

WS-SecurityPolicy 1.2

Editors Draft 01, 4 December 2006 with proposal for PR020

Artifact Identifier:

ws-securitypolicy-1.2-spec-ed-12

Location:

Current: docs.oasis-open.org/ws-sx/ws-securitypolicy/200512

This Version: docs.oasis-open.org/ws-sx/ws-securitypolicy/200512

Previous Version: [n/a](#)

Artifact Type:

specification

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

OASIS Conceptual Model topic area:

[Topic Area]

Related work:

N/A

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 Open 2006. 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.

Table of Contents

1	Introduction	6
1.1	Example.....	6
1.2	Namespaces	7
1.3	Schema Files.....	8
1.4	Terminology	8
1.4.1	Notational Conventions	8
1.5	Normative References	9
1.6	Non-Normative References	12
2	Security Policy Model	13
2.1	Security Assertion Model.....	13
2.2	Nested Policy Assertions.....	14
2.3	Security Binding Abstraction.....	14
3	Policy Considerations	15
3.1	Nested Policy	15
3.2	Policy Subjects.....	15
4	Protection Assertions	17
4.1	Integrity Assertions	17
4.1.1	SignedParts Assertion	17
4.1.2	SignedElements Assertion	18
4.2	Confidentiality Assertions	19
4.2.1	EncryptedParts Assertion	19
4.2.2	EncryptedElements Assertion	20
4.2.3	ContentEncryptedElements Assertion.....	20
4.3	Required Elements Assertion	21
4.3.1	RequiredElements Assertion.....	21
4.3.2	RequiredParts Assertion.....	22
5	Token Assertions	23
5.1	Token Inclusion	23
5.1.1	Token Inclusion Values	23
5.1.2	Token Inclusion and Token References	24
5.2	Token Properties	24
5.2.1	[Derived Keys] Property	24
5.2.2	[Explicit Derived Keys] Property	24
5.2.3	[Implicit Derived Keys] Property	24
5.3	Token Assertion Types	24
5.3.1	UsernameToken Assertion	24
5.3.2	IssuedToken Assertion.....	26
5.3.3	X509Token Assertion	27
5.3.4	KerberosToken Assertion	29
5.3.5	SpnegoContextToken Assertion.....	30
5.3.6	SecurityContextToken Assertion	31

5.3.7	SecureConversationToken Assertion.....	32
5.3.8	SamlToken Assertion	34
5.3.9	RelToken Assertion.....	36
5.3.10	HttpsToken Assertion.....	37
6	Security Binding Properties	38
6.1	[Algorithm Suite] Property	38
6.2	[Timestamp] Property	40
6.3	[Protection Order] Property	40
6.4	[Signature Protection] Property	40
6.5	[Token Protection] Property.....	40
6.6	[Entire Header and Body Signatures] Property	40
6.7	[Security Header Layout] Property.....	41
6.7.1	Strict Layout Rules for WSS 1.0	41
7	Security Binding Assertions	43
7.1	AlgorithmSuite Assertion	43
7.2	Layout Assertion	45
7.3	TransportBinding Assertion	46
7.4	SymmetricBinding Assertion.....	47
7.5	AsymmetricBinding Assertion	49
8	Supporting Tokens	52
8.1	SupportingTokens Assertion.....	53
8.2	SignedSupportingTokens Assertion	54
8.3	EndorsingSupportingTokens Assertion	56
8.4	SignedEndorsingSupportingTokens Assertion	58
8.5	SignedEncryptedSupportingTokens Assertion	60
8.6	EndorsingEncryptedSupportingTokens Assertion.....	60
8.7	SignedEndorsingEncryptedSupportingTokens Assertion.....	60
8.8	Interaction between [Token Protection] property and supporting token assertions.....	60
8.9	Example.....	60
9	WSS: SOAP Message Security Options	62
9.1	Wss10 Assertion.....	63
9.2	Wss11 Assertion.....	64
10	WS-Trust Options.....	66
10.1	Trust10 Assertion	67
11	Guidance on creating new assertions and assertion extensibility	68
11.1	General Design Points	68
11.2	Detailed Design Guidance.....	68
12	Security Considerations	70
A.	Assertions and WS-PolicyAttachment.....	71
A.1	Endpoint Policy Subject Assertions	71
A.1.1	Security Binding Assertions.....	71
A.1.2	Token Assertions	71
A.1.3	WSS: SOAP Message Security 1.0 Assertions.....	71

A.1.4 WSS: SOAP Message Security 1.1 Assertions	71
A.1.5 Trust 1.0 Assertions	71
A.2 Operation Policy Subject Assertions	71
A.2.1 Security Binding Assertions	71
A.2.2 Supporting Token Assertions	71
A.3 Message Policy Subject Assertions	72
A.3.1 Supporting Token Assertions	72
A.3.2 Protection Assertions	72
A.4 Assertions With Undefined Policy Subject	72
A.4.1 General Assertions	72
A.4.2 Token Usage Assertions	72
A.4.3 Token Assertions	72
B. Issued Token Policy	74
C. Strict Security Header Layout Examples	76
C.1 Transport Binding	76
C.1.1 Policy	76
C.1.2 Initiator to Recipient Messages	77
C.1.3 Recipient to Initiator Messages	78
C.2 Symmetric Binding	79
C.2.1 Policy	80
C.2.2 Initiator to Recipient Messages	81
C.2.3 Recipient to Initiator Messages	85
C.3 Asymmetric Binding	88
C.3.1 Policy	88
C.3.2 Initiator to Recipient Messages	90
C.3.3 Recipient to Initiator Messages	94
D. Signed and Encrypted Elements in the Security Header	98
D.1 Elements signed by the message signature	98
D.2 Elements signed by all endorsing signatures	98
D.3 Elements signed by a specific endorsing signature	98
D.4 Elements that are encrypted	98
E. Acknowledgements	99
F. Revision History	102

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]. 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
(20)         Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(21)       />
(22)     </sp:SignedParts>
(23)     <sp:EncryptedParts>
(24)       <sp:Body/>
(25)     </sp:EncryptedParts>
```

44 (24) </wsp:Policy>

45

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

58 1.2 Namespaces

59 The XML namespace URI that MUST be used by implementations of this specification is:

60 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512>

61

62 | [Table 2](#) lists XML namespaces that are used in this specification. The choice of any namespace prefix is
63 arbitrary and not semantically significant.

64 | [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-wssecurity-secext-1.1.xsd	[WSS11]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	[WS-Policy], [WS-PolicyAttachment]
xsd	http://www.w3.org/2001/XMLSchema	[XML-Schema1], [XML-Schema2]
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[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/200512	This specification
----	-----------------------------------------------------------------------------------------------------------------------------------	--------------------

65 1.3 Schema Files

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

68 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2.xsd>

69 1.4 Terminology

70 **Policy** - A collection of policy alternatives.

71 **Policy Alternative** - A collection of policy assertions.

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

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

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

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

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

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

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

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

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

85 **Supporting Token** - A token used to provide additional claims.

86 1.4.1 Notational Conventions

87 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
88 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described
89 in.

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

- 91 • The syntax appears as an XML instance, but values in italics indicate data types instead of literal
92 values.
- 93 • Characters are appended to elements and attributes to indicate cardinality:
 - 94 ○ "?" (0 or 1)
 - 95 ○ "*" (0 or more)
 - 96 ○ "+" (1 or more)
- 97 • The character "|" is used to indicate a choice between alternatives.
- 98 • The characters "(" and ")" are used to indicate that contained items are to be treated as a group
99 with respect to cardinality or choice.
- 100 • The characters "[" and "]" are used to call out references and property names.
- 101 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be
102 added at the indicated extension points but MUST NOT contradict the semantics of the parent
103 and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver

104 SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated
105 below.

- 106 • XML namespace prefixes (see [Table 2](#)) are used to indicate the namespace of the element being
107 defined.

108

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

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

117 Extensibility points in the exemplar may not be described in the corresponding text.

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

123

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

129 1.5 Normative References

- 130 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement
131 Levels", RFC 2119, Harvard University, March 1997.
132 <http://www.ietf.org/rfc/rfc2119.txt>
133
- 134 [SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.
135 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
136
- 137 [SOAP12] W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24
138 June 2003.
139 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
140
- 141 [SOAPNorm] W3C Working Group Note, "SOAP Version 1.2 Message
142 Normalization", 8 October 2003.
143 <http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>
144
- 145 [URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers
146 (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe
147 Systems, January 2005.
148 <http://www.ietf.org/rfc/rfc3986.txt>
149

150 [RFC2068] IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January
151 1997
152 <http://www.ietf.org/rfc/rfc2068.txt>
153

154 [RFC2246] IETF Standard, "The TLS Protocol", January 1999.
155 <http://www.ietf.org/rfc/rfc2246.txt>
156

157 [SwA] W3C Note, "SOAP Messages with Attachments", 11 December
158 2000
159
160 [http://www.w3.org/TR/2000/NOTE-SOAP-attachments-](http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211)
161 [20001211](http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211)
162

163 [WS-Addressing] W3C Recommendation, "Web Services Addressing (WS-Addressing)",
164 9 May 2006.
165 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>
166

167 [WS-Policy] W3C Member Submission "Web Services Policy 1.2 - Framework", 25
168 April 2006.
169 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
170

171 [WS-PolicyAttachment] W3C Member Submission "Web Services Policy 1.2 - Attachment", 25
172 April 2006.
173 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)
174 [20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)
175

176 [WS-Trust] OASIS Committee Draft, "WS-Trust 1.3", September 2006
177 <http://docs.oasis-open.org/ws-sx/ws-trust/200512>
178

179 [WS-SecureConversation] OASIS Committee Draft, "WS-SecureConversation 1.3", September
180 2006
181 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512>
182

183 [WSS10] OASIS Standard, "OASIS Web Services Security: SOAP Message
184 Security 1.0 (WS-Security 2004)", March 2004.
185 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
186 [message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
187

188 [WSS11] OASIS Standard, "OASIS Web Services Security: SOAP Message
189 Security 1.1 (WS-Security 2004)", February 2006.
190 [http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
191 [spec-os-SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
192

193 [WSS:UsernameToken1.0] OASIS Standard, "Web Services Security: UsernameToken Profile",
194 March 2004
195 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
196 [token-profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)

197		
198	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
199		
200		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
201		
202		
203	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
204		
205		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
206		
207		
208	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
209		
210		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
211		
212		
213	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
214		
215		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
216		
217		
218	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
219		
220		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
221		
222	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
223		
224		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
225		
226		
227	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
228		
229		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
230		
231	[WSS:RELTTokenProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile 1.1", February 2006
232		
233		http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
234		
235		
236	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1", February 2006
237		
238		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
239		
240		
241	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
242		
243		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

244
245 [XML-Signature] W3C Recommendation, "XML-Signature Syntax and Processing", 12
246 February 2002.
247 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>
248
249 [XPath] W3C Recommendation "XML Path Language (XPath) Version 1.0", 16
250 November 1999.
251 <http://www.w3.org/TR/1999/REC-xpath-19991116>
252
253 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second
254 Edition", 28 October 2004.
255 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
256
257 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second
258 Edition", 28 October 2004.
259 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
260
261

262 1.6 Non-Normative References

263 None.

264

265 2 Security Policy Model

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

270
271 The primary goal of this specification is to define an initial set of patterns or sets of assertions that
272 represent common ways to describe how messages are secured on a communication path. The intent is
273 to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging
274 transport security, but to be specific enough to ensure interoperability based on assertion matching.

275
276 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for
277 selecting policy alternatives and the attachment mechanism for associating policy assertions with web
278 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters
279 or attributes. This enables first-level, QName based assertion matching without security domain-specific
280 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed
281 set of policy alternatives that are shared by the two parties attempting to establish a secure
282 communication path.

283
284 In general, assertions defined in this specification allow additional attributes, based on schemas, to be
285 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not
286 match based on these attributes. Attributes specified on the assertion element that are not defined in this
287 specification or in WS-Policy are to be treated as informational properties.

288 2.1 Security Assertion Model

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

296
297 To indicate the scope of protection, assertions identify message parts that are to be protected in a
298 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

299
300 The general aspects of security includes the relationships between or characteristics of the environment
301 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality
302 protection and which are supporting, the applicable algorithms to use, etc.

303
304 The security binding assertion is a logical grouping which defines how the general aspects are used to
305 protect the indicated parts. For example, that an asymmetric token is used with a digital signature to
306 provide integrity protection, and that parts are encrypted with a symmetric key which is then encrypted

307 using the public key of the recipient. At its simplest form, the security binding restricts what can be placed
308 in the `wsse:Security` header and the associated processing rules.

309

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

314 **2.2 Nested Policy Assertions**

315 Assertions may be used to further qualify a specific aspect of another assertion. For example, an
316 assertion describing the set of algorithms to use may qualify the specific behavior of a security binding.

317 **2.3 Security Binding Abstraction**

318 As previously indicated, individual assertions are designed to be used in multiple combinations. The
319 binding represents common usage patterns for security mechanisms. These Security Binding assertions
320 are used to determine how the security is performed and what to expect in the `wsse:Security` header.

321 Bindings are described textually and enforced programmatically. This specification defines several
322 bindings but others can be defined and agreed to for interoperability if participating parties support it.

323

324 A binding defines the following security characteristics:

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

333

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

340

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

- 344 • TransportBinding (Section 7.3)
- 345 • SymmetricBinding (Section 7.4)
- 346 • AsymmetricBinding (Section 7.5)

347 **3 Policy Considerations**

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

350 **3.1 Nested Policy**

351 This specification makes extensive use of nested policy assertions as described in the [Policy Assertion](#)
352 [Nesting](#) section of WS-Policy.
353

354 **3.2 Policy Subjects**

355 WS-PolicyAttachment defines various attachment points for policy. This section defines properties that
356 are referenced later in this document describing the recommended or required attachment points for
357 various assertions. In addition, [Appendix A](#) groups the various assertions according to policy subject.

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

359 **[Message Policy Subject]**

360 This property identifies a Message Policy Subject [[WS-PolicyAttachment](#)]. WS-PolicyAttachment defines
361 seven WSDL [WSDL 1.1] policy attachment points with Message Policy Subject:
362

363 wsdl:message

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

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

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

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

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

372 **[Operation Policy Subject]**

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

374 wsdl:portType/wsdl:operation

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

377 wsdl:binding/wsdl:operation

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

380

381

382 **[Endpoint Policy Subject]**

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

385 wsdl:portType

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

388 wsdl:binding

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

391 wsdl:port

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

394 4 Protection Assertions

395 The following assertions are used to identify *what* is being protected and the level of protection provided.
396 These assertions SHOULD apply to [Message Policy Subject]. These assertions MAY apply to [Endpoint
397 Policy Subject] or [Operation Policy Subject]. Where they apply to [Operation Policy Subject] they apply to
398 all messages of that operation. Where they apply to [Endpoint Policy Subject] they apply to all operations
399 of that endpoint.

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

402 4.1 Integrity Assertions

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

406 4.1.1 SignedParts Assertion

407 The SignedParts assertion is used to specify the parts of the message outside of security headers that
408 require integrity protection. This assertion can be satisfied using WSS: SOAP Message Security
409 mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the
410 message over a secure transport protocol like HTTPS. The binding details the exact mechanism by
411 which the protection is provided.

412
413 There MAY be multiple SignedParts assertions present. Multiple SignedParts assertions present within a
414 policy alternative are equivalent to a single SignedParts assertion containing the union of all specified
415 message parts. Note that this assertion does not require that a given part appear in a message, just that if
416 such a part appears, it requires integrity protection.

417 Syntax

```
418 <sp:SignedParts xmlns:sp="..." ... >  
419   <sp:Body />?  
420   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
421   <sp:Attachments />?  
422   ...  
423 </sp:SignedParts>
```

424
425 The following describes the attributes and elements listed in the schema outlined above:

426 /sp:SignedParts

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

430 /sp:SignedParts/sp:Body

431 Presence of this optional empty element indicates that the entire body, that is the soap:Body
432 element, its attributes and content, of the message needs to be integrity protected.

433 /sp:SignedParts/sp:Header

434 Presence of this optional element indicates a specific SOAP header, its attributes and content (or
435 set of such headers) needs to be protected. There may be multiple sp:Header elements within a

436 single sp:SignedParts element. If multiple SOAP headers with the same local name but different
437 namespace names are to be integrity protected multiple sp:Header elements are needed, either
438 as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts assertions.
439 This element only applies to SOAP header elements targeted to the same actor/role as the
440 Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
441 SOAP Header elements targeted to a different actor/role, that may be accomplished using the
442 sp:SignedElements assertion.

443 /sp:SignedParts/sp:Header/@Name

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

447 /sp:SignedParts/sp:Header/@Namespace

448 This required attribute indicates the namespace of the SOAP header(s) to be integrity protected.

449 [/sp:SignedParts/sp:Attachments](#)

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

455

456 4.1.2 SignedElements Assertion

457 The SignedElements assertion is used to specify arbitrary elements in the message that require integrity
458 protection. This assertion can be satisfied using WSS: SOAP Message Security mechanisms or by
459 mechanisms out of scope of SOAP message security, for example by sending the message over a
460 secure transport protocol like HTTPS. The binding details the exact mechanism by which the protection
461 is provided.

462

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

466 Syntax

```
467 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
468   <sp:XPath>xs:string</sp:XPath>+  
469   ...  
470 </sp:SignedElements>
```

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

472 /sp:SignedElements

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

474 /sp:SignedElements/@XPathVersion

475 This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is
476 provided, then XPath 1.0 is assumed.

477 /sp:SignedElements/sp:XPath

478 This element contains a string specifying an XPath expression that identifies the nodes to be
479 integrity protected. The XPath expression is evaluated against the S:Envelope element node of

480 the message. Multiple instances of this element may appear within this assertion and should be
481 treated as separate references in a signature when message security is used.

482 4.2 Confidentiality Assertions

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

486 4.2.1 EncryptedParts Assertion

487 The EncryptedParts assertion is used to specify the parts of the message that require confidentiality. This
488 assertion can be satisfied with WSS: SOAP Message Security mechanisms or by mechanisms out of
489 scope of SOAP message security, for example by sending the message over a secure transport protocol
490 like HTTPS. The binding details the exact mechanism by which the protection is provided.

491

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

496 Syntax

```
497 <sp:EncryptedParts xmlns:sp="..." ... >  
498   <sp:Body/>?  
499   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... /*  
500     <sp:Attachments/>?  
501     ...  
502 </sp:EncryptedParts>
```

503

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

505 /sp:EncryptedParts

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

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

509 /sp:EncryptedParts/sp:Body

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

514 /sp:EncryptedParts/sp:Header

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

523 /sp:EncryptedParts/sp:Header/@Name

524 This optional attribute indicates the local name of the SOAP header to be confidentiality
525 protected. If this attribute is not specified, all SOAP headers whose namespace matches the
526 Namespace attribute are to be protected.

527 `/sp:EncryptedParts/sp:Header/@Namespace`

528 This required attribute indicates the namespace of the SOAP header(s) to be confidentiality
529 protected.

530 `/sp:EncryptedParts/sp:Attachments`

531 Presence of this optional empty element indicates that all SwA (SOAP Messages with
532 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message
533 Security is used to accomplish this, all message parts other than the part containing the primary
534 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security
535 [WSS:SwAProfile1.1].

536 4.2.2 EncryptedElements Assertion

537 The EncryptedElements assertion is used to specify arbitrary elements in the message that require
538 confidentiality protection. This assertion can be satisfied using WSS: SOAP Message Security
539 mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the
540 message over a secure transport protocol like HTTPS. The binding details the exact mechanism by
541 which the protection is provided.

542

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

546 Syntax

