>
3430        </xenc:EncryptedData>
3431      </S:Body>
3432    </S:Envelope>
```

3433 C.2.3 Recipient to Initiator Messages

3434 Messages send from recipient to initiator have the following layout for the security header:

3435   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3436   2. If the `sp:IncludeToken` attribute on the [Encryption Token] is .../IncludeToken/Always, then the
3437   [Encryption Token].

3438   3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3439   Derived Key Token is used for encryption.

3440   4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3441   reference list MUST include a reference to the message signature from 6 below, and the
3442   `wsse11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is
3443   'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3444   specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3445   the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2
3446   above.

3447   5. If [Signature Confirmation] is 'true' then a `wsse11:SignatureConfirmation` element for each
3448   signature in the corresponding message sent from initiator to recipient. If there are no signatures
3449   in the corresponding message from the initiator to the recipient, then a
3450   `wsse11:SignatureConfirmation` element with no Value attribute.

3451   6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3452   attribute on the [Signature Token] is .../IncludeToken/Always, then the [Signature Token].

3453   7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This
3454   Derived Key Token is used for signature.

3455   8. A signature over the wsu:Timestamp from 1 above, any `wsse11:SignatureConfirmation`
3456   elements from 5 above, and all the message parts specified in SignedParts assertions in the
3457   policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]
3458   regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token
3459   from 6 above MUST be used, otherwise the key in the [Signature Token].

3460   9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message
3461   parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3462   in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption
3463   Token].

3464   The following diagram illustrates the security header layout for the recipient to initiator message:

Encrypt Then Sign   Sign Then Encrypt

3465

3466   The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_1.
3467   The arrows on the left from boxes labeled Ref_1 indicate references to parts encrypted using a key based
3468   on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two
3469   `wsse11:SignatureConfirmation` elements labeled SC_1 and SC_2 corresponding to the two signatures
3470   in the initial message illustrated previously is included. In general, the ordering of the items in the security
3471   header follows the most optimal layout for a receiver to process its contents. The rules used to determine
3472   this ordering are described in Appendix C.

3473   *Example:*

3474   Recipient to initiator message using EncryptBeforeSigning:

```
3475    <S:Envelope>
3476      <S:Header>
3477        <x:Header1 wsu:Id="Header1" >
3478        ...
3479        </x:Header1>
3480        <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3481          <!-- Plaintext Header2
3482          <x:Header2 wsu:Id="Header2" >
3483          ...
3484          </x:Header2>
3485          -->
3486          ...
3487        </wsse11:EncryptedHeader>
3488        ...
3489        <wsse:Security>
3490          <wsu:Timestamp wsu:Id="Timestamp">
3491            <wsu:Created>...</wsu:Created>
3492            <wsu:Expires>...</wsu:Expires>
3493          </wsu:Timestamp>
3494          <xenc:ReferenceList>
3495            <xenc:DataReference URI="#enc_Signature" />
3496            <xenc:DataReference URI="#enc_SigConf1" />
3497            <xenc:DataReference URI="#enc_SigConf2" />
3498            ...
3499          </xenc:ReferenceList>
3500          <xenc:EncryptedData ID="enc_SigConf1" >
3501            <!-- Plaintext SignatureConfirmation
3502            <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3503            ...
3504            </wsse11:SignatureConfirmation>
3505            -->
3506          ...
3507          </xenc:EncryptedData>
3508          <xenc:EncryptedData ID="enc_SigConf2" >
3509            <!-- Plaintext SignatureConfirmation
3510            <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3511            ...
3512            </wsse11:SignatureConfirmation>
3513            -->
3514          ...
3515          </xenc:EncryptedData>
```

```
3516
3517            <xenc:EncryptedData Id="enc_Signature">
3518              <!-- Plaintext Signature
3519              <ds:Signature Id="Signature">
3520                <ds:SignedInfo>
3521                  <ds:References>
3522                    <ds:Reference URI="#Timestamp" >...</ds:Reference>
3523                    <ds:Reference URI="#SigConf1" >...</ds:Reference>
3524                    <ds:Reference URI="#SigConf2" >...</ds:Reference>
3525                    <ds:Reference URI="#Header1" >...</ds:Reference>
3526                    <ds:Reference URI="#Header2" >...</ds:Reference>
3527                    <ds:Reference URI="#Body" >...</ds:Reference>
3528                  </ds:References>
3529                </ds:SignedInfo>
3530                <ds:SignatureValue>...</ds:SignatureValue>
3531                <ds:KeyInfo>
3532                  <wsse:SecurityTokenReference>
3533                    <wsse:Reference URI="#_SomeIssuedToken" />
3534                  </wsse:SecurityTokenReference>
3535                </ds:KeyInfo>
3536              </ds:Signature>
3537              -->
3538            </xenc:EncryptedData>
3539            ...