```
547 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
548 <sp:XPath>xs:string</sp:XPath>+  
549 ...  
550 </sp:EncryptedElements>
```

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

552 `/sp:EncryptedElements`

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

555 `/sp:EncryptedElements/@XPathVersion`

556 This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is
557 provided, then XPath 1.0 is assumed.

558 `/sp:EncryptedElements/sp:XPath`

559 This element contains a string specifying an XPath expression that identifies the nodes to be
560 confidentiality protected. The XPath expression is evaluated against the S:Envelope element
561 node of the message. Multiple instances of this element may appear within this assertion and
562 should be treated as separate references.

563 4.2.3 ContentEncryptedElements Assertion

564 The ContentEncryptedElements assertion is used to specify arbitrary elements in the message that
565 require confidentiality protection of their content. This assertion can be satisfied using WSS: SOAP
566 Message Security mechanisms or by mechanisms out of scope of SOAP message security, for example
567 by sending the message over a secure transport protocol like HTTPS. The binding details the exact
568 mechanism by which the protection is provided.

569

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

573 **Syntax**

```
574 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
575   <sp:XPath>xs:string</sp:XPath>+  
576   ...  
577 </sp:ContentEncryptedElements>
```

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

579 /sp:ContentEncryptedElements

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

582 /sp:ContentEncryptedElements/@XPathVersion

583 This optional attribute contains a URI which indicates the version of XPath to use.

584 /sp:ContentEncryptedElements/sp:XPath

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

589 **4.3 Required Elements Assertion**

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

592

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

596 **4.3.1 RequiredElements Assertion**

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

599

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

603 **Syntax**

```
604 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
605   <sp:XPath>xs:string</sp:XPath> +  
606   ...  
607 </sp:RequiredElements>
```

608

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

610 /sp:RequiredElements

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

612 /sp:RequiredElements/@XPathVersion

613 This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is
614 provided, then XPath 1.0 is assumed.

615 /sp:RequiredElements/sp:XPath

616 This element contains a string specifying an XPath expression that identifies the header elements
617 that a message MUST contain. The XPath expression is evaluated against the
618 S:Envelope/S:Header element node of the message. Multiple instances of this element may
619 appear within this assertion and should be treated as a combined XPath expression.

620 4.3.2 RequiredParts Assertion

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

624

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

628 Syntax

```
629 <sp:RequiredParts XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
630 <sp:Header Name = "..." Namespace= "..." /> +  
631 </sp:RequiredParts>
```

632

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

634 /sp:RequiredParts/sp:Header

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

636 /sp:RequiredParts/sp:Header/@Name

637 This required attribute indicates the local name of the SOAPHeader that needs to be present in
638 the message.

639 /sp:RequiredParts/sp:Header/@Namespace

640 This required attribute indicates the namespace of the SOAP header that needs to be present in
641 the message.

642 5 Token Assertions

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

647 5.1 Token Inclusion

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

653 5.1.1 Token Inclusion Values

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

<code>http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/Never</code>	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.
<code>http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/Once</code>	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.
<code>http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/AlwaysToRecipient</code>	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.
<code>http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/AlwaysToInitiator</code>	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.
<code>http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/Always</code>	The token MUST be included in all messages sent between the initiator and the recipient. This is the default behavior.

655
656 Note: In examples, the namespace URI is replaced with "...". For example,
657 `.../IncludeToken/Never` is actually `http://docs.oasis-open.org/ws-sx/ws-`
658 `securitypolicy/200512/IncludeToken/Never`. Other token inclusion URI values MAY be defined but are out-
659 of-scope of this specification.

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

662 **5.1.2 Token Inclusion and Token References**

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

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

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

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

679 **5.2 Token Properties**

680 **5.2.1 [Derived Keys] Property**

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

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

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

689 **5.2.2 [Explicit Derived Keys] Property**

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

693 **5.2.3 [Implicit Derived Keys] Property**

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

697 **5.3 Token Assertion Types**

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

699 **5.3.1 UsernameToken Assertion**

700 This element represents a requirement to include a username token.

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

- 702 1. When transport security is not used.
- 703 2. When a plaintext password is used.
- 704 3. When a weak password hash is used.
- 705 4. When the username needs to be protected, e.g. for privacy reasons.

706 When the UsernameToken is to be encrypted it SHOULD be listed as a
707 SignedEncryptedSupportingToken (Section 8.5), EndorsingEncryptedSupportingToken (Section 8.6) or
708 SignedEndorsingEncryptedSupportingToken (Section 8.7).

709

710 Syntax

```
711 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
712   <wsp:Policy xmlns:wsp="...">  
713     (  
714       <sp:NoPassword ... /> |  
715       <sp:HashPassword ... />  
716     ) ?  
717     (  
718       <sp:RequireDerivedKeys /> |  
719       <sp:RequireImplicitDerivedKeys ... /> |  
720       <sp:RequireExplicitDerivedKeys ... />  
721     ) ?  
722     (  
723       <sp:WssUsernameToken10 ... /> |  
724       <sp:WssUsernameToken11 ... />  
725     ) ?  
726     ...  
727   </wsp:Policy> ?  
728   ...  
729 </sp:UsernameToken>
```

730

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

732 /sp:UsernameToken

733 This identifies a UsernameToken assertion.

734 /sp:UsernameToken/@sp:IncludeToken

735 This optional attribute identifies the token inclusion value for this token assertion.

736 /sp:UsernameToken/wsp:Policy

737 This optional element identifies additional requirements for use of the sp:UsernameToken
738 assertion.

739 /sp:UsernameToken/wsp:Policy/sp:NoPassword

740 This optional element is a policy assertion that indicates that the wsse:Password element MUST
741 NOT be present in the Username token.

742 /sp:UsernameToken/wsp:Policy/sp:HashPassword

743 This optional element is a policy assertion that indicates that the wsse:Password element MUST
744 be present in the Username token and that the content of the wsse:Password element MUST
745 contain a hash of the timestamp, nonce and password as defined in [WSS: Username Token
746 Profile].

747 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

748 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
749 and [Implicit Derived Keys] properties for this token to 'true'.

750 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 751 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
 752 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.
 753 /sp:UsernameToken/wsp:Policy/sp:RequireImplicitDerivedKeys
 754 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
 755 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.
 756 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10
 757 This optional element is a policy assertion that indicates that a Username token should be used
 758 as defined in [\[WSS:UsernameTokenProfile1.0\]](#).
 759 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11
 760 This optional element is a policy assertion that indicates that a Username token should be used
 761 as defined in [\[WSS:UsernameTokenProfile1.1\]](#).

762 5.3.2 IssuedToken Assertion

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

767 Syntax

```

768 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
769   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
770   <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
771     ...
772   </sp:RequestSecurityTokenTemplate>
773   <wsp:Policy xmlns:wsp="...">
774     (
775       <sp:RequireDerivedKeys ... /> |
776       <sp:RequireImplicitDerivedKeys ... /> |
777       <sp:RequireExplicitDerivedKeys ... />
778     ) ?
779   <sp:RequireExternalReference ... /> ?
780   <sp:RequireInternalReference ... /> ?
781   ...
782 </wsp:Policy> ?
783   ...
784 </sp:IssuedToken>
  
```

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

786 /sp:IssuedToken

787 This identifies an IssuedToken assertion.

788 /sp:IssuedToken/@sp:IncludeToken

789 This optional attribute identifies the token inclusion value for this token assertion.

790 /sp:IssuedToken/sp:Issuer

791 This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for the
 792 issued token.

793 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

794 This required element contains elements which MUST be copied into the
 795 wst:SecondaryParameters of the RST request sent to the specified issuer. Note: the initiator is
 796 not required to understand the contents of this element.

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

798 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion
799 This optional attribute contains a URI identifying the version of WS-Trust referenced by the
800 contents of this element.

801 /sp:IssuedToken/wsp:Policy
802 This optional element identifies additional requirements for use of the sp:IssuedToken assertion.

803 /sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys
804 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
805 and [Implicit Derived Keys] properties for this token to 'true'.

806 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys
807 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
808 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

809 /sp:IssuedToken/wsp:Policy/sp:RequireImplicitDerivedKeys
810 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
811 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

812 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference
813 This optional element is a policy assertion that indicates whether an internal reference is required
814 when referencing this token.
815 Note: This reference will be supplied by the issuer of the token.

816 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference
817 This optional element is a policy assertion that indicates whether an external reference is required
818 when referencing this token.
819 Note: This reference will be supplied by the issuer of the token.

820 Note: The IssuedToken may or may not be associated with key material and such key material may be
821 symmetric or asymmetric. The Binding assertion will imply the type of key associated with this token.
822 Services may also include information in the sp:RequestSecurityTokenTemplate element to
823 explicitly define the expected key type. See [Appendix B](#) for details of the
824 sp:RequestSecurityTokenTemplate element.

825 **5.3.3 X509Token Assertion**

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

827 **Syntax**

```

828 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
829   <wsp:Policy xmlns:wsp="...">
830     (
831       <sp:RequireDerivedKeys ... /> |
832       <sp:RequireExplicitDerivedKeys ... /> |
833       <sp:RequireImplicitDerivedKeys ... />
834     ) ?
835   <sp:RequireKeyIdentifierReference ... /> ?
836   <sp:RequireIssuerSerialReference ... /> ?
837   <sp:RequireEmbeddedTokenReference ... /> ?
838   <sp:RequireThumbprintReference ... /> ?
839   (
840     <sp:WssX509V3Token10 ... /> |
841     <sp:WssX509Pkcs7Token10 ... /> |
842     <sp:WssX509PkiPathV1Token10 ... /> |
843     <sp:WssX509V1Token11 ... /> |
844     <sp:WssX509V3Token11 ... /> |
845     <sp:WssX509Pkcs7Token11 ... /> |
846     <sp:WssX509PkiPathV1Token11 ... />
847   ) ?
848   ...
849 </wsp:Policy> ?
850   ...
851 </sp:X509Token>

```

852
853 The following describes the attributes and elements listed in the schema outlined above:

854 /sp:X509Token

855 This identifies an X509Token assertion.

856 /sp:X509Token/@sp:IncludeToken

857 This optional attribute identifies the token inclusion value for this token assertion.

858 /sp:X509Token/wsp:Policy

859 This optional element identifies additional requirements for use of the sp:X509Token assertion.

860 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

861 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
862 and [Implicit Derived Keys] properties for this token to 'true'.

863 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

864 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
865 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

866 /sp:X509Token/wsp:Policy/sp:RequireImplicitDerivedKeys

867 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
868 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

869 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

870 This optional element is a policy assertion that indicates that a key identifier reference is required
871 when referencing this token.

872 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

873 This optional element is a policy assertion that indicates that an issuer serial reference is required
874 when referencing this token.

875 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

876 This optional element is a policy assertion that indicates that an embedded token reference is
877 required when referencing this token.

878 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference
 879 This optional element is a policy assertion that indicates that a thumbprint reference is required
 880 when referencing this token.

881 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10
 882 This optional element is a policy assertion that indicates that an X509 Version 3 token should be
 883 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

884 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10
 885 This optional element is a policy assertion that indicates that an X509 PKCS7 token should be
 886 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

887 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10
 888 This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token
 889 should be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

890 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11
 891 This optional element is a policy assertion that indicates that an X509 Version 1 token should be
 892 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

893 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11
 894 This optional element is a policy assertion that indicates that an X509 Version 3 token should be
 895 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

896 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11
 897 This optional element is a policy assertion that indicates that an X509 PKCS7 token should be
 898 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

899 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11
 900 This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token
 901 should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

902 5.3.4 KerberosToken Assertion

903 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

904 Syntax

```

905 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
906   <wsp:Policy xmlns:wsp="...">
907     (
908       <sp:RequireDerivedKeys ... /> |
909       <sp:RequireImplicitDerivedKeys ... /> |
910       <sp:RequireExplicitDerivedKeys ... />
911     ) ?
912     <sp:RequireKeyIdentifierReference ... /> ?
913     (
914       <sp:WssKerberosV5ApReqToken11 ... /> |
915       <sp:WssGssKerberosV5ApReqToken11 ... />
916     ) ?
917     ...
918   </wsp:Policy> ?
919   ...
920 </sp:KerberosToken>
  
```

922
 923 The following describes the attributes and elements listed in the schema outlined above:

924 /sp:KerberosToken

925 This identifies a KerberosV5ApReqToken assertion.

926 /sp:KerberosToken/@sp:IncludeToken

927 This optional attribute identifies the token inclusion value for this token assertion.

928 /sp:KerberosToken/wsp:Policy

929 This optional element identifies additional requirements for use of the sp:KerberosToken
930 assertion.

931 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

932 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
933 and [Implicit Derived Keys] properties for this token to 'true'.

934 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

935 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
936 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

937 /sp:KerberosToken/wsp:Policy/sp:RequireImplicitDerivedKeys

938 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
939 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

940 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

941 This optional element is a policy assertion that indicates that a key identifier reference is required
942 when referencing this token.

943 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

944 This optional element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ token
945 should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

946 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

947 This optional element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-REQ
948 token should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

949 5.3.5 SpnegoContextToken Assertion

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

952 Syntax

```

953 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
954   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> ?
955   <wsp:Policy xmlns:wsp="...">
956     (
957       <sp:RequireDerivedKeys ... /> |
958       <sp:RequireImplicitDerivedKeys ... /> |
959       <sp:RequireExplicitDerivedKeys ... />
960     ) ?
961     ...
962   </wsp:Policy> ?
963   ...
964 </sp:SpnegoContextToken>

```

965

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

967 /sp:SpnegoContextToken

968 This identifies a SpnegoContextToken assertion.

969 /sp:SpnegoContextToken/@sp:IncludeToken

970 This optional attribute identifies the token inclusion value for this token assertion.

971 /sp:SpnegoContextToken/sp:Issuer

972 This optional element, of type `wsa:EndpointReferenceType`, contains a reference to the issuer for the

973 Spnego Context Token.

974 /sp:SpnegoContextToken/wsp:Policy

975 This optional element identifies additional requirements for use of the `sp:SpnegoContextToken`

976 assertion.

977 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

978 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]

979 and [Implicit Derived Keys] properties for this token to 'true'.

980 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

981 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]

982 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

983 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImplicitDerivedKeys

984 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]

985 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

986 5.3.6 SecurityContextToken Assertion

987 This element represents a requirement for a SecurityContextToken token.

988 Syntax

```

989 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
990 <wsp:Policy xmlns:wsp="...">
991   (
992     <sp:RequireDerivedKeys ... /> |
993     <sp:RequireImplicitDerivedKeys ... /> |
994     <sp:RequireExplicitDerivedKeys ... />
995   ) ?
996   <sp:RequireExternalUriReference ... /> ?
997   <sp:SC200502SecurityContextToken ... /> ?
998   ...
999 </wsp:Policy> ?
1000 ...
1001 </sp:SecurityContextToken>

```

1002

1003 The following describes the attributes and elements listed in the schema outlined above:

1004 /sp:SecurityContextToken

1005 This identifies a SecurityContextToken assertion.

1006 /sp:SecurityContextToken/@sp:IncludeToken

1007 This optional attribute identifies the token inclusion value for this token assertion.

1008 /sp:SecurityContextToken/wsp:Policy

1009 This optional element identifies additional requirements for use of the `sp:SecurityContextToken`

1010 assertion.

1011 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1012 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]

1013 and [Implicit Derived Keys] properties for this token to 'true'.

1014 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1015 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1016 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

1017 /sp:SecurityContextToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1018 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
1019 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1020 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1021 This optional element is a policy assertion that indicates that an external URI reference is
1022 required when referencing this token.

1023 /sp:SecurityContextToken/wsp:Policy/sp:SC200502SecurityContextToken

1024 This optional element is a policy assertion that indicates that a Security Context Token should be
1025 used as defined in [\[WS-SecureConversation\]](#).

1026

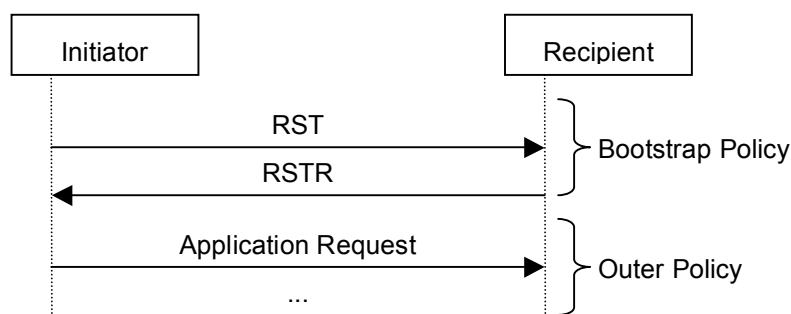
1027 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
1028 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
1029 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
1030 sp:SecureConversationToken or the sp:IssuedToken assertion should be used instead.

1031 **5.3.7 SecureConversationToken Assertion**

1032 This element represents a requirement for a Security Context Token retrieved from the indicated issuer
1033 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
1034 service endpoint address.

1035

1036 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1037 the target service and may not have a separate port (with separate policy), this assertion SHOULD
1038 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1039 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1040 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1041 below.



1042

1043 **Syntax**


```

1044 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1045   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
1046   <wsp:Policy xmlns:wsp="...">
1047     (
1048       <sp:RequireDerivedKeys ... /> |
1049       <sp:RequireImplicitDerivedKeys ... /> |
1050       <sp:RequireExplicitDerivedKeys ... />
1051     ) ?
1052     <sp:RequireExternalUriReference ... /> ?
1053     <sp:SC200502SecurityContextToken ... /> ?
1054     <sp:BootstrapPolicy ... > ?
1055     <wsp:Policy> ... </wsp:Policy>
1056   </sp:BootstrapPolicy>
1057 </wsp:Policy> ?
1058   ...
1059 </sp:SecureConversationToken>

```

1060

1061 The following describes the attributes and elements listed in the schema outlined above:

1062 /sp:SecureConversationToken

1063 This identifies a SecureConversationToken assertion.

1064 /sp:SecureConversationToken/@sp:IncludeToken

1065 This optional attribute identifies the token inclusion value for this token assertion.

1066 /sp:SecureConversationToken/sp:Issuer

1067 This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for the
1068 Security Context Token.

1069 /sp:SecureConversationToken/wsp:Policy

1070 This optional element identifies additional requirements for use of the
1071 sp:SecureConversationToken assertion.

1072 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1073 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1074 and [Implicit Derived Keys] properties for this token to 'true'.

1075 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1078 /sp:SecureConversationToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1079 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
1080 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1081 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1082 This optional element is a policy assertion that indicates that an external URI reference is
1083 required when referencing this token.

1084 /sp:SecureConversationToken/wsp:Policy/sp:SC200502SecurityContextToken

1085 This optional element is a policy assertion that indicates that a Security Context Token should be
1086 used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1087 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1088 This optional element is a policy assertion that contains the policy indicating the requirements for
1089 obtaining the Security Context Token.

1090 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1091 This element contains the security binding requirements for obtaining the Security Context Token. It
1092 will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with protection
1093 assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that are to be
1094 protected.

1095 **Example**

```
1096 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
1097   <sp:SymmetricBinding>  
1098     <wsp:Policy>  
1099       <sp:ProtectionToken>  
1100         <wsp:Policy>  
1101           <sp:SecureConversationToken>  
1102             <sp:Issuer>  
1103               <wsa:Address>http://example.org/sts</wsa:Address>  
1104             </sp:Issuer>  
1105           <wsp:Policy>  
1106             <sp:SC10SecurityContextToken />  
1107           <sp:BootstrapPolicy>  
1108             <wsp:Policy>  
1109               <sp:AsymmetricBinding>  
1110                 <wsp:Policy>  
1111                   <sp:InitiatorToken>  
1112                     ...  
1113                   </sp:InitiatorToken>  
1114                   <sp:RecipientToken>  
1115                     ...  
1116                   </sp:RecipientToken>  
1117                 </wsp:Policy>  
1118               </sp:AsymmetricBinding>  
1119             <sp:SignedParts>  
1120               ...  
1121             </sp:SignedParts>  
1122           </wsp:Policy>  
1123         </sp:BootstrapPolicy>  
1124       </wsp:Policy>  
1125     </sp:SecureConversationToken>  
1126   </wsp:Policy>  
1127 </sp:ProtectionToken>  
1128 </wsp:Policy>  
1129 ...  
1130 </wsp:Policy>  
1131 </sp:SymmetricBinding>  
1132 <sp:SignedParts>  
1133   ...  
1134 </sp:SignedParts>  
1135   ...  
1136 </wsp:Policy>
```

1137 **5.3.8 SamlToken Assertion**

1138 This element represents a requirement for a SAML token.

1139 **Syntax**

```

1140 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1141   <wsp:Policy xmlns:wsp="...">
1142     (
1143       <sp:RequireDerivedKeys ... /> |
1144       <sp:RequireImplicitDerivedKeys ... /> |
1145       <sp:RequireExplicitDerivedKeys ... />
1146     ) ?
1147   <sp:RequireKeyIdentifierReference ... /> ?
1148   (
1149     <sp:WssSamlV11Token10 ... /> |
1150     <sp:WssSamlV11Token11 ... /> |
1151     <sp:WssSamlV20Token11 ... />
1152   ) ?
1153   ...
1154 </wsp:Policy> ?
1155   ...
1156 </sp:SamlToken>

```

1157

1158 The following describes the attributes and elements listed in the schema outlined above:

1159 /sp:SamlToken

1160 This identifies a SamlToken assertion.

1161 /sp:SamlToken/@sp:IncludeToken

1162 This optional attribute identifies the token inclusion value for this token assertion.

1163 /sp:SamlToken/wsp:Policy

1164 This optional element identifies additional requirements for use of the sp:SamlToken assertion.

1165 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

1166 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1167 and [Implicit Derived Keys] properties for this token to 'true'.

1168 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1169 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
 1170 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

1171 /sp:SamlToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1172 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
 1173 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1174 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1175 This optional element is a policy assertion that indicates that a key identifier reference is required
 1176 when referencing this token.

1177 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10

1178 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1179 used as defined in [\[WSS:SAMLTokenProfile1.0\]](#).

1180 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11

1181 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1182 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1183 /sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11

1184 This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be
 1185 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1186

1187 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1188 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1189 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion should
1190 be used instead.

1191 5.3.9 RelToken Assertion

1192 This element represents a requirement for a REL token.

1193 Syntax

```
1194 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1195   <wsp:Policy xmlns:wsp="...">  
1196     (  
1197       <sp:RequireDerivedKeys ... /> |  
1198       <sp:RequireImplicitDerivedKeys ... /> |  
1199       <sp:RequireExplicitDerivedKeys ... />  
1200     ) ?  
1201     <sp:RequireKeyIdentifierReference ... /> ?  
1202     (  
1203       <sp:WssRelV10Token10 ... /> |  
1204       <sp:WssRelV20Token10 ... /> |  
1205       <sp:WssRelV10Token11 ... /> |  
1206       <sp:WssRelV20Token11 ... />  
1207     ) ?  
1208     ...  
1209   </wsp:Policy> ?  
1210   ...  
1211 </sp:RelToken>
```

1212
1213 The following describes the attributes and elements listed in the schema outlined above:

1214 /sp:RelToken

1215 This identifies a RelToken assertion.

1216 /sp:RelToken/@sp:IncludeToken

1217 This optional attribute identifies the token inclusion value for this token assertion.

1218 /sp:RelToken/wsp:Policy

1219 This optional element identifies additional requirements for use of the sp:RelToken assertion.

1220 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1221 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1222 and [Implicit Derived Keys] property for this token to 'true'.

1223 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1224 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1225 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

1226 /sp:RelToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1227 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
1228 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1229 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1230 This optional element is a policy assertion that indicates that a key identifier reference is required
1231 when referencing this token.

1232 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1233 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1234 used as defined in [\[WSS:RELTOKENPROFILE1.0\]](#).

1235 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1236 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1237 used as defined in [\[WSS:RELTOKENPROFILE1.0\]](#).

1238 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1239 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1240 used as defined in [\[WSS:RELTOKENPROFILE1.1\]](#).

1241 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1242 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1243 used as defined in [\[WSS:RELTOKENPROFILE1.1\]](#).

1244

1245 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
1246 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1247 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion should
1248 be used instead.

1249 5.3.10 HttpsToken Assertion

1250 This element represents a requirement for a transport binding to support the use of HTTPS.

1251 Syntax

```
1252 <sp:HttpsToken xmlns:sp="..." ... >  
1253   <wsp:Policy xmlns:wsp="...">  
1254     (  
1255       <sp:HttpBasicAuthentication /> |  
1256       <sp:HttpDigestAuthentication /> |  
1257       <sp:RequireClientCertificate /> |  
1258       ...  
1259     )?  
1260     ...  
1261   </wsp:Policy> ?  
1262   ...  
1263 </sp:HttpsToken>
```

1264 The following describes the attributes and elements listed in the schema outlined above:

1265 /sp:HttpsToken

1266 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1267 supported.

1268 /sp:HttpsToken/wsp:Policy

1269 This optional element identifies additional requirements for use of the sp:HttpsToken assertion.

1270 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1271 This optional element is a policy assertion that indicates that the client MUST use HTTP Basic
1272 Authentication [\[RFC2068\]](#) to authenticate to the service.

1273 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1274 This optional element is a policy assertion that indicates that the client MUST use HTTP Digest
1275 Authentication [\[RFC2068\]](#) to authenticate to the service.

1276 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1277 This optional element is a policy assertion that indicates that the client MUST provide a certificate
1278 when negotiating the HTTPS session.

1279 6 Security Binding Properties

1280 This section defines the various properties or conditions of a security binding, their semantics, values and
1281 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are
1282 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that
1283 populates a value of a property appears in a policy, that property is set to the value indicated by the
1284 assertion. The security binding then uses the value of the property to control its behavior. The properties
1285 listed here are common to the various security bindings described in Section 7. Assertions that define
1286 values for these properties are defined in Section 7. The following properties are used by the security
1287 binding assertions.

1288 6.1 [Algorithm Suite] Property

1289 This property specifies the algorithm suite required for performing cryptographic operations with
1290 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and
1291 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This
1292 property defines the set of available algorithms. The value of this property is typically referenced by a
1293 security binding and is used to specify the algorithms used for all message level cryptographic operations
1294 performed under the security binding.

1295 Note: In some cases, this property MAY be referenced under a context other than a security binding and
1296 used to control the algorithms used under that context. For example, supporting token assertions define
1297 such a context. In such contexts, the specified algorithms still apply to message level cryptographic
1298 operations.

1299 An algorithm suite defines values for each of the following operations and properties:

- 1300 • [Sym Sig] Symmetric Key Signature
- 1301 • [Asym Sig] Signature with an asymmetric key
- 1302 • [Dig] Digest
- 1303 • [Enc] Encryption
- 1304 • [Sym KW] Symmetric Key Wrap
- 1305 • [Asym KW] Asymmetric Key Wrap
- 1306 • [Comp Key] Computed key
- 1307 • [Enc KD] Encryption key derivation
- 1308 • [Sig KD] Signature key derivation
- 1309 • [Min SKL] Minimum symmetric key length
- 1310 • [Max SKL] Maximum symmetric key length
- 1311 • [Min AKL] Minimum asymmetric key length
- 1312 • [Max AKL] Maximum asymmetric key length

1313

1314 The following table provides abbreviations for the algorithm URI used in the table below:

Abbreviation	Algorithm URI
HmacSha1	http://www.w3.org/2000/09/xmlsig#hmac-sha1
RsaSha1	http://www.w3.org/2000/09/xmlsig#rsa-sha1
Sha1	http://www.w3.org/2000/09/xmlsig#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/2001/10/xml-c14n#>
ExC14n <http://www.w3.org/2001/10/xml-exc-c14n#>
SNT <http://www.w3.org/TR/soap12-n11n>
STR10 <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>

1315

1316 The tables below show all the base algorithm suites defined by this specification. This table defines

1317 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

1318 This table defines additional properties whose values can be specified along with the default value for that
1319 property.

Property	Algorithm / Value
[C14n Algorithm]	ExC14n
[Soap Norm]	None
[STR Trans]	None
[XPath]	None

1320 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
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

1321 6.2 [Timestamp] Property

1322 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`
1323 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected
1324 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be
1325 present. The default value for this property is 'false'.

1326 6.3 [Protection Order] Property

1327 This property indicates the order in which integrity and confidentiality are applied to the message, in
1328 cases where both integrity and confidentiality are required:

EncryptBeforeSigning	Signature MUST be 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.

1329 The default value for this property is 'SignBeforeEncrypting'.

1330 6.4 [Signature Protection] Property

1331 This boolean property specifies whether the signature must be encrypted. If the value is 'true', the primary
1332 signature MUST be encrypted and any signature confirmation elements MUST also be encrypted. If the
1333 value is 'false', the primary signature MUST NOT be encrypted and any signature confirmation elements
1334 MUST NOT be encrypted. The default value for this property is 'false'.

1335 6.5 [Token Protection] Property

1336 This boolean property specifies whether signatures must cover the token used to generate that signature.
1337 If the value is 'true', then each token used to generate a signature MUST be covered by that signature. If
1338 the value is 'false', then the token MUST NOT be covered by the signature. Note that in cases where
1339 derived keys are used the 'main' token, and NOT the derived key token, is covered by the signature. It is
1340 recommended that assertions that define values for this property apply to [Endpoint Policy Subject]. The
1341 default value for this property is 'false'.

1342 6.6 [Entire Header and Body Signatures] Property

1343 This boolean property specifies whether signature digests over the SOAP body and SOAP headers must
1344 only cover the entire body and entire header elements. If the value is 'true', then each digest over the
1345 SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In
1346 addition each digest over a SOAP header MUST be over an actual header element and not a descendant
1347 of a header element. This restriction does not specifically apply to the `wsse:Security` header. However
1348 signature digests over child elements of the `wsse:Security` header MUST be over the entire child element
1349 and not a descendant of that element. If the value is 'false', then signature digests MAY be over a
1350 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to
1351 'true' mitigates against some possible re-writing attacks. It is recommended that assertions that define
1352 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

1353 6.7 [Security Header Layout] Property

1354 This property indicates which layout rules to apply when adding items to the security header. The
1355 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 wsse: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 wsse:Timestamp. Note that the [Timestamp] property MUST also be set to 'true' in this case.

1356

1357 6.7.1 Strict Layout Rules for WSS 1.0

- 1358 1. Tokens that are included in the message MUST be declared before use. For example:
- 1359 a. A local signing token MUST occur before the signature that uses it.
- 1360 b. A local token serving as the source token for a derived key token MUST occur before that
- 1361 derived key token.
- 1362 c. A local encryption token MUST occur before the reference list that points to
- 1363 xenc:EncryptedData elements that use it.
- 1364 d. If the same token is used for both signing and encryption, then it should appear before
- 1365 the ds:Signature and xenc:ReferenceList elements in the security header that are
- 1366 generated using the token.
- 1367 2. Signed elements inside the security header MUST occur before the signature that signs them.
- 1368 For example:
- 1369 a. A timestamp MUST occur before the signature that signs it.
- 1370 b. A Username token (usually in encrypted form) MUST occur before the signature that
- 1371 signs it.
- 1372 c. A primary signature MUST occur before the supporting token signature that signs the
- 1373 primary signature's signature value element.
- 1374 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
- 1375 has the same order requirements as the source plain text element, unless requirement 4
- 1376 indicates otherwise. For example, an encrypted primary signature MUST occur before any
- 1377 supporting token signature per 2.c above and an encrypted token has the same ordering
- 1378 requirements as the unencrypted token.

1379 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1380 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1381 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1382 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1383 Layout Rules for WSS 1.1

- 1384 1. Tokens that are included in the message MUST be declared before use. For example:
1385 a. A local signing token MUST occur before the signature that uses it.
1386 b. A local token serving as the source token for a derived key token MUST occur before that
1387 derived key token.
1388 c. A local encryption token MUST occur before the reference list that points to
1389 xenc:EncryptedData elements that use it.
1390 d. If the same token is used for both signing and encryption, then it should appear before
1391 the ds:Signature and xenc:ReferenceList elements in the security header that are
1392 generated using the token.
- 1393 2. Signed elements inside the security header MUST occur before the signature that signs them.
1394 For example:
1395 a. A timestamp MUST occur before the signature that signs it.
1396 b. A Username token (usually in encrypted form) MUST occur before the signature that
1397 signs it.
1398 c. A primary signature MUST occur before the supporting token signature that signs the
1399 primary signature's signature value element.
1400 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1401 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1402 has the same order requirements as the source plain text element, unless requirement 4
1403 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1404 supporting token signature per 2.c above and an encrypted token has the same ordering
1405 requirements as the unencrypted token.
- 1406 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1407 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1408 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1409 However, the xenc:ReferenceList is not required to appear before independently encrypted
1410 tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1411 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1412 1.1] MUST obey rule 1 above.

1413 7 Security Binding Assertions

1414 The appropriate representation of the different facets of security mechanisms requires distilling the
1415 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy
1416 scope of assertions defined in this section is the policy scope of their containing element.

1417 7.1 AlgorithmSuite Assertion

1418 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]
1419 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

1420 Syntax

```
1421 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1422   <wsp:Policy xmlns:wsp="...">  
1423     (<sp:Basic256 ... /> |  
1424     <sp:Basic192 ... /> |  
1425     <sp:Basic128 ... /> |  
1426     <sp:TripleDes ... /> |  
1427     <sp:Basic256Rsa15 ... /> |  
1428     <sp:Basic192Rsa15 ... /> |  
1429     <sp:Basic128Rsa15 ... /> |  
1430     <sp:TripleDesRsa15 ... /> |  
1431     <sp:Basic256Sha256 ... /> |  
1432     <sp:Basic192Sha256 ... /> |  
1433     <sp:Basic128Sha256 ... /> |  
1434     <sp:TripleDesSha256 ... /> |  
1435     <sp:Basic256Sha256Rsa15 ... /> |  
1436     <sp:Basic192Sha256Rsa15 ... /> |  
1437     <sp:Basic128Sha256Rsa15 ... /> |  
1438     <sp:TripleDesSha256Rsa15 ... /> |  
1439     ...)  
1440     <sp:InclusiveC14N ... /> ?  
1441     <sp:SOAPNormalization10 ... /> ?  
1442     <sp:STRTransform10 ... /> ?  
1443     (<sp:XPath10 ... /> |  
1444     <sp:XPathFilter20 ... /> |  
1445     <sp:AbsXPath ... /> |  
1446     ...)?  
1447     ...  
1448   </wsp:Policy>  
1449   ...  
1450 </sp:AlgorithmSuite>
```

1451
1452 The following describes the attributes and elements listed in the schema outlined above:

1453 /sp:AlgorithmSuite

1454 This identifies an AlgorithmSuite assertion.

1455 /sp:AlgorithmSuite/wsp:Policy

1456 This element contains one or more policy assertions that indicate the specific algorithm suite to use.

1457 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1458 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1459 to 'Basic256'.

1460 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

1461 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1462 to 'Basic192'.

1463 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1464 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1465 to 'Basic128'.

1466 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1467 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1468 to 'TripleDes'.

1469 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1470 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1471 to 'Basic256Rsa15'.

1472 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1473 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1474 to 'Basic192Rsa15'.

1475 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1476 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1477 to 'Basic128Rsa15'.

1478 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1479 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1480 to 'TripleDesRsa15'.

1481 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1482 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1483 to 'Basic256Sha256'.

1484 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1485 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1486 to 'Basic192Sha256'.

1487 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1488 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1489 to 'Basic128Sha256'.

1490 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1491 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1492 to 'TripleDesSha256'.

1493 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1494 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1495 to 'Basic256Sha256Rsa15'.

1496 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1497 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1498 to 'Basic192Sha256Rsa15'.

1499 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1500 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1501 to 'Basic128Sha256Rsa15'.

1502 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1503 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1504 to 'TripleDesSha256Rsa15'.

1505 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1506 This optional element is a policy assertion that indicates that the [C14N] property of an algorithm
1507 suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N] property is
1508 'ExcC14N'.

1509 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1510 This optional element is a policy assertion that indicates that the [SOAP Norm] property is set to
1511 'SNT'.

1512 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1513 This optional element is a policy assertion that indicates that the [STR Transform] property is set
1514 to 'STRT10'.

1515 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1516 This optional element is a policy assertion that indicates that the [XPath] property is set to 'XPath'.

1517 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1518 This optional element is a policy assertion that indicates that the [XPath] property is set to
1519 'XPath20'.

1520 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1521 This optional element is a policy assertion that indicates that the [XPath] property is set to
1522 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1523

1524 7.2 Layout Assertion

1525 This assertion indicates a requirement for a particular security header layout as defined under the
1526 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1527 containing assertion.

1528 Syntax

```
1529 <sp:Layout xmlns:sp="..." ... >  
1530   <wsp:Policy xmlns:wsp="...">  
1531     <sp:Strict ... /> |  
1532     <sp:Lax ... /> |  
1533     <sp:LaxTsFirst ... /> |  
1534     <sp:LaxTsLast ... /> |  
1535     ...  
1536   </wsp:Policy>  
1537   ...  
1538 </sp:Layout>
```

1539

1540 The following describes the attributes and elements listed in the schema outlined above:

1541 /sp:Layout

1542 This identifies a Layout assertion.

1543 /sp:Layout/wsp:Policy

1544 This element contains one or more policy assertions that indicate the specific security header layout
1545 to use.

1546 /sp:Layout/wsp:Policy/sp:Strict

1547 This optional element is a policy assertion that indicates that the [Security Header Layout]
1548 property is set to 'Strict'.

1549 /sp:Layout/wsp:Policy/sp:Lax

1550 This optional element is a policy assertion that indicates that the [Security Header Layout]
1551 property is set to 'Lax'.

1552 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1553 This optional element is a policy assertion that indicates that the [Security Header Layout]
1554 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1555 'true' by the presence of an sp:IncludeTimestamp assertion.

1556 /sp:Layout/wsp:Policy/sp:LaxTsLast

1557 This optional element is a policy assertion that indicates that the [Security Header Layout]
1558 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
1559 'true' by the presence of an sp:IncludeTimestamp assertion.

1560 7.3 TransportBinding Assertion

1561 The TransportBinding assertion is used in scenarios in which message protection and security correlation
1562 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
1563 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
1564 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
1565 MUST apply to [Endpoint Policy Subject].

1566 Syntax