3540            <ds:KeyInfo>
3541              <wsse:SecurityTokenReference>
3542                <wsse:Reference URI="#_SomeIssuedToken" />
3543              </wsse:SecurityTokenReference>
3544            </ds:KeyInfo>
3545          <xenc:EncryptedData>
3546          <xenc:ReferenceList>
3547            <xenc:DataReference URI="#enc_Body" />
3548            <xenc:DataReference URI="#enc_Header2" />
3549            ...
3550          </xenc:ReferenceList>
3551       </xenc:EncryptedData>
3552        </wsse:Security>
3553      </S:Header>
3554      <S:Body wsu:Id="Body">
3555        <xenc:EncryptedData Id="enc_Body">
3556          ...
3557          <ds:KeyInfo>
3558            <wsse:SecurityTokenReference>
3559              <wsse:Reference URI="#_SomeIssuedToken" />
3560            </wsse:SecurityTokenReference>
3561          </ds:KeyInfo>
3562        </xenc:EncryptedData>
3563      </S:Body>
3564    </S:Envelope>
```

C.3 Asymmetric Binding

3565

3566 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

C.3.1 Policy

3567

3568 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator
3569 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the
3570 message parts before signing, a requirement to encrypt the message signature, a requirement to include
3571 tokens in the message signature and the supporting signatures, a requirement to include
3572 wsse11:SignatureConfirmation elements, a username token attached to the message, and finally

3573 an X509 token attached to the message and endorsing the message signature. Minimum message
3574 protection requirements are described as well.

```
3575   <!-- Example Endpoint Policy -->
3576   <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3577     <sp:AsymmetricBinding>
3578       <wsp:Policy>
3579         <sp:RecipientToken>
3580           <wsp:Policy>
3581             <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3582           </wsp:Policy>
3583         </sp:RecipientToken>
3584         <sp:InitiatorToken>
3585           <wsp:Policy>
3586             <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3587           </wsp:Policy>
3588         </sp:InitiatorToken>
3589         <sp:AlgorithmSuite>
3590           <wsp:Policy>
3591             <sp:Basic256 />
3592           </wsp:Policy>
3593         </sp:AlgorithmSuite>
3594         <sp:Layout>
3595           <wsp:Policy>
3596             <sp:Strict />
3597           </wsp:Policy>
3598         </sp:Layout>
3599         <sp:IncludeTimestamp />
3600         <sp:EncryptBeforeSigning />
3601         <sp:EncryptSignature />
3602         <sp:ProtectTokens />
3603       </wsp:Policy>
3604     </sp:AsymmetricBinding>
3605     <sp:SignedEncryptedSupportingTokens>
3606       <wsp:Policy>
3607         <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3608       </wsp:Policy>
3609     </sp:SignedEncryptedSupportingTokens>
3610     <sp:SignedEndorsingSupportingTokens>
3611       <wsp:Policy>
3612         <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3613           <wsp:Policy>
3614             <sp:WssX509v3Token10 />
3615           </wsp:Policy>
3616         </sp:X509Token>
3617       </wsp:Policy>
3618     </sp:SignedEndorsingSupportingTokens>
3619     <sp:Wss11>
3620       <wsp:Policy>
3621         <sp:RequireSignatureConfirmation />
3622       </wsp:Policy>
3623     </sp:Wss11>
3624   </wsp:Policy>
3625
```

3626

```
3627   <!-- Example Message Policy -->
3628   <wsp:All xmlns:wsp="..." xmlns:sp="...">
3629     <sp:SignedParts>
3630       <sp:Header Name="Header1" Namespace="..." />
3631       <sp:Header Name="Header2" Namespace="..." />
3632       <sp:Body/>
3633     </sp:SignedParts>
3634     <sp:EncryptedParts>
3635       <sp:Header Name="Header2" Namespace="..." />
3636       <sp:Body/>
3637     </sp:EncryptedParts>
3638   </wsp:All>
```

3639

3640   This policy is used as the basis for the examples shown in the subsequent section describing the security
3641   header layout for this binding.