```
1567 <sp:TransportBinding xmlns:sp="..." ... >  
1568   <wsp:Policy xmlns:wsp="...">  
1569     <sp:TransportToken ... >  
1570       <wsp:Policy> ... </wsp:Policy>  
1571       ...  
1572     </sp:TransportToken>  
1573     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1574     <sp:Layout ... > ... </sp:Layout> ?  
1575     <sp:IncludeTimestamp ... /> ?  
1576     ...  
1577   </wsp:Policy>  
1578   ...  
1579 </sp:TransportBinding>
```

1580
1581 The following describes the attributes and elements listed in the schema outlined above:

1582 /sp:TransportBinding

1583 This identifies a TransportBinding assertion.

1584 /sp:TransportBinding/wsp:Policy

1585 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
1586 assertion.

1587 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1588 This required element is a policy assertion that indicates a requirement for a Transport Token.
1589 The specified token populates the [Transport Token] property and indicates how the transport is
1590 secured.

1591 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1592 This indicates a nested policy that identifies the type of Transport Token to use.

1593 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

1594 This required element is a policy assertion that indicates a value that populates the [Algorithm
1595 Suite] property. See Section 6.1 for more details.

1596 /sp:TransportBinding/wsp:Policy/sp:Layout

1597 This optional element is a policy assertion that indicates a value that populates the [Security
1598 Header Layout] property. See Section 6.7 for more details.

1599 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

1600 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1601 'true'.

1602 7.4 SymmetricBinding Assertion

1603 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
1604 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
1605 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
1606 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
1607 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
1608 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
1609 properties and is used as the basis for both encryption and signature in both directions. This assertion
1610 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

1611 Syntax

```
1612 <sp:SymmetricBinding xmlns:sp="..." ... >  
1613   <wsp:Policy xmlns:wsp="...">  
1614     (  
1615       <sp:EncryptionToken ... >  
1616         <wsp:Policy> ... </wsp:Policy>  
1617       </sp:EncryptionToken>  
1618       <sp:SignatureToken ... >  
1619         <wsp:Policy> ... </wsp:Policy>  
1620       </sp:SignatureToken>  
1621     ) | (  
1622       <sp:ProtectionToken ... >  
1623         <wsp:Policy> ... </wsp:Policy>  
1624       </sp:ProtectionToken>  
1625     )  
1626     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1627     <sp:Layout ... > ... </sp:Layout> ?  
1628     <sp:IncludeTimestamp ... /> ?  
1629     <sp:EncryptBeforeSigning ... /> ?  
1630     <sp:EncryptSignature ... /> ?  
1631     <sp:ProtectTokens ... /> ?  
1632     <sp:OnlySignEntireHeadersAndBody ... /> ?  
1633     ...  
1634   </wsp:Policy>  
1635   ...  
1636 </sp:SymmetricBinding>
```

1637

1638 The following describes the attributes and elements listed in the schema outlined above:

1639 /sp:SymmetricBinding

1640 This identifies a SymmetricBinding assertion.

1641 /sp:SymmetricBinding/wsp:Policy

1642 This indicates a nested `wsp:Policy` element that defines the behavior of the SymmetricBinding
1643 assertion.

1644 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken

1645 This optional element is a policy assertion that indicates a requirement for an Encryption Token.
1646 The specified token populates the [Encryption Token] property and is used for encryption. It is an
1647 error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

1648 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
1649 The policy contained here MUST identify exactly one token to use for encryption.

1650 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
1651 This optional element is a policy assertion that indicates a requirement for a Signature Token.
1652 The specified token populates the [Signature Token] property and is used for the message
1653 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
1654 specified.

1655 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
1656 The policy contained here MUST identify exactly one token to use for signatures.

1657 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
1658 This optional element is a policy assertion that indicates a requirement for a Protection Token.
1659 The specified token populates the [Encryption Token] and [Signature Token properties] and is
1660 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
1661 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
1662 specified.

1663 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
1664 The policy contained here MUST identify exactly one token to use for protection.

1665 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
1666 This required element is a policy assertion that indicates a value that populates the [Algorithm
1667 Suite] property. See Section 6.1 for more details.

1668 /sp:SymmetricBinding/wsp:Policy/sp:Layout
1669 This optional element is a policy assertion that indicates a value that populates the [Security
1670 Header Layout] property. See Section 6.7 for more details.

1671 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
1672 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1673 'true'.

1674 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
1675 This optional element is a policy assertion that indicates that the [Protection Order] property is set
1676 to 'EncryptBeforeSigning'.

1677 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
1678 This optional element is a policy assertion that indicates that the [Signature Protection] property is
1679 set to 'true'.

1680 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
1681 This optional element is a policy assertion that indicates that the [Token Protection] property is
1682 set to 'true'.

1683 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
1684 This optional element is a policy assertion that indicates that the [Entire Header And Body
1685 Signatures] property is set to 'true'.

1686 7.5 AsymmetricBinding Assertion

1687 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means
1688 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly
1689 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and
1690 signature. However it is also common practice to use distinct keys for encryption and signature, because
1691 of their different lifecycles.

1692
1693 This binding enables either of these practices by means of four binding specific token properties: [Initiator
1694 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption
1695 Token].

1696
1697 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator
1698 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient
1699 Encryption Token] will both refer to the same token.

1700
1701 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator
1702 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient
1703 Encryption Token] will refer to different tokens.

1704
1705 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the
1706 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response
1707 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the
1708 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for
1709 the message encryption from initiator to the recipient. Note that in each case, the token is associated with
1710 the party (initiator or recipient) who knows the secret.

1711 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy
1712 Subject].

1713 Syntax

```
1714 <sp:AsymmetricBinding xmlns:sp="..." ... >  
1715   <wsp:Policy xmlns:wsp="...">  
1716     (  
1717       <sp:InitiatorToken>  
1718         <wsp:Policy> ... </wsp:Policy>  
1719       </sp:InitiatorToken>  
1720     ) | (  
1721       <sp:InitiatorSignatureToken>  
1722         <wsp:Policy> ... </wsp:Policy>  
1723       </sp:InitiatorSignatureToken>  
1724       <sp:InitiatorEncryptionToken>  
1725         <wsp:Policy> ... </wsp:Policy>  
1726       </sp:InitiatorEncryptionToken>  
1727     )  
1728     (  
1729       <sp:RecipientToken>  
1730         <wsp:Policy> ... </wsp:Policy>  
1731       </sp:RecipientToken>  
1732     ) | (  
1733       <sp:RecipientSignatureToken>  
1734         <wsp:Policy> ... </wsp:Policy>  
1735       </sp:RecipientSignatureToken>  
1736       <sp:RecipientEncryptionToken>  
1737         <wsp:Policy> ... </wsp:Policy>
```

```

1738     </sp:RecipientEncryptionToken>
1739   )
1740   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1741   <sp:Layout ... > ... </sp:Layout> ?
1742   <sp:IncludeTimestamp ... /> ?
1743   <sp:EncryptBeforeSigning ... /> ?
1744   <sp:EncryptSignature ... /> ?
1745   <sp:ProtectTokens ... /> ?
1746   <sp:OnlySignEntireHeadersAndBody ... /> ?
1747   ...
1748 </wsp:Policy>
1749   ...
1750 </sp:AsymmetricBinding>

```

- 1751
- 1752 The following describes the attributes and elements listed in the schema outlined above:
- 1753 `/sp:AsymmetricBinding`
- 1754 This identifies a `AsymmetricBinding` assertion.
- 1755 `/sp:AsymmetricBinding/wsp:Policy`
- 1756 This indicates a nested `wsp:Policy` element that defines the behavior of the `AsymmetricBinding` assertion.
- 1757
- 1758 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken`
- 1759 This optional element is a policy assertion that indicates a requirement for an Initiator Token. The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token] properties and is used for the message signature from initiator to recipient, and encryption from recipient to initiator.
- 1760
- 1761
- 1762
- 1763 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy`
- 1764 The policy contained here MUST identify one or more token assertions.
- 1765 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken`
- 1766 This optional element is a policy assertion that indicates a requirement for an Initiator Signature Token. The specified token populates the [Initiator Signature Token] property and is used for the message signature from initiator to recipient.
- 1767
- 1768
- 1769 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy`
- 1770 The policy contained here MUST identify one or more token assertions.
- 1771 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken`
- 1772 This optional element is a policy assertion that indicates a requirement for an Initiator Encryption Token. The specified token populates the [Initiator Encryption Token] property and is used for the message encryption from recipient to initiator.
- 1773
- 1774
- 1775 `/sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy`
- 1776 The policy contained here MUST identify one or more token assertions.
- 1777 `/sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken`
- 1778 This optional element is a policy assertion that indicates a requirement for a Recipient Token. The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token] property and is used for encryption from initiator to recipient, and for the message signature from recipient to initiator.
- 1779
- 1780
- 1781
- 1782 `/sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy`
- 1783 The policy contained here MUST identify one or more token assertions.
- 1784 `/sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken`

1785 This optional element is a policy assertion that indicates a requirement for a Recipient Signature
1786 Token. The specified token populates the [Recipient Signature Token] property and is used for
1787 the message signature from Recipient to recipient.

1788 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
1789 The policy contained here MUST identify one or more token assertions.

1790 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
1791 This optional element is a policy assertion that indicates a requirement for a Recipient Encryption
1792 Token. The specified token populates the [Recipient Encryption Token] property and is used for
1793 the message encryption from recipient to Recipient.

1794 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
1795 The policy contained here MUST identify one or more token assertions.

1796 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
1797 This required element is a policy assertion that indicates a value that populates the [Algorithm
1798 Suite] property. See Section 6.1 for more details.

1799 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
1800 This optional element is a policy assertion that indicates a value that populates the [Security
1801 Header Layout] property. See Section 6.7 for more details.

1802 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
1803 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1804 'true'.

1805 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
1806 This optional element is a policy assertion that indicates that the [Protection Order] property is set
1807 to 'EncryptBeforeSigning'.

1808 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
1809 This optional element is a policy assertion that indicates that the [Signature Protection] property is
1810 set to 'true'.

1811 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
1812 This optional element is a policy assertion that indicates that the [Token Protection] property is
1813 set to 'true'.

1814 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
1815 This optional element is a policy assertion that indicates that the [Entire Header And Body
1816 Signatures] property is set to 'true'.

1817 8 Supporting Tokens

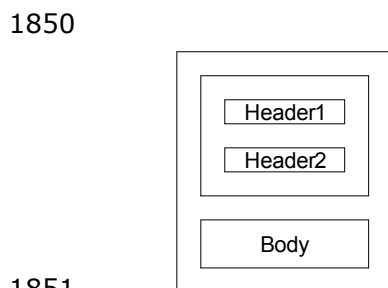
1818 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to
1819 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore
1820 be referred to as the “message signature”. Additional tokens may be specified to augment the claims
1821 provided by the token associated with the “message signature” provided by the Security Binding. This
1822 section defines seven properties related to supporting token requirements which may be referenced by a
1823 Security Binding: [Supporting Tokens], [Signed Supporting Tokens], [Endorsing Supporting Tokens],
1824 [Signed Endorsing Supporting Tokens], [Signed Encrypted Supporting Tokens], [Endorsing Encrypted
1825 Supporting Tokens] and [Signed Endorsing Encrypted Supporting Tokens]. Seven assertions are defined
1826 to populate those properties: SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,
1827 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,
1828 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These
1829 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy
1830 Subject] or [Operation Policy Subject].

1831
1832 Supporting tokens may be specified at a different scope than the binding assertion which provides
1833 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while
1834 the supporting tokens might be defined at the scope of a message. When assertions that populate this
1835 property are defined in overlapping scopes, the sender should merge the requirements by including all
1836 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

1837
1838 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the
1839 tokens should sign and encrypt the various message parts. In such cases ordering of elements (tokens,
1840 signatures, reference lists etc.) in the security header would be used to determine which order signature
1841 and encryptions occurred in.

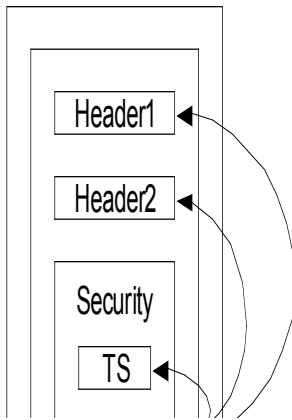
1842
1843 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional
1844 constraints defined by the supporting token assertion. For example, if the supporting token assertion
1845 specifies message parts that need to be encrypted, the specified tokens need to be capable of
1846 encryption.

1847
1848 To illustrate the different ways that supporting tokens may be bound to the message, let’s consider a
1849 message with three components: Header1, Header2, and Body.

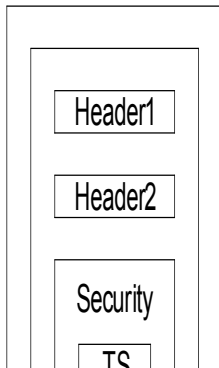


1851
1852 Even before any supporting tokens are added, each binding requires that the message is signed using a
1853 token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts

1854 of the message including the message timestamp (TS) facilitate replay detection. The signature is then
1855 included as part of the Security header as illustrated below:
1856



1857
1858 Note: if required, the initiator may also include in the Security header the token used as the basis for the
1859 message signature (Sig1), not shown in the diagram.
1860 If transport security is used, only the message timestamp (TS) is included in the Security header as
1861 illustrated below:



1862

1863 8.1 Supporting Tokens Assertion

1864 Supporting tokens are included in the security header and may optionally include additional message
1865 parts to sign and/or encrypt.

1866 Syntax

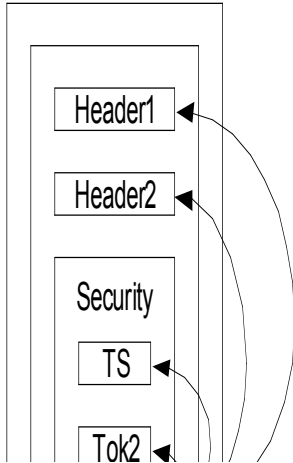
```
1867 <sp:SupportingTokens xmlns:sp="..." ... >  
1868   <wsp:Policy xmlns:wsp="...">  
1869     [Token Assertion]+  
1870     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?  
1871     (  
1872       <sp:SignedParts ... > ... </sp:SignedParts> |  
1873       <sp:SignedElements ... > ... </sp:SignedElements> |  
1874       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |  
1875       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |  
1876     ) *  
1877     ...  
1878   </wsp:Policy>  
1879   ...  
1880 </sp:SupportingTokens>
```

1881

1882 The following describes the attributes and elements listed in the schema outlined above:
1883 /sp:SupportingTokens
1884 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting Tokens]
1885 property.
1886 /sp:SupportingTokens/wsp:Policy
1887 This describes additional requirements for satisfying the SupportingTokens assertion.
1888 /sp:SupportingTokens/wsp:Policy/[Token Assertion]
1889 The policy MUST identify one or more token assertions.
1890 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite
1891 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
1892 describes the algorithms to use for cryptographic operations performed with the tokens identified
1893 by this policy assertion.
1894 /sp:SupportingTokens/wsp:Policy/sp:SignedParts
1895 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
1896 describes additional message parts that MUST be included in the signature generated with the
1897 token identified by this policy assertion.
1898 /sp:SupportingTokens/wsp:Policy/sp:SignedElements
1899 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
1900 describes additional message elements that MUST be included in the signature generated with
1901 the token identified by this policy assertion.
1902 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts
1903 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
1904 describes additional message parts that MUST be encrypted using the token identified by this
1905 policy assertion.
1906 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements
1907 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
1908 describes additional message elements that MUST be encrypted using the token identified by this
1909 policy assertion.

1910 **8.2 SignedSupportingTokens Assertion**

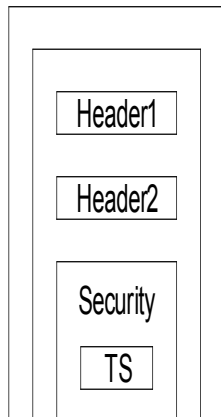
1911 Signed tokens are included in the “message signature” as defined above and may optionally include
1912 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
1913 (Tok2) is signed by the message signature (Sig1):
1914



1915

1916 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

1917



1918

1919 **Syntax**

1920

1921

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

```

<sp:SignedSupportingTokens 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:SignedSupportingTokens>

```

1934

1935 The following describes the attributes and elements listed in the schema outlined above:

1936 /sp:SignedSupportingTokens

1937 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed
1938 Supporting Tokens] property.

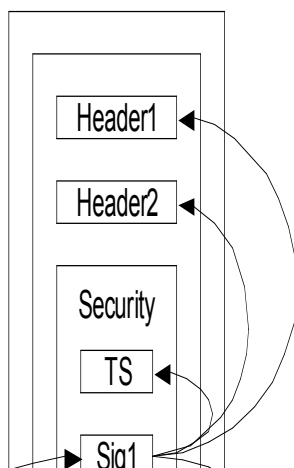
1939 /sp:SignedSupportingTokens/wsp:Policy

1940 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

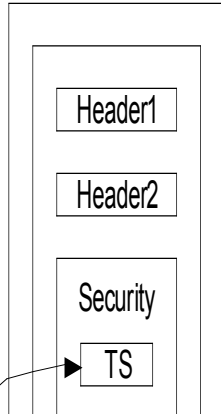
- 1941 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 1942 The policy MUST identify one or more token assertions.
 1943 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 1944 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
 1945 describes the algorithms to use for cryptographic operations performed with the tokens identified
 1946 by this policy assertion.
- 1947 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 1948 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
 1949 describes additional message parts that MUST be included in the signature generated with the
 1950 token identified by this policy assertion.
- 1951 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 1952 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
 1953 describes additional message elements that MUST be included in the signature generated with
 1954 the token identified by this policy assertion.
- 1955 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 1956 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
 1957 describes additional message parts that MUST be encrypted using the token identified by this
 1958 policy assertion.
- 1959 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 1960 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
 1961 describes additional message elements that MUST be encrypted using the token identified by this
 1962 policy assertion.

1963 8.3 EndorsingSupportingTokens Assertion

1964 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 1965 produced from the message signature and may optionally include additional message parts to sign and/or
 1966 encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message signature
 1967 (Sig1):
 1968



1969
 1970 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 1971 below:
 1972



1973

1974 **Syntax**