C.3.2 Initiator to Recipient Messages

3643   Messages sent from initiator to recipient have the following layout:

3644   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3645   2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3646   .../IncludeToken/Once or .../IncludeToken/Always, then the [Recipient Token].

3647   3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3648   [SignatureProtection] is 'true' then an xenc:EncryptedKey element, containing a key encrypted for
3649   the recipient. The xenc:EncryptedKey element MUST include an xenc:ReferenceList containing a
3650   reference to all the message parts specified in EncryptedParts assertions in the policy. If
3651   [Signature Protection] is 'true' then the reference list MUST contain a reference to the message
3652   signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message
3653   signature.

3654   4. Any tokens from the supporting tokens properties (as defined in section 8) whose
3655   `sp:IncludeToken` attribute is .../IncludeToken/Once or .../IncludeToken/Always.

3656   5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3657   .../IncludeToken/Once or .../IncludeToken/Always, then the [Initiator Token].

3658   6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from
3659   1 above, any tokens from 4 above regardless of whether they are included in the message, and
3660   any message parts specified in SignedParts assertions in the policy. If [Token Protection] is 'true',
3661   the signature MUST also cover the [Initiator Token] regardless of whether it is included in the
3662   message.

3663   7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting
3664   Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a
3665   symmetric key, then a Derived Key Token, based on the supporting token, appears before the
3666   signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token
3667   regardless of whether it is included in the message.

3668   8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
3669   [Signature Protection] is 'false' then an xenc:EncryptedKey element, containing a key encrypted
3670   for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
3671   reference list includes a reference to all the message parts specified in EncryptedParts assertions
3672   in the policy. The encrypted parts MUST reference the key contained in the xenc:EncryptedKey
3673   element from 3 above.

3674

3675 The following diagram illustrates the security header layout for the initiator to recipient messages:

Encrypt Then Sign

Header1
Header2
Security
TS
ST_1
EK_1
ST_4
ST_3
ST_2
Sig_2
Sig_3
Ref_1
Body

Sign Then Encrypt

Header1
Header2
Security
TS
ST_1
EK_1
ST_4
ST_3
ST_2
Sig_2
Sig_3
Body

3676

3677 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_2
3678 using the [Initiator Token] labeled ST_2. The dashed arrows on the left from the box labeled Sig_3 indicate
3679 the parts signed by the supporting token ST_3, namely the message signature Sig_2 and the token used as
3680 the basis for the signature labeled ST_3. The arrows on the left from boxes labeled EK_1 indicate references
3681 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on the left
3682 from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the
3683 encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token used as
3684 the basis for each cryptographic operation. In general, the ordering of the items in the security header
3685 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
3686 ordering are described in Appendix C.

3687

3688 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
3689 key contains an external reference to the token containing the encryption key. The diagram illustrates
3690 how one might attach a security token related to the encrypted key for completeness. One possible use-

3691 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3692 wishes to include the encryption token in the message signature.

3693 Initiator to recipient message *Example*

3694
```
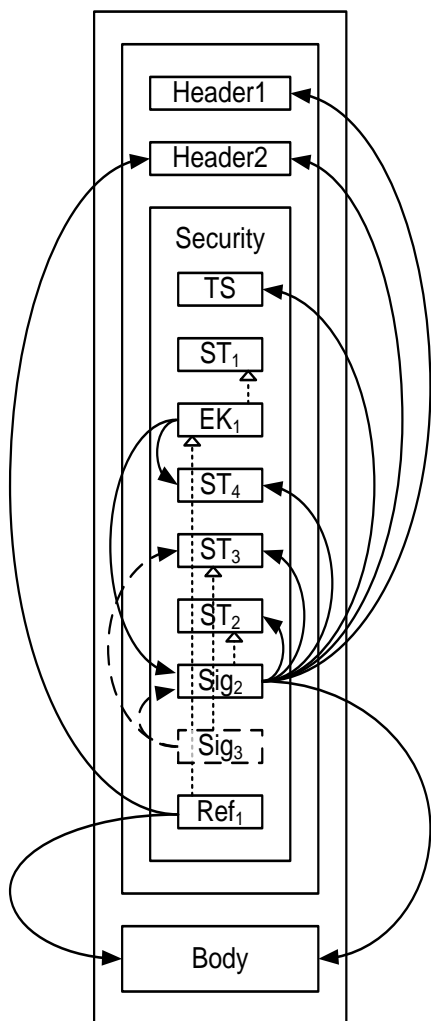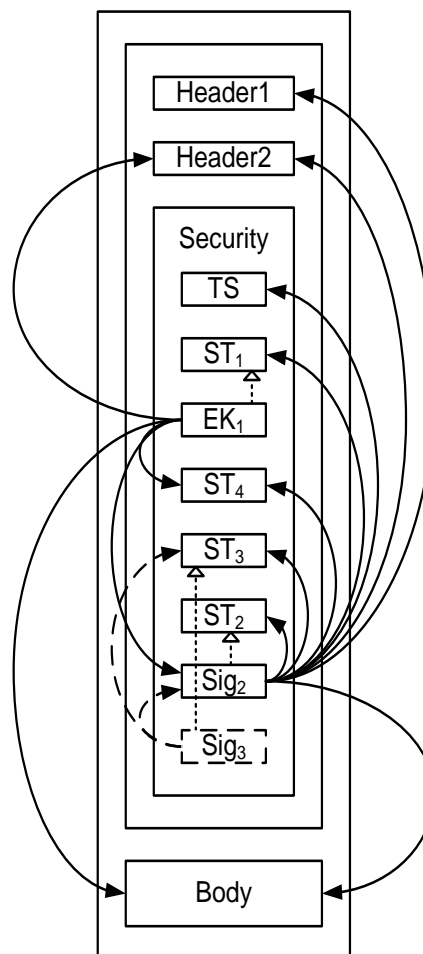<S:Envelope  xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
```

```
3695              xmlns:wsse11="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3696           <S:Header>
3697             <x:Header1 wsu:Id="Header1" >
3698             ...
3699             </x:Header1>
3700             <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3701               <!-- Plaintext Header2
3702               <x:Header2 wsu:Id="Header2" >
3703               ...
3704               </x:Header2>
3705               -->
3706               ...
3707             </wsse11:EncryptedHeader>
3708             ...
3709             <wsse:Security>
3710               <wsu:Timestamp wsu:Id="Timestamp">
3711                 <wsu:Created>...</wsu:Created>
3712                 <wsu:Expires>...</wsu:Expires>
3713               </wsu:Timestamp>
3714               <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3715               ...
3716               </wsse:BinarySecurityToken>
3717               <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3718                 ...
3719                 <xenc:ReferenceList>
3720                   <xenc:DataReference URI="#enc_Signature" />
3721                   <xenc:DataReference URI="#enc_SomeUsernameToken" />
3722                   ...
3723                 </xenc:ReferenceList>
3724               </xenc:EncryptedKey>
3725               <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3726                 <!-- Plaintext UsernameToken
3727                 <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3728                 ...
3729                 </wsse:UsernameToken>
3730                 -->
3731                 ...
3732               </xenc:EncryptedData>
3733               <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3734               ...
3735               </wsse:BinarySecurityToken>
3736               <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3737               ...
3738               </wsse:BinarySecurityToken>
3739               <xenc:EncryptedData ID="enc_Signature">
3740                 <!-- Plaintext Signature
3741                 <ds:Signature Id="Signature">
3742                   <ds:SignedInfo>
3743                     <ds:References>
3744                       <ds:Reference URI="#Timestamp" >...</ds:Reference>
3745                       <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3746                       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3747                       <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3748                       <ds:Reference URI="#Header1" >...</ds:Reference>
3749                       <ds:Reference URI="#Header2" >...</ds:Reference>
3750                       <ds:Reference URI="#Body" >...</ds:Reference>
3751                     </ds:References>
3752                   </ds:SignedInfo>
3753                   <ds:SignatureValue>...</ds:SignatureValue>
3754                   <ds:KeyInfo>
3755                     <wsse:SecurityTokenReference>
3756                       <wsse:Reference URI="#InitiatorToken" />
3757                     </wsse:SecurityTokenReference>
3758                   </ds:KeyInfo>
```

```
3759              </ds:Signature>
3760              -->
3761              ...