```

1975 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
1976   <wsp:Policy xmlns:wsp="...">
1977     [Token Assertion]+
1978     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
1979     (
1980       <sp:SignedParts ... > ... </sp:SignedParts> |
1981       <sp:SignedElements ... > ... </sp:SignedElements> |
1982       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
1983       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
1984     ) *
1985     ...
1986   </wsp:Policy>
1987   ...
1988 </sp:EndorsingSupportingTokens>

```

1989

1990 The following describes the attributes and elements listed in the schema outlined above:

1991 /sp:EndorsingSupportingTokens

1992 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
1993 [Endorsing Supporting Tokens] property.

1994 /sp:EndorsingSupportingTokens/wsp:Policy

1995 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

1996 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

1997 The policy MUST identify one or more token assertions.

1998 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

1999 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2000 describes the algorithms to use for cryptographic operations performed with the tokens identified
2001 by this policy assertion.

2002 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2003 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2004 describes additional message parts that MUST be included in the signature generated with the
2005 token identified by this policy assertion.

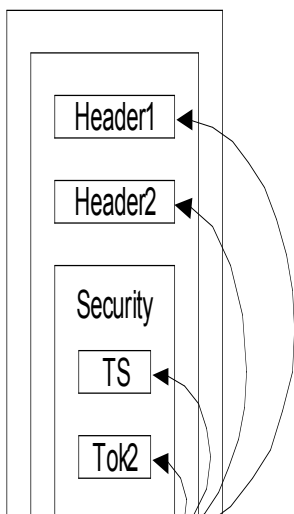
2006 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2007 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
2008 describes additional message elements that MUST be included in the signature generated with
2009 the token identified by this policy assertion.

- 2010 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
- 2011 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
- 2012 describes additional message parts that MUST be encrypted using the token identified by this
- 2013 policy assertion.
- 2014 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
- 2015 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
- 2016 describes additional message elements that MUST be encrypted using the token identified by this
- 2017 policy assertion.

2018 **8.4 SignedEndorsingSupportingTokens Assertion**

2019 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
 2020 and are themselves signed by that message signature, that is both tokens (the token used for the
 2021 message signature and the signed endorsing token) sign each other. This assertion may optionally
 2022 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
 2023 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
 2024 message signature (Sig1):
 2025

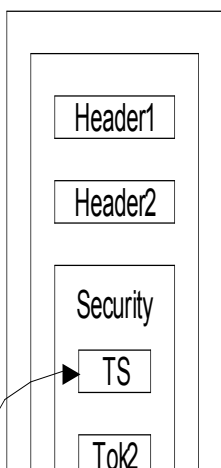


2026

2027 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)

2028 should cover the message timestamp as illustrated below:

2029



2031 Syntax

```
2032 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2033   <wsp:Policy xmlns:wsp="...">
2034     [Token Assertion]+
2035     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2036     (
2037       <sp:SignedParts ... > ... </sp:SignedParts> |
2038       <sp:SignedElements ... > ... </sp:SignedElements> |
2039       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2040       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2041     ) *
2042     ...
2043   </wsp:Policy>
2044   ...
2045 </sp:SignedEndorsingSupportingTokens>
```

2046

2047 The following describes the attributes and elements listed in the schema outlined above:

2048 /sp:SignedEndorsingSupportingTokens

2049 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the
2050 [Signed Endorsing Supporting Tokens] property.

2051 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2052 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2053 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2054 The policy MUST identify one or more token assertions.

2055 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2056 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2057 describes the algorithms to use for cryptographic operations performed with the tokens identified
2058 by this policy assertion.

2059 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2060 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2061 describes additional message parts that MUST be included in the signature generated with the
2062 token identified by this policy assertion.

2063 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2064 This optional element follows the schema outlined in Section 4.1.2 and describes additional
2065 message elements that MUST be included in the signature generated with the token identified by
2066 this policy assertion.

2067 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2068 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2069 describes additional message parts that MUST be encrypted using the token identified by this
2070 policy assertion.

2071 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2072 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2073 describes additional message elements that MUST be encrypted using the token identified by this
2074 policy assertion.

2075 **8.5 SignedEncryptedSupportingTokens Assertion**

2076 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also
2077 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2078 encrypting the supporting tokens.

2079 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of
2080 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2081 sp:SignedSupportingTokens assertion.

2082 **8.6 EndorsingEncryptedSupportingTokens Assertion**

2083 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also
2084 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2085 encrypting the supporting tokens.

2086 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of
2087 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2088 sp:EndorsingSupportingTokens assertion.

2089 **8.7 SignedEndorsingEncryptedSupportingTokens Assertion**

2090 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section
2091 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD
2092 be used for encrypting the supporting tokens.

2093 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of
2094 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per
2095 the sp:SignedEndorsingSupportingTokens assertion.

2096 **8.8 Interaction between [Token Protection] property and supporting 2097 token assertions**

2098 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that
2099 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2100 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or
2101 the [Signature Token] in the Symmetric binding case, covers that token.
- 2102 • Endorsing signatures cover the main signature and the endorsing token.
- 2103 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the
2104 message signature and once by the endorsing signature.

2105 In addition, signed supporting tokens are covered by the message signature, although this is independent
2106 of [Token Protection].

2107 **8.9 Example**

2108 Example policy containing supporting token assertions:

2109

```
<!-- Example Endpoint Policy -->
```

```

2110 <wsp:Policy xmlns:wsp="...">
2111   <sp:SymmetricBinding xmlns:sp="...">
2112     <wsp:Policy>
2113       <sp:ProtectionToken>
2114         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2115           <sp:Issuer>...</sp:Issuer>
2116           <sp:RequestSecurityTokenTemplate>
2117             ...
2118           </sp:RequestSecurityTokenTemplate>
2119         </sp:IssuedToken>
2120       </sp:ProtectionToken>
2121       <sp:AlgorithmSuite>
2122         <wsp:Policy>
2123           <sp:Basic256 />
2124         </wsp:Policy>
2125       </sp:AlgorithmSuite>
2126       ...
2127     </wsp:Policy>
2128   </sp:SymmetricBinding>
2129   ...
2130   <sp:SignedSupportingTokens>
2131     <wsp:Policy>
2132       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2133     </wsp:Policy>
2134   </sp:SignedSupportingTokens>
2135   <sp:SignedEndorsingSupportingTokens>
2136     <wsp:Policy>
2137       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2138         <wsp:Policy>
2139           <sp:WssX509v3Token10 />
2140         </wsp:Policy>
2141       </sp:X509Token>
2142     </wsp:Policy>
2143   </sp:SignedEndorsingSupportingTokens>
2144   ...
2145 </wsp:Policy>

```

2146 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be
2147 included in the security header and covered by the message signature. The
2148 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the
2149 security header and covered by the message signature. In addition, a signature over the message
2150 signature based on the key material associated with the X509 certificate must be included in the security
2151 header.

2152

9 WSS: SOAP Message Security Options

2153
2154
2155
2156

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].

2157
2158
2159

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.

2160
2161

Note: This approach is chosen because:

2162
2163

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

2164
2165

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.

2166

2167
2168

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

2169

2170

WSS: SOAP Message Security 1.0 Properties

2171

[Direct References]

2172
2173
2174

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.

2175

2176

[Key Identifier References]

2177
2178
2179
2180
2181

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'.

2182

2183

[Issuer Serial References]

2184
2185
2186
2187
2188

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'.

2189

2190

[External URI References]

2191
2192
2193

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

2194 generate such references and that the initiator and recipient MAY send a fault if such references are
2195 encountered. This property has a default value of 'false'.

2196 **[Embedded Token References]**

2197 This boolean property indicates whether the initiator and recipient MUST be able to process references
2198 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to
2199 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2200 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2201 This property has a default value of 'false'.

2202

2203 **WSS: SOAP Message Security 1.1 Properties**

2204 **[Thumbprint References]**

2205 This boolean property indicates whether the initiator and recipient MUST be able to process references
2206 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to
2207 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2208 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2209 This property has a default value of 'false'.

2210

2211 **[EncryptedKey References]**

2212 This boolean property indicates whether the initiator and recipient MUST be able to process references
2213 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to
2214 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2215 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2216 This property has a default value of 'false'.

2217

2218 **[Signature Confirmation]**

2219 This boolean property specifies whether `wss11:SignatureConfirmation` elements should be used
2220 as defined in WSS: Soap Message Security 1.1. If the value is 'true',
2221 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If
2222 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property
2223 applies to all signatures that are included in the security header. This property has a default value of
2224 'false'.

2225 **9.1 Wss10 Assertion**

2226 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are
2227 supported.

2228 **Syntax**

```
2229 <sp:Wss10 xmlns:sp="..." ... >  
2230   <wsp:Policy xmlns:wsp="...">  
2231     <sp:MustSupportRefKeyIdentifier ... /> ?  
2232     <sp:MustSupportRefIssuerSerial ... /> ?  
2233     <sp:MustSupportRefExternalURI ... /> ?  
2234     <sp:MustSupportRefEmbeddedToken ... /> ?  
2235     ...  
2236   </wsp:Policy>  
2237   ...  
2238 </sp:Wss10>
```

2239

2240 The following describes the attributes and elements listed in the schema outlined above:

2241 /sp:Wss10
 2242 This identifies a WSS10 assertion.
 2243 /sp:Wss10/wsp:Policy
 2244
 2245 This indicates a policy that controls WSS: SOAP Message Security 1.0
 2246 options./sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2247 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2248 is set to 'true'.
 2249 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial
 2250 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2251 set to 'true'.
 2252 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI
 2253 This optional element is a policy assertion indicates that the [External URI References] property is
 2254 set to 'true'.
 2255 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken
 2256 This optional element is a policy assertion indicates that the [Embedded Token References]
 2257 property is set to 'true'.

2258 9.2 Wss11 Assertion

2259 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
 2260 supported.

2261 Syntax

```

2262 <sp:Wss11 xmlns:sp="..." ... >
2263   <wsp:Policy xmlns:wsp="...">
2264     <sp:MustSupportRefKeyIdentifier ... /> ?
2265     <sp:MustSupportRefIssuerSerial ... /> ?
2266     <sp:MustSupportRefExternalURI ... /> ?
2267     <sp:MustSupportRefEmbeddedToken ... /> ?
2268     <sp:MustSupportRefThumbprint ... /> ?
2269     <sp:MustSupportRefEncryptedKey ... /> ?
2270     <sp:RequireSignatureConfirmation ... /> ?
2271     ...
2272   </wsp:Policy>
2273 </sp:Wss11>
  
```

2274
 2275 The following describes the attributes and elements listed in the schema outlined above:

2276 /sp:Wss11
 2277 This identifies an WSS11 assertion.
 2278 /sp:Wss11/wsp:Policy
 2279 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.
 2280 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2281 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2282 is set to 'true'.
 2283 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial
 2284 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2285 set to 'true'.

- 2286 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2287 This optional element is a policy assertion indicates that the [External URI References] property is
2288 set to 'true'.
- 2289 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2290 This optional element is a policy assertion indicates that the [Embedded Token References]
2291 property is set to 'true'.
- 2292 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2293 This optional element is a policy assertion indicates that the [Thumbprint References] property is
2294 set to 'true'.
- 2295 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2296 This optional element is a policy assertion indicates that the [EncryptedKey References] property
2297 is set to 'true'.
- 2298 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2299 This optional element is a policy assertion indicates that the [Signature Confirmation] property is
2300 set to 'true'.

2301 10 WS-Trust Options

2302 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically
2303 with client and server challenges and entropy behaviors. These assertions relate to interactions with a
2304 Security Token Service and may augment the behaviors defined by the Binding Property Assertions
2305 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2306

2307 **WS-Trust 1.0 Properties**

2308 **[Client Challenge]**

2309 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a
2310 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of
2311 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of
2312 messages exchanged by the client and service in satisfying the RST. This property has a default value of
2313 'false'.

2314

2315 **[Server Challenge]**

2316 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a
2317 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of
2318 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server may
2319 increase the number of messages exchanged by the client and service in order to accommodate the
2320 `wst:SignChallengeResponse` element sent by the client to the server in response to the
2321 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the
2322 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2323

2324 **[Client Entropy]**

2325 This boolean property indicates whether client entropy is required to be used as key material for a
2326 requested proof token. A value of 'true' indicates that client entropy is required. A value of 'false' indicates
2327 that client entropy is not required. This property has a default value of 'false'.

2328

2329 **[Server Entropy]**

2330 This boolean property indicates whether server entropy is required to be used as key material for a
2331 requested proof token. A value of 'true' indicates that server entropy is required. A value of 'false'
2332 indicates that server entropy is not required. This property has a default value of 'false'.

2333 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy
2334 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm
2335 Suite] property.

2336

2337 **[Issued Tokens]**

2338 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in
2339 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'
2340 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of
2341 'false'.

2342 **[Collection]**

2343 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A
2344 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and
2345 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that
2346 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default
2347 value of 'false'.
2348

2349 10.1 Trust10 Assertion

2350 The Trust10 assertion allows you to specify which WS-Trust 1.0 options are supported.

2351 Syntax

```
2352 <sp:Trust10 xmlns:sp="..." ... >  
2353   <wsp:Policy xmlns:wsp="...">  
2354     <sp:MustSupportClientChallenge ... />?  
2355     <sp:MustSupportServerChallenge ... />?  
2356     <sp:RequireClientEntropy ... />?  
2357     <sp:RequireServerEntropy ... />?  
2358     <sp:MustSupportIssuedTokens ... />?  
2359     <sp:RequireRequestSecurityTokenCollection />?  
2360     ...  
2361   </wsp:Policy>  
2362   ...  
2363 </sp:Trust10 ... >
```

2364
2365 The following describes the attributes and elements listed in the schema outlined above:

2366 /sp:Trust10

2367 This identifies a Trust10 assertion.

2368 /sp:Trust10/wsp:Policy

2369 This indicates a policy that controls WS-Trust 1.0 options.

2370 /sp:Trust10/wsp:Policy/sp:MustSupportClientChallenge

2371 This optional element is a policy assertion indicates that the [Client Challenge] property is set to
2372 'true'.

2373 /sp:Trust10/wsp:Policy/sp:MustSupportServerChallenge

2374 This optional element is a policy assertion indicates that the [Server Challenge] property is set to
2375 'true'.

2376 /sp:Trust10/wsp:Policy/sp:RequireClientEntropy

2377 This optional element is a policy assertion indicates that the [Client Entropy] property is set to
2378 'true'.

2379 /sp:Trust10/wsp:Policy/sp:RequireServerEntropy

2380 This optional element is a policy assertion indicates that the [Server Entropy] property is set to
2381 'true'.

2382 /sp:Trust10/wsp:Policy/sp:MustSupportIssuedTokens

2383 This optional element is a policy assertion indicates that the [Issued Tokens] property is set to
2384 'true'.

2385 /sp:Trust10/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2386 This optional element is a policy assertion that indicates that the [Collection] property is set to
2387 'true'.

2388 11 Guidance on creating new assertions and assertion 2389 extensibility

2390 This non-normative appendix provides guidance for designers of new assertions intended for use with this
2391 specification.

2392 11.1 General Design Points

- 2393 • Prefer Distinct QNames
- 2394 • Parameterize using nested policy where possible.
- 2395 • Parameterize using attributes and/or child elements where necessary.

2396 11.2 Detailed Design Guidance

2397 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per
2398 WS-Policy is performed by matching element QNames. Matching does not take into account attributes
2399 that are present on the assertion element. Nor does it take into account child elements except for
2400 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions
2401 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2402
2403 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken
2404 into account. In general, multiple assertions with distinct QNames are preferably to a single assertion that
2405 uses attributes and/or content to distinguish different cases. For example, given two possible assertion
2406 designs;

```
2407  
2408 Design 1  
2409  
2410 <A1/>  
2411 <A2/>  
2412 <A3/>  
2413  
2414 Design 2.  
2415  
2416 <A Parameter='1' />  
2417 <A Parameter='2' />  
2418 <A Parameter='3' />  
2419
```

2420 then design 1. would generally be preferred because it allows the policy matching logic to provide more
2421 accurate matches between policies.

2422
2423 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct
2424 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These
2425 distinct token assertions make policy matching much more useful as less false positives are generated
2426 when performing policy matching.

2427
2428 There are cases where using attributes or child elements as parameters in assertion design is
2429 reasonable. Examples include cases when implementations are expected to understand all the values for
2430 a given parameter and when encoding the parameter information into the assertion QName would result
2431 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2432 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2433 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2434 was encoded into the assertion QNames, each existing token assertion would require five variants, one
2435 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2436

2437 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2438 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2439 assertion is parameterized by the nested token version assertions. Policy matching can use these
2440 parameters to find matches between policies where the broad token type is support by both parties but
2441 they might not support the same specific versions.

2442

2443 Note, when designing assertions for new token types such assertions SHOULD allow the
2444 sp:IncludeToken attribute and SHOULD allow nested policy.

2445

2446 **12 Security Considerations**

2447 It is strongly recommended that policies and assertions be signed to prevent tampering.

2448 It is recommended that policies should not be accepted unless they are signed and have an associated
2449 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely
2450 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that
2451 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2452
2453 It should be noted that the mechanisms described in this document could be secured as part of a SOAP
2454 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using
2455 object-specific security mechanisms.

2456
2457 It is recommended that policies not specify two (or more) SignedSupportingTokens or
2458 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are
2459 subject to modification which may be undetectable.

2460
2461 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest
2462 of the policy in order to combat certain XML substitution attacks.

2463 **A. Assertions and WS-PolicyAttachment**

2464 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per
2465 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical
2466 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any
2467 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy
2468 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

2469 **A.1 Endpoint Policy Subject Assertions**

2470 **A.1.1 Security Binding Assertions**

2471 [TransportBinding Assertion](#) (Section 7.3)
2472 [SymmetricBinding Assertion](#) (Section 7.4)
2473 [AsymmetricBinding Assertion](#) (Section 7.5)

2474 **A.1.2 Token Assertions**

2475 [SupportingTokens Assertion](#) (Section 8.1)
2476 [SignedSupportingTokens Assertion](#) (Section 8.2)
2477 [EndorsingSupportingTokens Assertion](#) (Section 8.3)
2478 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)
2479 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)
2480 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)
2481 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

2482 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

2483 [Wss10 Assertion](#) (Section 9.1)

2484 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

2485 [Wss11 Assertion](#) (Section 9.2)

2486 **A.1.5 Trust 1.0 Assertions**

2487 [Trust10 Assertion](#) (Section 10.1)

2488 **A.2 Operation Policy Subject Assertions**

2489 **A.2.1 Security Binding Assertions**

2490 [SymmetricBinding Assertion](#) (Section 7.4)
2491 [AsymmetricBinding Assertion](#) (Section 7.5)

2492 **A.2.2 Supporting Token Assertions**

2493 [SupportingTokens Assertion](#) (Section 8.1)
2494 [SignedSupportingTokens Assertion](#) (Section 8.2)

2495	EndorsingSupportingTokens Assertion	(Section 8.3)
2496	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2497	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2498	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2499	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2500 **A.3 Message Policy Subject Assertions**

2501 **A.3.1 Supporting Token Assertions**

2502	SupportingTokens Assertion	(Section 8.1)
2503	SignedSupportingTokens Assertion	(Section 8.2)
2504	EndorsingSupportingTokens Assertion	(Section 8.3)
2505	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2506	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2507	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2508	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2509 **A.3.2 Protection Assertions**

2510	SignedParts Assertion	(Section 4.1.1)
2511	SignedElements Assertion	(Section 4.1.2)
2512	EncryptedParts Assertion	(Section 4.2.1)
2513	EncryptedElements Assertion	(Section 4.2.2)
2514	ContentEncryptedElements Assertion	(Section 4.2.3)
2515	RequiredElements Assertion	(Section 4.3.1)
2516	RequiredParts Assertion	(Section 4.3.2)

2517 **A.4 Assertions With Undefined Policy Subject**

2518 The assertions listed in this section do not have a defined policy subject because they appear nested
 2519 inside some other assertion which does have a defined policy subject. This list is derived from nested
 2520 assertions in the specification that have independent sections. It is not a complete list of nested
 2521 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
 2522 additional nested assertions.

2523 **A.4.1 General Assertions**

2524	AlgorithmSuite Assertion	(Section 7.1)
2525	Layout Assertion	(Section 7.2)

2526 **A.4.2 Token Usage Assertions**

2527 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
 2528 assertions.

2529 **A.4.3 Token Assertions**

2530	UsernameToken Assertion	(Section 5.3.1)
------	-----------------------------------------	-----------------

2531	IssuedToken Assertion	(Section 5.3.2)
2532	X509Token Assertion	(Section 5.3.3)
2533	KerberosToken Assertion	(Section 5.3.4)
2534	SpnegoContextToken Assertion	(Section 5.3.5)
2535	SecurityContextToken Assertion	(Section 5.3.6)
2536	SecureConversationToken Assertion	(Section 5.3.7)
2537	SamlToken Assertion	(Section 5.3.8)
2538	RelToken Assertion	(Section 5.3.9)
2539	HttpsToken Assertion	(Section 5.3.10)

2540

B. Issued Token Policy

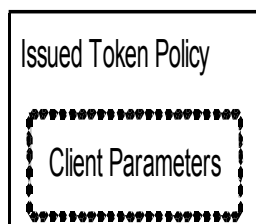
2541 The section provides further detail about behavior associated with the IssuedToken assertion in section
2542 5.3.2.

2543

2544 The issued token security model involves a three-party setup. There's a target Server, a Client, and a
2545 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from
2546 STS to Client. Policy may be embedded inside an Issued Token assertion, or acquired out-of-band. There
2547 may be an explicit trust relationship between the Server and the STS. There must be a trust relationship
2548 between the Client and the STS.

2549

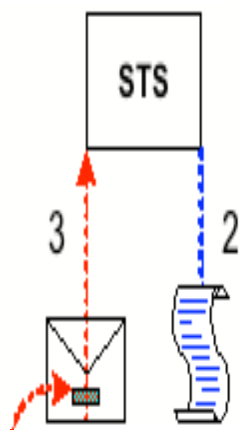
2550 The Issued Token policy assertion includes two parts: 1) client-specific parameters that must be
2551 understood and processed by the client and 2) STS specific parameters which are to be processed by the
2552 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



2553

2554 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server
2555 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are
2556 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the
2557 RST request sent by the Client to the STS as illustrated in the figure below.

2558



2559

2560 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to
2561 formulate the RST request and will include any security-specific requirements of the STS.

2562

2563 The Client may augment or replace the contents of the RST made to the STS based on the Client-specific
2564 parameters received from the Issued Token policy assertion contained in the Server policy, from policy it
2565 received for the STS, or any other local parameters.

2566

2567 The Issued Token Policy Assertion contains elements which must be understood by the Client. The
2568 assertion contains one element which contains a list of arbitrary elements which should be sent along to
2569 the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
2570 request sent by the Client to the STS following the protocol defined in WS-Trust.

2571
2572 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [[WS-](#)
2573 [Trust](#)]. All items are optional, since the Server and STS may already have a pre-arranged relationship
2574 which specifies some or all of the conditions and constraints for issued tokens.

2575

C. Strict Security Header Layout Examples

2576 The following sections describe the security header layout for specific bindings when applying the 'Strict'
2577 layout rules defined in Section 6.7.

2578 C.1 Transport Binding

2579 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

2580 C.1.1 Policy

2581 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport
2582 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username
2583 token attached to the message, and finally an X509 token attached to the message and endorsing the
2584 message signature. No message protection requirements are described since the transport covers all
2585 message parts.

```
2586 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
2587   <sp:TransportBinding>  
2588     <wsp:Policy>  
2589       <sp:TransportToken>  
2590         <wsp:Policy>  
2591           <sp:HttpsToken />  
2592         </wsp:Policy>  
2593       </sp:TransportToken>  
2594       <sp:AlgorithmSuite>  
2595         <wsp:Policy>  
2596           <sp:Basic256 />  
2597         </wsp:Policy>  
2598       </sp:AlgorithmSuite>  
2599       <sp:Layout>  
2600         <wsp:Policy>  
2601           <sp:Strict />  
2602         </wsp:Policy>  
2603       </sp:Layout>  
2604       <sp:IncludeTimestamp />  
2605     </wsp:Policy>  
2606   </sp:TransportBinding>  
2607   <sp:SignedSupportingTokens>  
2608     <wsp:Policy>  
2609       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />  
2610     </wsp:Policy>  
2611   </sp:SignedSupportingTokens>  
2612   <sp:SignedEndorsingSupportingTokens>  
2613     <wsp:Policy>  
2614       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">  
2615         <wsp:Policy>  
2616           <sp:WssX509v3Token10 />  
2617         </wsp:Policy>  
2618       </sp:X509Token>  
2619     </wsp:Policy>  
2620   </sp:SignedEndorsingSupportingTokens>  
2621   <sp:Wss11>  
2622     <sp:RequireSignatureConfirmation />  
2623   </sp:Wss11>  
2624 </wsp:Policy>
```

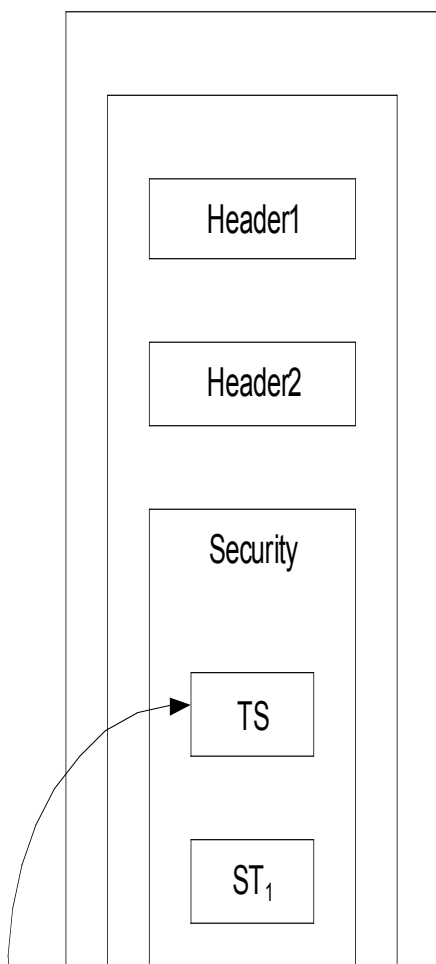
2625 This policy is used as the basis for the examples shown in the subsequent section describing the security
2626 header layout for this binding.

2627 **C.1.2 Initiator to Recipient Messages**

2628 Messages sent from initiator to recipient have the following layout for the security header:

- 2629 1. A `wsu:Timestamp` element.
- 2630 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 2631 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the
2632 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1
2633 above and **SHOULD** cover any other unique identifier for the message in order to prevent
2634 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If
2635 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a
2636 Derived Key Token, based on the supporting token, appears between the supporting token and
2637 the signature.
- 2638 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each
2639 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least
2640 some other unique identifier for the message in order to prevent replays. If [Token Protection] is
2641 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the
2642 supporting token is associated with a symmetric key, then a Derived Key Token, based on the
2643 supporting token, appears before the signature.

2644 The following diagram illustrates the security header layout for the initiator to recipient message:



2645

2646 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The
 2647 arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token labeled ST₂,
 2648 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST₂.
 2649 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering
 2650 of the items in the security header follows the most optimal layout for a receiver to process its contents.

2651 *Example:*

2652 Initiator to recipient message

```

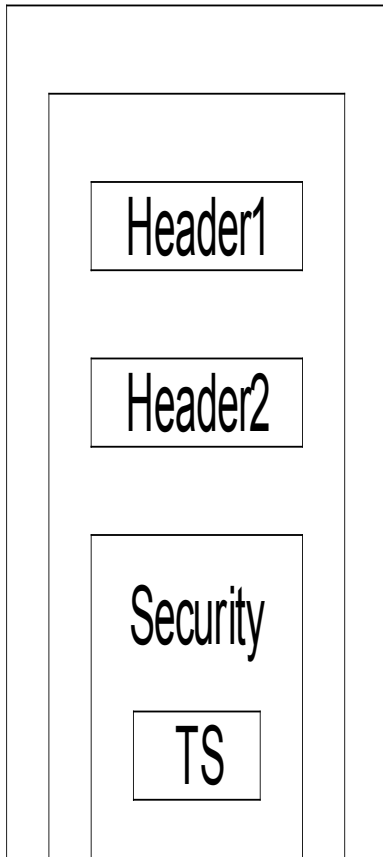
2653 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
2654   <S:Header>
2655     ...
2656     <wsse:Security>
2657       <wsu:Timestamp wsu:Id="timestamp">
2658         <wsu:Created>[datetime]</wsu:Created>
2659         <wsu:Expires>[datetime]</wsu:Expires>
2660       </wsu:Timestamp>
2661       <wsse:UsernameToken wsu:Id='SomeSignedToken' >
2662         ...
2663       </wsse:UsernameToken>
2664       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
2665         ...
2666       </wsse:BinarySecurityToken>
2667       <ds:Signature>
2668         <ds:SignedInfo>
2669           <ds:References>
2670             <ds:Reference URI="#timestamp" />
2671             <ds:Reference URI="#SomeSignedEndorsingToken" />
2672           </ds:References>
2673         </ds:SignedInfo>
2674         <ds:SignatureValue>...</ds:SignatureValue>
2675         <ds:KeyInfo>
2676           <wsse:SecurityTokenReference>
2677             <wsse:Reference URI="#SomeSignedEndorsingToken" />
2678           </wsse:SecurityTokenReference>
2679         </ds:KeyInfo>
2680       </ds:Signature>
2681     ...
2682   </wsse:Security>
2683   ...
2684 </S:Header>
2685 <S:Body>
2686   ...
2687 </S:Body>
2688 </S:Envelope>
  
```

2689 C.1.3 Recipient to Initiator Messages

2690 Messages sent from recipient to initiator have the following layout for the security header:

- 2691 1. A `wsu:Timestamp` element.
- 2692 2. If the [Signature Confirmation] property has a value of 'true', then a
 2693 `wsse11:SignatureConfirmation` element for each signature in the corresponding message
 2694 sent from initiator to recipient. If there are no signatures in the corresponding message from the
 2695 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no `Value`
 2696 attribute.

2697 The following diagram illustrates the security header layout for the recipient to initiator message:



2698

2699 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 2700 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 2701 message illustrated previously is included. In general, the ordering of the items in the security header
 2702 follows the most optimal layout for a receiver to process its contents.

2703 *Example:*

2704 Recipient to initiator message

```

2705 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
2706   <S:Header>
2707     ...
2708     <wsse:Security>
2709       <wsu:Timestamp wsu:Id="timestamp">
2710         <wsu:Created>[datetime]</wsu:Created>
2711         <wsu:Expires>[datetime]</wsu:Expires>
2712       </wsu:Timestamp>
2713       <wsse11:SignatureConfirmation Value="..." />
2714       ...
2715     </wsse:Security>
2716     ...
2717   </S:Header>
2718   <S:Body>
2719     ...
2720   </S:Body>
2721 </S:Envelope>
  
```

2722 C.2 Symmetric Binding

2723 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

2724 C.2.1 Policy

2725 The following example shows a policy indicating a Symmetric Binding, a symmetric key based
2726 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message
2727 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in
2728 the message signature and the supporting signatures, a username token attached to the message, and
2729 finally an X509 token attached to the message and endorsing the message signature. Minimum message
2730 protection requirements are described as well.

```
2731 <!-- Example Endpoint Policy -->
2732 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2733   <sp:SymmetricBinding>
2734     <wsp:Policy>
2735       <sp:ProtectionToken>
2736         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2737           <sp:Issuer>...</sp:Issuer>
2738           <sp:RequestSecurityTokenTemplate>
2739             ...
2740           </sp:RequestSecurityTokenTemplate>
2741         </sp:IssuedToken>
2742       </sp:ProtectionToken>
2743       <sp:AlgorithmSuite>
2744         <wsp:Policy>
2745           <sp:Basic256 />
2746         </wsp:Policy>
2747       </sp:AlgorithmSuite>
2748       <sp:Layout>
2749         <wsp:Policy>
2750           <sp:Strict />
2751         </wsp:Policy>
2752       </sp:Layout>
2753       <sp:IncludeTimestamp />
2754       <sp:EncryptBeforeSigning />
2755       <sp:EncryptSignature />
2756       <sp:ProtectTokens />
2757     </wsp:Policy>
2758   </sp:SymmetricBinding>
2759   <sp:SignedSupportingTokens>
2760     <wsp:Policy>
2761       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2762     </wsp:Policy>
2763   </sp:SignedSupportingTokens>
2764   <sp:SignedEndorsingSupportingTokens>
2765     <wsp:Policy>
2766       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
2767         <wsp:Policy>
2768           <sp:WssX509v3Token10 />
2769         </wsp:Policy>
2770       </sp:X509Token>
2771     </wsp:Policy>
2772   </sp:SignedEndorsingSupportingTokens>
2773   <sp:Wss11>
2774     <wsp:Policy>
2775       <sp:RequireSignatureConfirmation />
2776     </wsp:Policy>
2777   </sp:Wss11>
2778 </wsp:Policy>
2779
```



```

2780
2781 <!-- Example Message Policy -->
2782 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2783   <sp:SignedParts>
2784     <sp:Header Name="Header1" Namespace="..." />
2785     <sp:Header Name="Header2" Namespace="..." />
2786     <sp:Body/>
2787   </sp:SignedParts>
2788   <sp:EncryptedParts>
2789     <sp:Header Name="Header2" Namespace="..." />
2790     <sp:Body/>
2791   </sp:EncryptedParts>
2792 </wsp:Policy>

```

2793 This policy is used as the basis for the examples shown in the subsequent section describing the security
2794 header layout for this binding.

2795 C.2.2 Initiator to Recipient Messages

2796 Messages sent from initiator to recipient have the following layout for the security header:

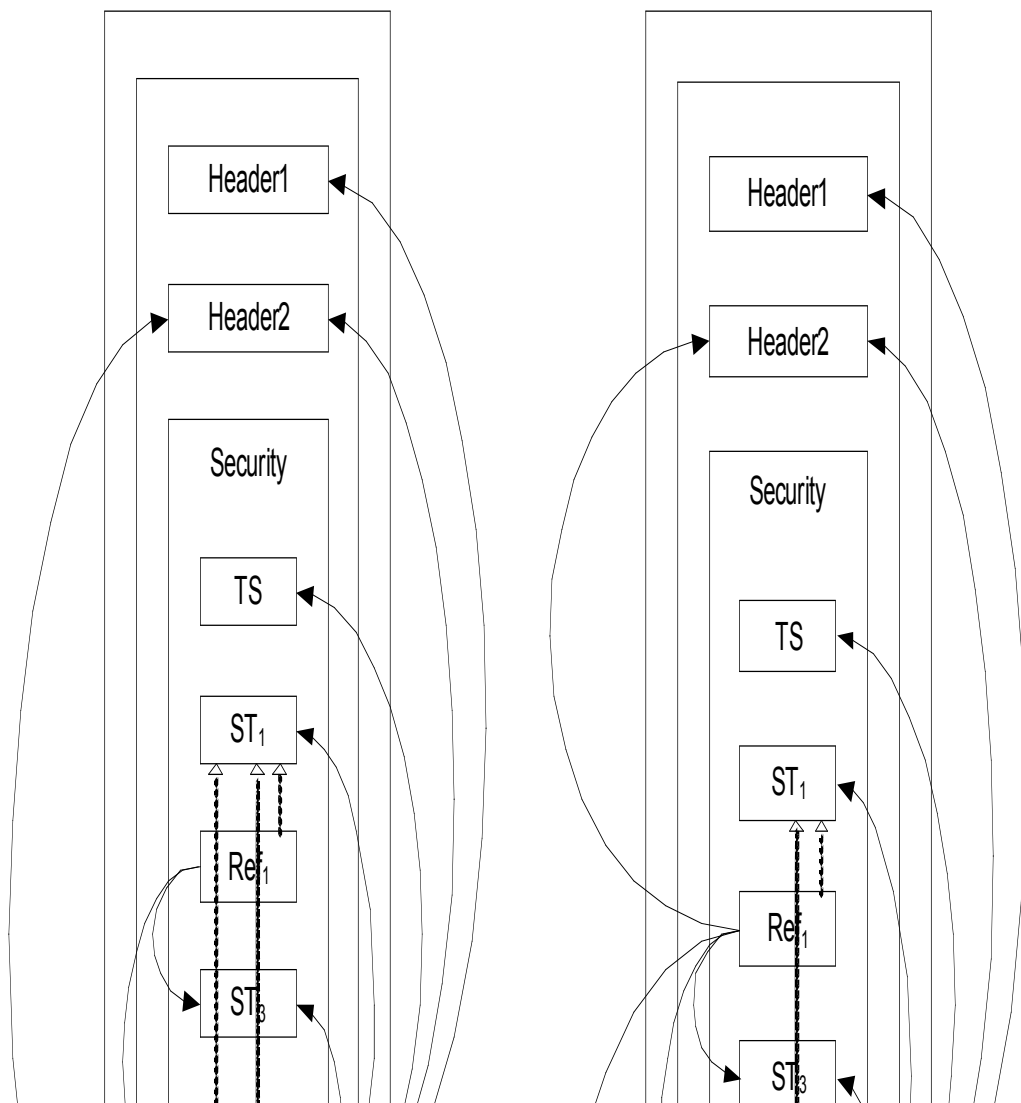
- 2797 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 2798 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
2799 `.../IncludeToken/Always`, then the [Encryption Token].
- 2800 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
2801 Derived Key Token is used for encryption.
- 2802 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
2803 reference list MUST include a reference to the message signature. If [Protection Order] is
2804 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
2805 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
2806 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 2807 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
2808 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
2809 `.../IncludeToken/Always`.
- 2810 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
2811 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
2812 [Signature Token].
- 2813 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
2814 Derived Key Token is used for signature.
- 2815 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
2816 whether they are included in the message, and any message parts specified in SignedParts
2817 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
2818 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
2819 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 2820 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
2821 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
2822 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
2823 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
2824 endorsing token, appears before the signature.
- 2825 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
2826 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
2827 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
2828 above.

2829

2830 The following diagram illustrates the security header layout for the initiator to recipient message:

Encrypt Then Sign

Sign Then Encrypt



2831

2832 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
2833 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
2834 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
2835 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
2836 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
2837 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
2838 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
2839 process its contents.

2840 *Example:*

2841 Initiator to recipient message using EncryptBeforeSigning:

```
2842 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
2843   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
2844   xmlns:xenc="..." xmlns:ds="...">
2845   <S:Header>
2846     <x:Header1 wsu:Id="Header1" >
2847       ...
2848     </x:Header1>
2849
```

```

2850 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
2851   <!-- Plaintext Header2
2852   <x:Header2 wsu:Id="Header2" >
2853     ...
2854   </x:Header2>
2855   -->
2856   ...
2857 </wsse1:EncryptedHeader>
2858 ...
2859 <wsse:Security>
2860   <wsu:Timestamp wsu:Id="Timestamp">
2861     <wsu:Created>...</wsu:Created>
2862     <wsu:Expires>...</wsu:Expires>
2863   </wsu:Timestamp>
2864   <saml:Assertion AssertionId="_SharedSecretToken" ...>
2865     ...
2866   </saml:Assertion>
2867   <xenc:ReferenceList>
2868     <xenc:DataReference URI="#enc_Signature" />
2869     <xenc:DataReference URI="#enc_SomeUsernameToken" />
2870     ...
2871   </xenc:ReferenceList>
2872   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
2873     <!-- Plaintext UsernameToken
2874     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
2875       ...
2876     </wsse:UsernameToken>
2877     -->
2878     ...
2879     <ds:KeyInfo>
2880       <wsse:SecurityTokenReference>
2881         <wsse:Reference URI="#_SharedSecretToken" />
2882       </wsse:SecurityTokenReference>
2883     </ds:KeyInfo>
2884   </xenc:EncryptedData>
2885   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
2886     ...
2887   </wsse:BinarySecurityToken>
2888   <xenc:EncryptedData ID="enc_Signature">
2889     <!-- Plaintext Signature
2890     <ds:Signature Id="Signature">
2891       <ds:SignedInfo>
2892         <ds:References>
2893           <ds:Reference URI="#Timestamp" >...</ds:Reference>
2894           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
2895           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
2896           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
2897           <ds:Reference URI="#Header1" >...</ds:Reference>
2898           <ds:Reference URI="#Header2" >...</ds:Reference>
2899           <ds:Reference URI="#Body" >...</ds:Reference>
2900         </ds:References>
2901       </ds:SignedInfo>
2902     </ds:SignatureValue>...</ds:SignatureValue>
2903     <ds:KeyInfo>
2904       <wsse:SecurityTokenReference>
2905         <wsse:Reference URI="#_SharedSecretToken" />
2906       </wsse:SecurityTokenReference>
2907     </ds:KeyInfo>
2908   </xenc:EncryptedData>
2909   -->
2910   ...
2911   <ds:KeyInfo>
2912     <wsse:SecurityTokenReference>
2913       <wsse:Reference URI="#_SharedSecretToken" />

```

```

2914     </wsse:SecurityTokenReference>
2915     </ds:KeyInfo>
2916   </xenc:EncryptedData>
2917   <ds:Signature>
2918     <ds:SignedInfo>
2919       <ds:References>
2920         <ds:Reference URI="#Signature" >...</ds:Reference>
2921         <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
2922       </ds:References>
2923     </ds:SignedInfo>
2924     <ds:SignatureValue>...</ds:SignatureValue>
2925     <ds:KeyInfo>
2926       <wsse:SecurityTokenReference>
2927         <wsse:Reference URI="#SomeSupportingToken" />
2928       </wsse:SecurityTokenReference>
2929     </ds:KeyInfo>
2930   </ds:Signature>
2931 <xenc:ReferenceList>
2932   <xenc:DataReference URI="#enc_Body" />
2933   <xenc:DataReference URI="#enc_Header2" />
2934   ...
2935 </xenc:ReferenceList>
2936 </wsse:Security>
2937 </S:Header>
2938 <S:Body wsu:Id="Body">
2939   <xenc:EncryptedData Id="enc_Body">
2940     ...
2941     <ds:KeyInfo>
2942       <wsse:SecurityTokenReference>
2943         <wsse:Reference URI="#_SharedSecretToken" />
2944       </wsse:SecurityTokenReference>
2945     </ds:KeyInfo>
2946   </xenc:EncryptedData>
2947 </S:Body>
2948 </S:Envelope>

```

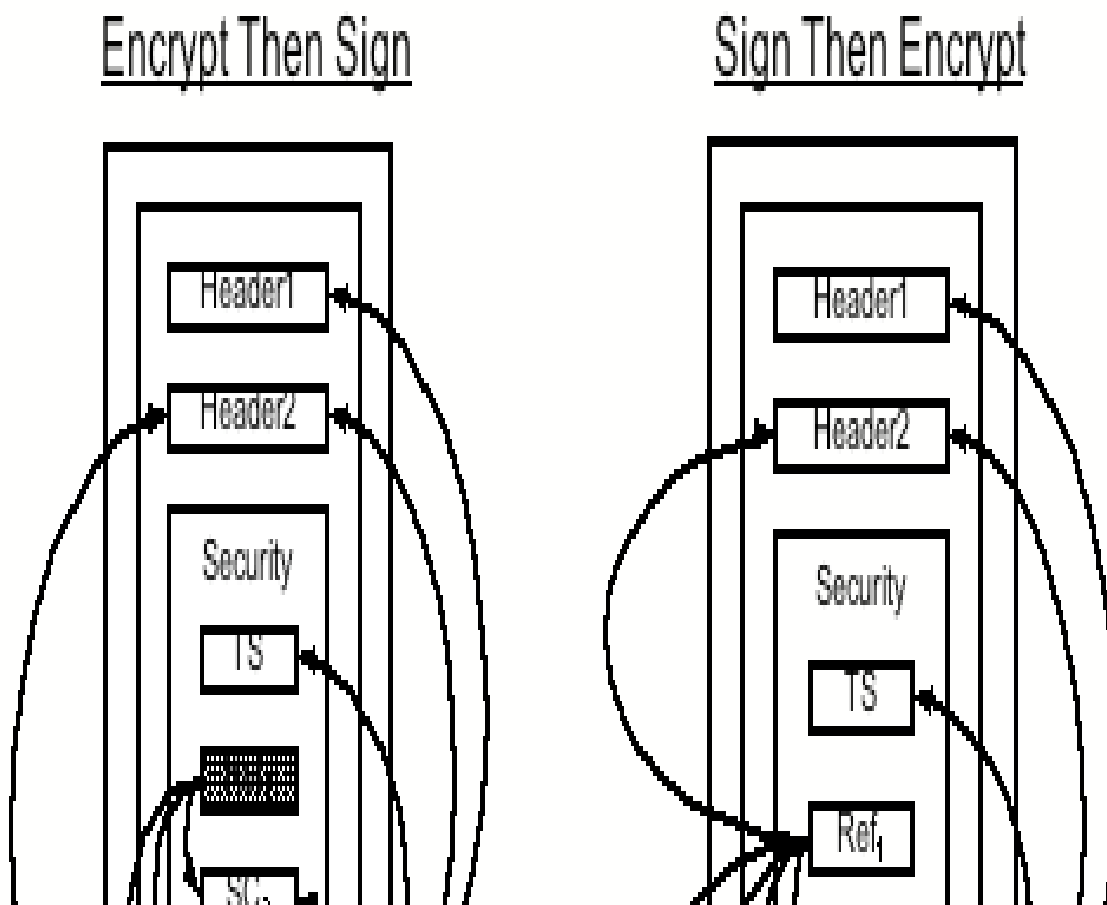
2949 C.2.3 Recipient to Initiator Messages

2950 Messages send from recipient to initiator have the following layout for the security header:

- 2951 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 2952 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the
2953 [Encryption Token].
- 2954 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
2955 Derived Key Token is used for encryption.
- 2956 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
2957 reference list MUST include a reference to the message signature from 6 below, and the
2958 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is
2959 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
2960 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
2961 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2
2962 above.
- 2963 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each
2964 signature in the corresponding message sent from initiator to recipient. If there are no signatures
2965 in the corresponding message from the initiator to the recipient, then a
2966 `wss11:SignatureConfirmation` element with no Value attribute.
- 2967 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
2968 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].

- 2969 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This
 2970 Derived Key Token is used for signature.
- 2971 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation
 2972 elements from 5 above, and all the message parts specified in SignedParts assertions in the
 2973 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]
 2974 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token
 2975 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 2976 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message
 2977 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
 2978 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption
 2979 Token].

2980 The following diagram illustrates the security header layout for the recipient to initiator message:



2981

2982 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 2983 The arrows on the left from boxes labeled Ref₁ indicate references to parts encrypted using a key based
 2984 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two
 2985 wssell:SignatureConfirmation elements labeled SC₁ and SC₂ corresponding to the two signatures
 2986 in the initial message illustrated previously is included. In general, the ordering of the items in the security
 2987 header follows the most optimal layout for a receiver to process its contents. The rules used to determine
 2988 this ordering are described in Appendix C.

2989 *Example:*

2990 Recipient to initiator message using EncryptBeforeSigning:

```
2991 <S:Envelope>
2992   <S:Header>
2993     <x:Header1 wsu:Id="Header1" >
2994       ...
2995     </x:Header1>
2996     <wsse1:EncryptedHeader wsu:Id="enc_Header2">
2997       <!-- Plaintext Header2
2998       <x:Header2 wsu:Id="Header2" >
2999         ...
3000       </x:Header2>
3001       -->
3002       ...
3003     </wsse1:EncryptedHeader>
3004     ...
3005     <wsse:Security>
3006       <wsu:Timestamp wsu:Id="Timestamp">
3007         <wsu:Created>...</wsu:Created>
3008         <wsu:Expires>...</wsu:Expires>
3009       </wsu:Timestamp>
3010       <xenc:ReferenceList>
3011         <xenc:DataReference URI="#enc_Signature" />
3012         <xenc:DataReference URI="#enc_SigConf1" />
3013         <xenc:DataReference URI="#enc_SigConf2" />
3014         ...
3015       </xenc:ReferenceList>
3016       <xenc:EncryptedData ID="enc_SigConf1" >
3017         <!-- Plaintext SignatureConfirmation
3018         <wsse1:SignatureConfirmation wsu:Id="SigConf1" >
3019           ...
3020         </wsse1:SignatureConfirmation>
3021         -->
3022         ...
3023       </xenc:EncryptedData>
3024       <xenc:EncryptedData ID="enc_SigConf2" >
3025         <!-- Plaintext SignatureConfirmation
3026         <wsse1:SignatureConfirmation wsu:Id="SigConf2" >
3027           ...
3028         </wsse1:SignatureConfirmation>
3029         -->
3030         ...
3031       </xenc:EncryptedData>
```

```

3032
3033
3034 <xenc:EncryptedData Id="enc_Signature">
3035   <!-- Plaintext Signature
3036   <ds:Signature Id="Signature">
3037     <ds:SignedInfo>
3038       <ds:References>
3039         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3040         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3041         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3042         <ds:Reference URI="#Header1" >...</ds:Reference>
3043         <ds:Reference URI="#Header2" >...</ds:Reference>
3044         <ds:Reference URI="#Body" >...</ds:Reference>
3045       </ds:References>
3046     </ds:SignedInfo>
3047     <ds:SignatureValue>...</ds:SignatureValue>
3048     <ds:KeyInfo>
3049       <wsse:SecurityTokenReference>
3050         <wsse:Reference URI="#_SomeIssuedToken" />
3051       </wsse:SecurityTokenReference>
3052     </ds:KeyInfo>
3053   </ds:Signature>
3054   -->
3055 </xenc:EncryptedData>
3056   ...
3057   <ds:KeyInfo>
3058     <wsse:SecurityTokenReference>
3059       <wsse:Reference URI="#_SomeIssuedToken" />
3060     </wsse:SecurityTokenReference>
3061   </ds:KeyInfo>
3062 <xenc:EncryptedData>
3063 <xenc:ReferenceList>
3064   <xenc:DataReference URI="#enc_Body" />
3065   <xenc:DataReference URI="#enc_Header2" />
3066   ...
3067 </xenc:ReferenceList>
3068 </xenc:EncryptedData>
3069 </wsse:Security>
3070 </S:Header>
3071 <S:Body wsu:Id="Body">
3072   <xenc:EncryptedData Id="enc_Body">
3073     ...
3074     <ds:KeyInfo>
3075       <wsse:SecurityTokenReference>
3076         <wsse:Reference URI="#_SomeIssuedToken" />
3077       </wsse:SecurityTokenReference>
3078     </ds:KeyInfo>
3079   </xenc:EncryptedData>
3080 </S:Body>
</S:Envelope>

```

3081 C.3 Asymmetric Binding

3082 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

3083 C.3.1 Policy

3084 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator
3085 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the
3086 message parts before signing, a requirement to encrypt the message signature, a requirement to include
3087 tokens in the message signature and the supporting signatures, a requirement to include
3088 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3089 an X509 token attached to the message and endorsing the message signature. Minimum message
3090 protection requirements are described as well.

```
3091 <!-- Example Endpoint Policy -->
3092 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3093   <sp:AsymmetricBinding>
3094     <wsp:Policy>
3095       <sp:RecipientToken>
3096         <wsp:Policy>
3097           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3098         </wsp:Policy>
3099       </sp:RecipientToken>
3100     <sp:InitiatorToken>
3101       <wsp:Policy>
3102         <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3103       </wsp:Policy>
3104     </sp:InitiatorToken>
3105     <sp:AlgorithmSuite>
3106       <wsp:Policy>
3107         <sp:Basic256 />
3108       </wsp:Policy>
3109     </sp:AlgorithmSuite>
3110     <sp:Layout>
3111       <wsp:Policy>
3112         <sp:Strict />
3113       </wsp:Policy>
3114     </sp:Layout>
3115     <sp:IncludeTimestamp />
3116     <sp:EncryptBeforeSigning />
3117     <sp:EncryptSignature />
3118     <sp:ProtectTokens />
3119   </wsp:Policy>
3120 </sp:AsymmetricBinding>
3121 <sp:SignedEncryptedSupportingTokens>
3122   <wsp:Policy>
3123     <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3124   </wsp:Policy>
3125 </sp:SignedEncryptedSupportingTokens>
3126 <sp:SignedEndorsingSupportingTokens>
3127   <wsp:Policy>
3128     <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3129       <wsp:Policy>
3130         <sp:WssX509v3Token10 />
3131       </wsp:Policy>
3132     </sp:X509Token>
3133   </wsp:Policy>
3134 </sp:SignedEndorsingSupportingTokens>
3135 <sp:Wss11>
3136   <wsp:Policy>
3137     <sp:RequireSignatureConfirmation />
3138   </wsp:Policy>
3139 </sp:Wss11>
3140 </wsp:Policy>
3141
```

3142