3762            </xenc:EncryptedData>
3763            <ds:Signature>
3764              <ds:SignedInfo>
3765                <ds:References>
3766                  <ds:Reference URI="#Signature" >...</ds:Reference>
3767                  <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3768                </ds:References>
3769              </ds:SignedInfo>
3770              <ds:SignatureValue>...</ds:SignatureValue>
3771              <ds:KeyInfo>
3772                <wsse:SecurityTokenReference>
3773                  <wsse:Reference URI="#SomeSupportingToken" />
3774                </wsse:SecurityTokenReference>
3775              </ds:KeyInfo>
3776            </ds:Signature>
3777            <xenc:ReferenceList>
3778              <xenc:DataReference URI="#enc_Body" />
3779              <xenc:DataReference URI="#enc_Header2" />
3780              ...
3781            </xenc:ReferenceList>
3782          </wsse:Security>
3783        </S:Header>
3784        <S:Body wsu:Id="Body">
3785          <xenc:EncryptedData Id="enc_Body">
3786            ...
3787            <ds:KeyInfo>
3788              <wsse:SecurityTokenReference>
3789                <wsse:Reference URI="#RecipientEncryptedKey" />
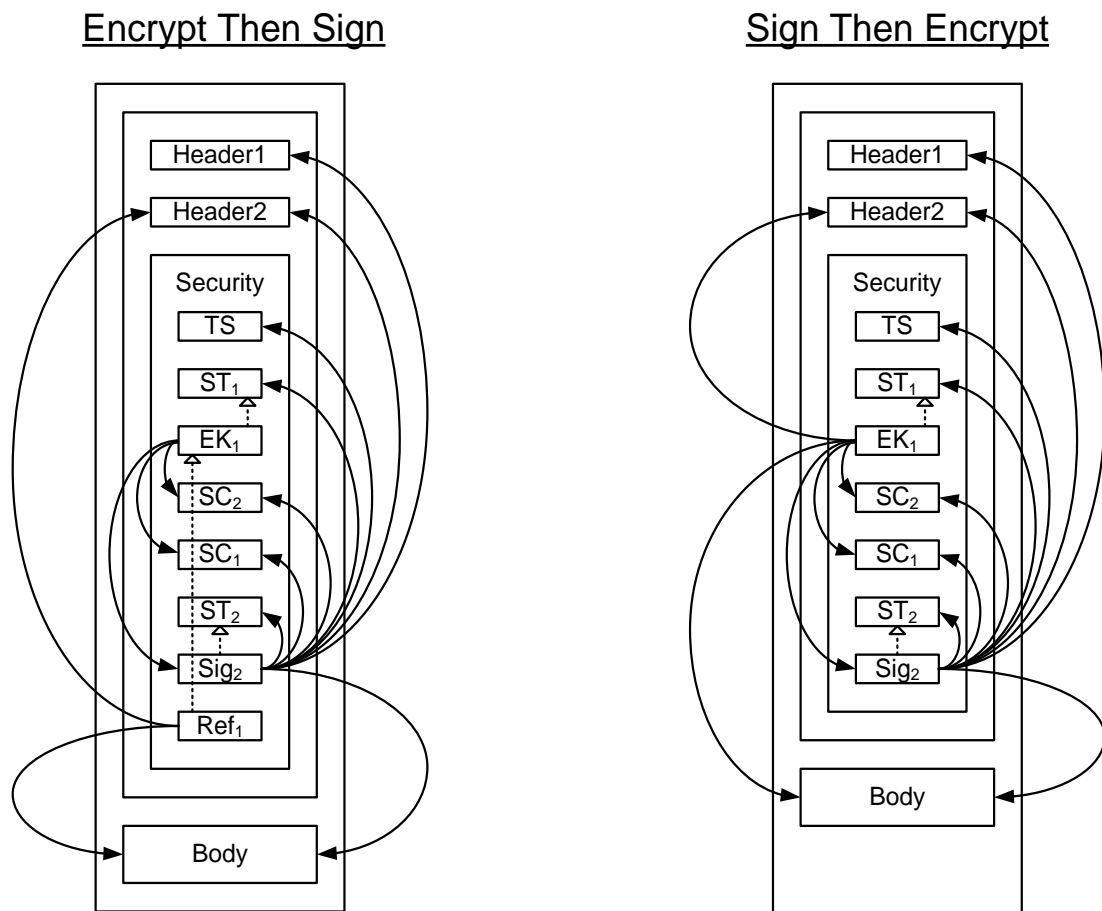3790              </wsse:SecurityTokenReference>
3791            </ds:KeyInfo>
3792          </xenc:EncryptedData>
3793        </S:Body>
3794      </S:Envelope>
```

3795 C.3.3 Recipient to Initiator Messages

3796 Messages sent from recipient to initiator have the following layout:

3797   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3798   2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3799   .../IncludeToken/Always, then the [Initiator Token].

3800   3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3801   [SignatureProtection] is 'true' then an xenc:EncryptedKey element, containing a key encrypted for
3802   the initiator. The xenc:EncryptedKey element MUST include an xenc:ReferenceList containing a
3803   reference to all the message parts specified in EncryptedParts assertions in the policy. If
3804   [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3805   message signature from 6 below, if any and references to the
3806   `wsse11:SignatureConfirmation` elements from 4 below, if any.

3807   4. If [Signature Confirmation] is 'true', then a `wsse11:SignatureConfirmation` element for each
3808   signature in the corresponding message sent from initiator to recipient. If there are no signatures
3809   in the corresponding message from the initiator to the recipient, then a
3810   `wsse11:SignatureConfirmation` element with no Value attribute.

3811   5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3812   .../IncludeToken/Always, then the [Recipient Token].

3813   6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],
3814   over the `wsu:Timestamp` from 1 above, the `wsse11:SignatureConfirmation` elements
3815   from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token
3816   Protection] is 'true' then the signature MUST also cover the [Recipient Token].

3817   7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
3818   [Signature Protection] is 'false' then an xenc:EncryptedKey element, containing a key encrypted
3819   for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
3820   reference list includes a reference to all the message parts specified in EncryptedParts assertions
3821   in the policy. The encrypted parts MUST reference the key contained in the xenc:EncryptedKey
3822   element from 3 above.

3823

3824 The following diagram illustrates the security header layout for the recipient to initiator messages:

Encrypt Then Sign   ## Sign Then Encrypt

3825

3826 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_2
3827 using the [Recipient Token] labeled ST_2. The arrows on the left from boxes labeled EK_1 indicate
3828 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on
3829 the left from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained
3830 in the encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token
3831 used as the basis for each cryptographic operation. Two `wsse11:SignatureConfirmation` elements
3832 labeled SC_1 and SC_2 corresponding to the two signatures in the initial message illustrated previously is
3833 included. In general, the ordering of the items in the security header follows the most optimal layout for a
3834 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

3835 Recipient to initiator message *Example:*

```
<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
  xmlns:wsse11="..." xmlns:wsse="..."
  xmlns:xenc="..." xmlns:ds="...">
  <S:Header>
    <x:Header1 wsu:Id="Header1" >
    ...
    </x:Header1>
    <wsse11:EncryptedHeader wsu:Id="enc_Header2">
      <!-- Plaintext Header2
      <x:Header2 wsu:Id="Header2" >
      ...
      </x:Header2>
      -->
      ...
    </wsse11:EncryptedHeader>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp">
        <wsu:Created>...</wsu:Created>
        <wsu:Expires>...</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
      ...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
        ...
        <xenc:ReferenceList>
          <xenc:DataReference URI="#enc_Signature" />
          <xenc:DataReference URI="#enc_SigConf1" />
          <xenc:DataReference URI="#enc_SigConf2" />
          ...
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <xenc:EncryptedData ID="enc_SigConf2" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf2" ...>
        ...
        </wsse11:SignatureConfirmation>
        -->
        ...
      </xenc:EncryptedData>
      <xenc:EncryptedData ID="enc_SigConf1" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf1" ...>
        ...
        </wsse11:SignatureConfirmation>
        -->
        ...
      </xenc:EncryptedData>
      <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
      ...
      </wsse:BinarySecurityToken>
```

```
3889            <xenc:EncryptedData ID="enc_Signature">
3890              <!-- Plaintext Signature
3891              <ds:Signature Id="Signature">
3892                <ds:SignedInfo>
3893                  <ds:References>
3894                    <ds:Reference URI="#Timestamp" >...</ds:Reference>
3895                    <ds:Reference URI="#SigConf1" >...</ds:Reference>
3896                    <ds:Reference URI="#SigConf2" >...</ds:Reference>
3897                    <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3898                    <ds:Reference URI="#Header1" >...</ds:Reference>
3899                    <ds:Reference URI="#Header2" >...</ds:Reference>
3900                    <ds:Reference URI="#Body" >...</ds:Reference>
3901                  </ds:References>
3902                </ds:SignedInfo>
3903                <ds:SignatureValue>...</ds:SignatureValue>
3904                <ds:KeyInfo>
3905                  <wsse:SecurityTokenReference>
3906                    <wsse:Reference URI="#RecipientToken" />
3907                  </wsse:SecurityTokenReference>
3908                </ds:KeyInfo>
3909              </ds:Signature>
3910              -->
3911              ...
3912            </xenc:EncryptedData>
3913            <xenc:ReferenceList>
3914              <xenc:DataReference URI="#enc_Body" />
3915              <xenc:DataReference URI="#enc_Header2" />
3916              ...
3917            </xenc:ReferenceList>
3918          </wsse:Security>
3919        </S:Header>
3920        <S:Body wsu:Id="Body">
3921          <xenc:EncryptedData Id="enc_Body">
3922            ...
3923            <ds:KeyInfo>
3924              <wsse:SecurityTokenReference>
3925                <wsse:Reference URI="#InitiatorEncryptedKey" />
3926              </wsse:SecurityTokenReference>
3927            </ds:KeyInfo>
3928          </xenc:EncryptedData>
3929        </S:Body>
3930      </S:Envelope>
```

D. Signed and Encrypted Elements in the Security Header

This section lists the criteria for when various child elements of the Security header are signed and/or encrypted at the message level including whether they are signed by the message signature or a supporting signature. It assumes that there are no `sp:SignedElements` and no `sp:EncryptedElements` assertions in the policy. If such assertions are present in the policy then additional child elements of the security header might be signed and/or encrypted.

D.1 Elements signed by the message signature

1. The `wsu:Timestamp` element (Section 6.2).
2. All `wsse11:SignatureConfirmation` elements (Section 9).
3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token], [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption Token] when [Token Protection] has a value of 'true' (Section 6.5).
4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed Endorsing Supporting Tokens] (Section 8.5).

D.2 Elements signed by all endorsing signatures

1. The `ds:Signature` element that forms the message signature (Section 8.3).
2. The `wsu:Timestamp` element in the case of a transport binding (Section 8.3).

D.3 Elements signed by a specific endorsing signature

1. Security Tokens corresponding to [Endorsing Supporting Tokens] or [Signed Endorsing Supporting Tokens] when [Token Protection] has a value of 'true' (Section 8.8).

D.4 Elements that are encrypted

1. The `ds:Signature` element that forms the message signature when [Signature Protection] has a value of 'true' (Section 6.4).
2. All `wsse11:SignatureConfirmation` elements when [Signature Protection] has a value of 'true' (Section 6.4).
3. A `wsse:UsernameToken` MAY be encrypted when a transport binding is not being used (Section 5.3.1).

E. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Original Authors of the intial contribution:

Giovanni Della-Libera, Microsoft
Martin Gudgin, Microsoft
Phillip Hallam-Baker, VeriSign
Maryann Hondo, IBM
Hans Granqvist, Verisign
Chris Kaler, Microsoft (editor)
Hiroshi Maruyama, IBM
Michael McIntosh, IBM
Anthony Nadalin, IBM (editor)
Nataraj Nagaratnam, IBM
Rob Philpott, RSA Security
Hemma Prafullchandra, VeriSign
John Shewchuk, Microsoft
Doug Walter, Microsoft
Riaz Zolfonoon, RSA Security

Original Acknowledgements of the initial contribution:

Vaithialingam B. Balayoghan, Microsoft
Francisco Curbera, IBM
Christopher Ferris, IBM
Cédric Fournet, Microsoft
Andy Gordon, Microsoft
Tomasz Janczuk, Microsoft
David Melgar, IBM
Mike Perks, IBM
Bruce Rich, IBM
Jeffrey Schlimmer, Microsoft
Chris Sharp, IBM
Kent Tamura, IBM
T.R. Vishwanath, Microsoft

Elliot Waingold, Microsoft

TC Members during the development of this specification:

Don Adams, Tibco Software Inc.
Jan Alexander, Microsoft Corporation
Steve Anderson, BMC Software
Donal Arundel, IONA Technologies
Howard Bae, Oracle Corporation
Abbie Barbir, Nortel Networks Limited
Charlton Barreto, Adobe Systems
Mighael Botha, Software AG, Inc.
Toufic Boubez, Layer 7 Technologies Inc.
Norman Brickman, Mitre Corporation
Melissa Brumfield, Booz Allen Hamilton

4008	Geoff Bullen, Microsoft Corporation
4009	Lloyd Burch, Novell
4010	Scott Cantor, Internet2
4011	Greg Carpenter, Microsoft Corporation
4012	Steve Carter, Novell
4013	Symon Chang, Oracle Corporation Ching-Yun (C.Y.) Chao, IBM
4014	Martin Chapman, Oracle Corporation
4015	Kate Cherry, Lockheed Martin
4016	Henry (Hyenvui) Chung, IBM
4017	Luc Clement, Systinet Corp.
4018	Paul Cotton, Microsoft Corporation
4019	Glen Daniels, Sonic Software Corp.
4020	Peter Davis, Neustar, Inc.
4021	Duane DeCouteau, Veterans Health Administration
4022	Martijn de Boer, SAP AG
4023	Werner Dittmann, Siemens AG
4024	Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
4025	Fred Dushin, IONA Technologies
4026	Petr Dvorak, Systinet Corp.
4027	Colleen Evans, Microsoft Corporation
4028	Ruchith Fernando, WSO2
4029	Mark Fussell, Microsoft Corporation
4030	Vijay Gajjala, Microsoft Corporation
4031	Marc Goodner, Microsoft Corporation
4032	Hans Granqvist, VeriSign
4033	Martin Gudgin, Microsoft Corporation
4034	Tony Gullotta, SOA Software Inc.
4035	Jiandong Guo, Sun Microsystems
4036	Phillip Hallam-Baker, VeriSign
4037	Patrick Harding, Ping Identity Corporation
4038	Heather Hinton, IBM
4039	Frederick Hirsch, Nokia Corporation
4040	Jeff Hodges, Neustar, Inc.
4041	Will Hopkins, Oracle Corporation
4042	Alex Hristov, Otecia Incorporated
4043	John Hughes, PA Consulting
4044	Diane Jordan, IBM
4045	Venugopal K, Sun Microsystems
4046	Chris Kaler, Microsoft Corporation
4047	Dana Kaufman, Forum Systems, Inc.
4048	Paul Knight, Nortel Networks Limited
4049	Ramanathan Krishnamurthy, IONA Technologies
4050	Christopher Kurt, Microsoft Corporation
4051	Kelvin Lawrence, IBM
4052	Hubert Le Van Gong, Sun Microsystems
4053	Jong Lee, Oracle Corporation
4054	Rich Levinson, Oracle Corporation
4055	Tommy Lindberg, Dajeil Ltd.
4056	Mark Little, JBoss Inc.
4057	Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.
4058	Eve Maler, Sun Microsystems
4059	Ashok Malhotra, Oracle Corporation
4060	Anand Mani, CrimsonLogic Pte Ltd
4061	Jonathan Marsh, Microsoft Corporation
4062	Robin Martherus, Oracle Corporation
4063	Miko Matsumura, Infravio, Inc.
4064	Gary McAfee, IBM

4065   Michael McIntosh, IBM
4066   John Merrells, Sxip Networks SRL
4067   Jeff Mischkinsky, Oracle Corporation
4068   Prateek Mishra, Oracle Corporation
4069   Bob Morgan, Internet2
4070   Vamsi Motukuru, Oracle Corporation
4071   Raajmohan Na, EDS
4072   Anthony Nadalin, IBM
4073   Andrew Nash, Reactivity, Inc.
4074   Eric Newcomer, IONA Technologies
4075   Duane Nickull, Adobe Systems
4076   Toshihiro Nishimura, Fujitsu Limited
4077   Rob Philpott, RSA Security
4078   Denis Pilipchuk, Oracle Corporation.
4079   Darren Platt, Ping Identity Corporation
4080   Martin Raepple, SAP AG
4081   Nick Ragouzis, Enosis Group LLC
4082   Prakash Reddy, CA
4083   Alain Regnier, Ricoh Company, Ltd.
4084   Irving Reid, Hewlett-Packard
4085   Bruce Rich, IBM
4086   Tom Rutt, Fujitsu Limited
4087   Maneesh Sahu, Actional Corporation
4088   Frank Siebenlist, Argonne National Laboratory
4089   Joe Smith, Apani Networks
4090   Davanum Srinivas, WSO2
4091   David Staggs, Veterans Health Administration
4092   Yakov Sverdlov, CA
4093   Gene Thurston, AmberPoint
4094   Victor Valle, IBM
4095   Asir Vedamuthu, Microsoft Corporation
4096   Greg Whitehead, Hewlett-Packard
4097   Ron Williams, IBM
4098   Corinna Witt, Oracle Corporation
4099   Kyle Young, Microsoft Corporation
4100
4101