```

3143 <!-- Example Message Policy -->
3144 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3145   <sp:SignedParts>
3146     <sp:Header Name="Header1" Namespace="..." />
3147     <sp:Header Name="Header2" Namespace="..." />
3148     <sp:Body/>
3149   </sp:SignedParts>
3150   <sp:EncryptedParts>
3151     <sp:Header Name="Header2" Namespace="..." />
3152     <sp:Body/>
3153   </sp:EncryptedParts>
3154 </wsp:All>

```

3155
3156 This policy is used as the basis for the examples shown in the subsequent section describing the security
3157 header layout for this binding.

3158 **C.3.2 Initiator to Recipient Messages**

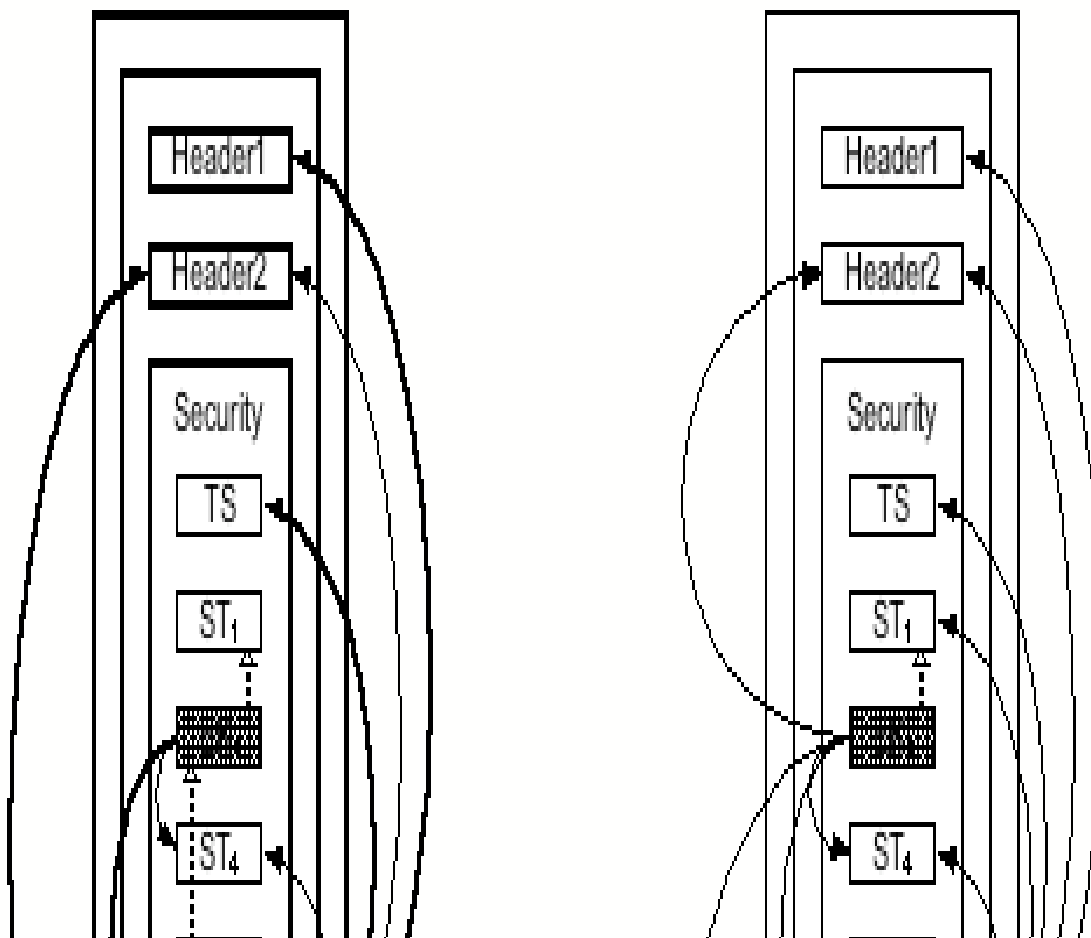
3159 Messages sent from initiator to recipient have the following layout:

- 3160 1. A `wsu:Timestamp` element if `[Timestamp]` is 'true'.
- 3161 2. If a `[Recipient Token]` is specified, and the associated `sp:IncludeToken` attribute is
3162 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Recipient Token]`.
- 3163 3. If a `[Recipient Token]` is specified and `[Protection Order]` is 'SignBeforeEncrypting' or
3164 `[SignatureProtection]` is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3165 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3166 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If
3167 `[Signature Protection]` is 'true' then the reference list MUST contain a reference to the message
3168 signature from 6 below. It is an error if `[Signature Protection]` is 'true' and there is not a message
3169 signature.
- 3170 4. Any tokens from the supporting tokens properties (as defined in section 8) whose
3171 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3172 5. If an `[Initiator Token]` is specified, and the associated `sp:IncludeToken` attribute is
3173 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Initiator Token]`.
- 3174 6. A signature based on the key in the `[Initiator Token]` if specified, over the `wsu:Timestamp` from
3175 1 above, any tokens from 4 above regardless of whether they are included in the message, and
3176 any message parts specified in `SignedParts` assertions in the policy. If `[Token Protection]` is 'true',
3177 the signature MUST also cover the `[Initiator Token]` regardless of whether it is included in the
3178 message.
- 3179 7. Signatures for tokens from the `[Endorsing Supporting Tokens]` and `[Signed Endorsing Supporting`
3180 `Tokens]` properties. If `[Derived Keys]` is 'true' and the supporting token is associated with a
3181 symmetric key, then a `Derived Key Token`, based on the supporting token, appears before the
3182 signature. If `[Token Protection]` is 'true', the signature MUST also cover the supporting token
3183 regardless of whether it is included in the message.
- 3184 8. If a `[Recipient Token]` is specified and `[Protection Order]` is 'EncryptBeforeSigning' then if
3185 `[Signature Protection]` is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
3186 for the recipient and a reference list, else if `[Signature Protection]` is 'true', a reference list. The
3187 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions
3188 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
3189 element from 3 above.

3191 The following diagram illustrates the security header layout for the initiator to recipient messages:

Encrypt Then Sign

Sign Then Encrypt



3192
3193 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
3194 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate
3195 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as
3196 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references
3197 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left
3198 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the
3199 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as
3200 the basis for each cryptographic operation. In general, the ordering of the items in the security header
3201 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
3202 ordering are described in Appendix C.

3203
3204 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
3205 key contains an external reference to the token containing the encryption key. The diagram illustrates
3206 how one might attach a security token related to the encrypted key for completeness. One possible use-

3207 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3208 wishes to include the encryption token in the message signature.

3209 Initiator to recipient message *Example*

3210 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3211     xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3212 <S:Header>
3213   <x:Header1 wsu:Id="Header1" >
3214     ...
3215   </x:Header1>
3216   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3217     <!-- Plaintext Header2
3218     <x:Header2 wsu:Id="Header2" >
3219       ...
3220     </x:Header2>
3221     -->
3222     ...
3223   </wssell1:EncryptedHeader>
3224   ...
3225   <wsse:Security>
3226     <wsu:Timestamp wsu:Id="Timestamp">
3227       <wsu:Created>...</wsu:Created>
3228       <wsu:Expires>...</wsu:Expires>
3229     </wsu:Timestamp>
3230     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3231       ...
3232     </wsse:BinarySecurityToken>
3233     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3234       ...
3235       <xenc:ReferenceList>
3236         <xenc:DataReference URI="#enc_Signature" />
3237         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3238         ...
3239       </xenc:ReferenceList>
3240     </xenc:EncryptedKey>
3241     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3242       <!-- Plaintext UsernameToken
3243       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3244         ...
3245       </wsse:UsernameToken>
3246       -->
3247       ...
3248     </xenc:EncryptedData>
3249     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3250       ...
3251     </wsse:BinarySecurityToken>
3252     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3253       ...
3254     </wsse:BinarySecurityToken>
3255     <xenc:EncryptedData ID="enc_Signature">
3256       <!-- Plaintext Signature
3257       <ds:Signature Id="Signature">
3258         <ds:SignedInfo>
3259           <ds:References>
3260             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3261             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3262             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3263             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3264             <ds:Reference URI="#Header1" >...</ds:Reference>
3265             <ds:Reference URI="#Header2" >...</ds:Reference>
3266             <ds:Reference URI="#Body" >...</ds:Reference>
3267           </ds:References>
3268         </ds:SignedInfo>
3269         <ds:SignatureValue>...</ds:SignatureValue>
3270       </ds:Signature>
3271       <ds:KeyInfo>
3272         <wsse:SecurityTokenReference>
3273           <wsse:Reference URI="#InitiatorToken" />
3274         </wsse:SecurityTokenReference>
3275       </ds:KeyInfo>

```

```

3275     </ds:Signature>
3276     -->
3277     ...
3278 </xenc:EncryptedData>
3279 <ds:Signature>
3280   <ds:SignedInfo>
3281     <ds:References>
3282       <ds:Reference URI="#Signature" >...</ds:Reference>
3283       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3284     </ds:References>
3285   </ds:SignedInfo>
3286   <ds:SignatureValue>...</ds:SignatureValue>
3287   <ds:KeyInfo>
3288     <wsse:SecurityTokenReference>
3289       <wsse:Reference URI="#SomeSupportingToken" />
3290     </wsse:SecurityTokenReference>
3291   </ds:KeyInfo>
3292 </ds:Signature>
3293 <xenc:ReferenceList>
3294   <xenc:DataReference URI="#enc_Body" />
3295   <xenc:DataReference URI="#enc_Header2" />
3296   ...
3297 </xenc:ReferenceList>
3298 </wsse:Security>
3299 </S:Header>
3300 <S:Body wsu:Id="Body">
3301   <xenc:EncryptedData Id="enc_Body">
3302     ...
3303     <ds:KeyInfo>
3304       <wsse:SecurityTokenReference>
3305         <wsse:Reference URI="#RecipientEncryptedKey" />
3306       </wsse:SecurityTokenReference>
3307     </ds:KeyInfo>
3308   </xenc:EncryptedData>
3309 </S:Body>
3310 </S:Envelope>

```

3311 C.3.3 Recipient to Initiator Messages

3312 Messages sent from recipient to initiator have the following layout:

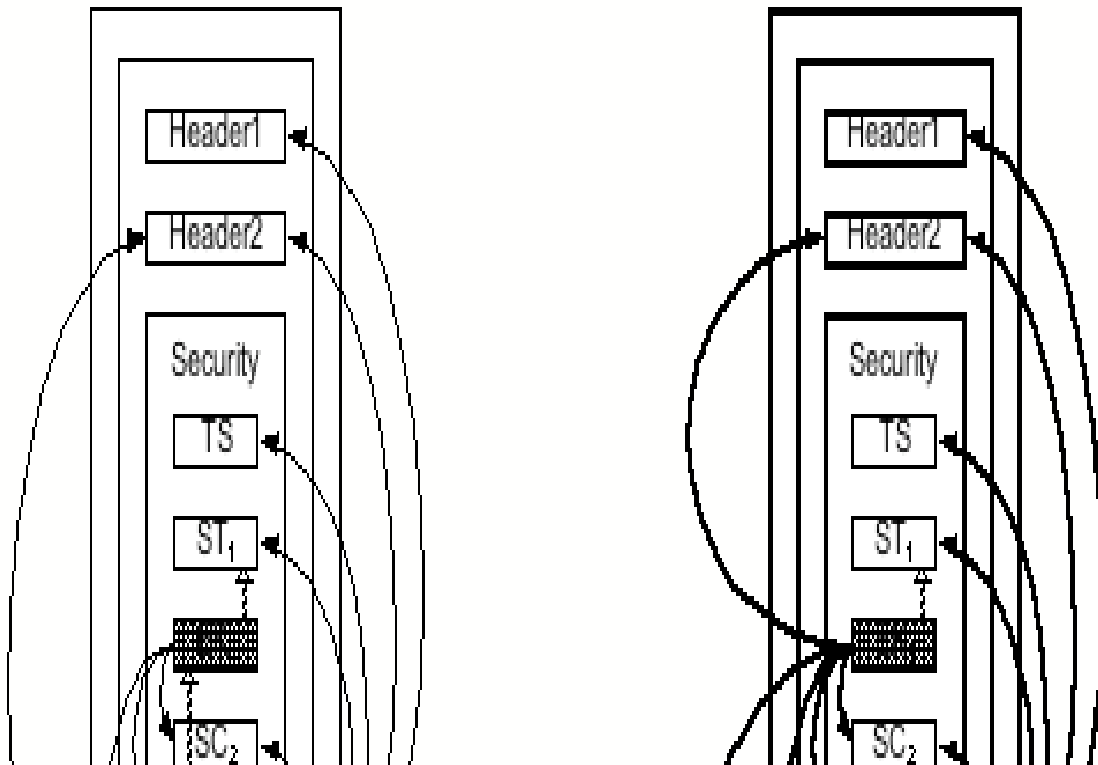
- 3313 1. A `wsu:Timestamp` element if `[Timestamp]` is 'true'.
- 3314 2. If an `[Initiator Token]` is specified, and the associated `sp:IncludeToken` attribute is
3315 `.../IncludeToken/Always`, then the `[Initiator Token]`.
- 3316 3. If an `[Initiator Token]` is specified and `[Protection Order]` is 'SignBeforeEncrypting' or
3317 `[SignatureProtection]` is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3318 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3319 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If
3320 `[Signature Protection]` is 'true' then the reference list MUST also contain a reference to the
3321 message signature from 6 below, if any and references to the
3322 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3323 4. If `[Signature Confirmation]` is 'true', then a `wss11:SignatureConfirmation` element for each
3324 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3325 in the corresponding message from the initiator to the recipient, then a
3326 `wss11:SignatureConfirmation` element with no `Value` attribute.
- 3327 5. If a `[Recipient Token]` is specified, and the associated `sp:IncludeToken` attribute is
3328 `.../IncludeToken/Always`, then the `[Recipient Token]`.

- 3329 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],
 3330 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements
 3331 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token
 3332 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3333 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
 3334 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
 3335 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
 3336 reference list includes a reference to all the message parts specified in EncryptedParts assertions
 3337 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
 3338 element from 3 above.

3339
 3340 The following diagram illustrates the security header layout for the recipient to initiator messages:

Encrypt Then Sign

Sign Then Encrypt



3341
 3342 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
 3343 using the [Recipient Token] labeled ST₂. The arrows on the left from boxes labeled EK₁ indicate
 3344 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on
 3345 the left from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained
 3346 in the encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token
 3347 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements
 3348 labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is
 3349 included. In general, the ordering of the items in the security header follows the most optimal layout for a
 3350 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.
 3351 Recipient to initiator message *Example*:

```

3352 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3353     xmlns:wssell="..." xmlns:wsse="..."
3354     xmlns:xenc="..." xmlns:ds="...">
3355 <S:Header>
3356   <x:Header1 wsu:Id="Header1" >
3357     ...
3358   </x:Header1>
3359   <wssell:EncryptedHeader wsu:Id="enc_Header2">
3360     <!-- Plaintext Header2
3361     <x:Header2 wsu:Id="Header2" >
3362       ...
3363     </x:Header2>
3364     -->
3365     ...
3366   </wssell:EncryptedHeader>
3367   ...
3368   <wsse:Security>
3369     <wsu:Timestamp wsu:Id="Timestamp">
3370       <wsu:Created>...</wsu:Created>
3371       <wsu:Expires>...</wsu:Expires>
3372     </wsu:Timestamp>
3373     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3374       ...
3375     </wsse:BinarySecurityToken>
3376     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3377       ...
3378       <xenc:ReferenceList>
3379         <xenc:DataReference URI="#enc_Signature" />
3380         <xenc:DataReference URI="#enc_SigConf1" />
3381         <xenc:DataReference URI="#enc_SigConf2" />
3382         ...
3383       </xenc:ReferenceList>
3384     </xenc:EncryptedKey>
3385     <xenc:EncryptedData ID="enc_SigConf2" >
3386       <!-- Plaintext SignatureConfirmation
3387       <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3388         ...
3389       </wssell:SignatureConfirmation>
3390       -->
3391       ...
3392     </xenc:EncryptedData>
3393     <xenc:EncryptedData ID="enc_SigConf1" >
3394       <!-- Plaintext SignatureConfirmation
3395       <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3396         ...
3397       </wssell:SignatureConfirmation>
3398       -->
3399       ...
3400     </xenc:EncryptedData>
3401     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3402       ...
3403     </wsse:BinarySecurityToken>
3404

```



```

3405 <xenc:EncryptedData ID="enc_Signature">
3406   <!-- Plaintext Signature
3407   <ds:Signature Id="Signature">
3408     <ds:SignedInfo>
3409       <ds:References>
3410         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3411         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3412         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3413         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3414         <ds:Reference URI="#Header1" >...</ds:Reference>
3415         <ds:Reference URI="#Header2" >...</ds:Reference>
3416         <ds:Reference URI="#Body" >...</ds:Reference>
3417       </ds:References>
3418     </ds:SignedInfo>
3419     <ds:SignatureValue>...</ds:SignatureValue>
3420     <ds:KeyInfo>
3421       <wsse:SecurityTokenReference>
3422         <wsse:Reference URI="#RecipientToken" />
3423       </wsse:SecurityTokenReference>
3424     </ds:KeyInfo>
3425   </ds:Signature>
3426   -->
3427   ...
3428 </xenc:EncryptedData>
3429 <xenc:ReferenceList>
3430   <xenc:DataReference URI="#enc_Body" />
3431   <xenc:DataReference URI="#enc_Header2" />
3432   ...
3433 </xenc:ReferenceList>
3434 </wsse:Security>
3435 </S:Header>
3436 <S:Body wsu:Id="Body">
3437   <xenc:EncryptedData Id="enc_Body">
3438     ...
3439     <ds:KeyInfo>
3440       <wsse:SecurityTokenReference>
3441         <wsse:Reference URI="#InitiatorEncryptedKey" />
3442       </wsse:SecurityTokenReference>
3443     </ds:KeyInfo>
3444   </xenc:EncryptedData>
3445 </S:Body>
3446 </S:Envelope>

```

3447 **D. Signed and Encrypted Elements in the Security**
3448 **Header**

3449 This section lists the criteria for when various child elements of the Security header are signed and/or
3450 encrypted at the message level including whether they are signed by the message signature or a
3451 supporting signature. It assumes that there are no `sp:SignedElements` and no
3452 `sp:EncryptedElements` assertions in the policy. If such assertions are present in the policy then
3453 additional child elements of the security header might be signed and/or encrypted.

3454 **D.1 Elements signed by the message signature**

- 3455 1. The `wsu:Timestamp` element (Section 6.2).
- 3456 2. All `wssell:SignatureConfirmation` elements (Section 9).
- 3457 3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token],
3458 [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption
3459 Token] when [Token Protection] has a value of 'true' (Section 6.5).
- 3460 4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed
3461 Endorsing Supporting Tokens] (Section 8.5).

3462 **D.2 Elements signed by all endorsing signatures**

- 3463 1. The `ds:Signature` element that forms the message signature (Section 8.3).
- 3464 2. The `wsu:Timestamp` element in the case of a transport binding (Section 8.3).

3465 **D.3 Elements signed by a specific endorsing signature**

- 3466 1. Security Tokens corresponding to [Endorsing Supporting Tokens] or [Signed Endorsing
3467 Supporting Tokens] when [Token Protection] has a value of 'true' (Section 8.8).

3468 **D.4 Elements that are encrypted**

- 3469 1. The `ds:Signature` element that forms the message signature when [Signature Protection]
3470 has a value of 'true' (Section 6.4).
- 3471 2. All `wssell:SignatureConfirmation` elements when [Signature Protection] has a value
3472 of 'true' (Section 6.4).
- 3473 3. A `wsse:UsernameToken` may be encrypted when a transport binding is not being used
3474 (Section 5.3.1).

3475

3476

E. Acknowledgements

3477 The following individuals have participated in the creation of this specification and are gratefully
3478 acknowledged:

3479 **Original Authors of the initial contribution:**

3480 Giovanni Della-Libera, Microsoft
3481 Martin Gudgin, Microsoft
3482 Phillip Hallam-Baker, VeriSign
3483 Maryann Hondo, IBM
3484 Hans Granqvist, Verisign
3485 Chris Kaler, Microsoft (editor)
3486 Hiroshi Maruyama, IBM
3487 Michael McIntosh, IBM
3488 Anthony Nadalin, IBM (editor)
3489 Nataraj Nagaratnam, IBM
3490 Rob Philpott, RSA Security
3491 Hemma Prafullchandra, VeriSign
3492 John Shewchuk, Microsoft
3493 Doug Walter, Microsoft
3494 Riaz Zolfonoon, RSA Security

3495

3496 **Original Acknowledgements of the initial contribution:**

3497 Vaithialingam B. Balayoghan, Microsoft
3498 Francisco Curbera, IBM
3499 Christopher Ferris, IBM
3500 Cédric Fournet, Microsoft
3501 Andy Gordon, Microsoft
3502 Tomasz Janczuk, Microsoft
3503 David Melgar, IBM
3504 Mike Perks, IBM
3505 Bruce Rich, IBM
3506 Jeffrey Schlimmer, Microsoft
3507 Chris Sharp, IBM
3508 Kent Tamura, IBM
3509 T.R. Vishwanath, Microsoft
3510 Elliot Waingold, Microsoft

3511

3512 **TC Members during the development of this specification:**

3513 Don Adams, Tibco Software Inc.
3514 Jan Alexander, Microsoft Corporation
3515 Steve Anderson, BMC Software
3516 Donal Arundel, IONA Technologies
3517 Howard Bae, Oracle Corporation
3518 Abbie Barbir, Nortel Networks Limited
3519 Charlton Barreto, Adobe Systems
3520 Michael Botha, Software AG, Inc.
3521 Toufic Boubez, Layer 7 Technologies Inc.
3522 Norman Brickman, Mitre Corporation
3523 Melissa Brumfield, Booz Allen Hamilton

3524 Lloyd Burch, Novell
3525 Scott Cantor, Internet2
3526 Greg Carpenter, Microsoft Corporation
3527 Steve Carter, Novell
3528 Ching-Yun (C.Y.) Chao, IBM
3529 Martin Chapman, Oracle Corporation
3530 Kate Cherry, Lockheed Martin
3531 Henry (Hyenvui) Chung, IBM
3532 Luc Clement, Systinet Corp.
3533 Paul Cotton, Microsoft Corporation
3534 Glen Daniels, Sonic Software Corp.
3535 Peter Davis, Neustar, Inc.
3536 Martijn de Boer, SAP AG
3537 Werner Dittmann, Siemens AG
3538 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
3539 Fred Dushin, IONA Technologies
3540 Petr Dvorak, Systinet Corp.
3541 Colleen Evans, Microsoft Corporation
3542 Ruchith Fernando, WSO2
3543 Mark Fussell, Microsoft Corporation
3544 Vijay Gajjala, Microsoft Corporation
3545 Marc Goodner, Microsoft Corporation
3546 Hans Granqvist, VeriSign
3547 Martin Gudgin, Microsoft Corporation
3548 Tony Gullotta, SOA Software Inc.
3549 Jiandong Guo, Sun Microsystems
3550 Phillip Hallam-Baker, VeriSign
3551 Patrick Harding, Ping Identity Corporation
3552 Heather Hinton, IBM
3553 Frederick Hirsch, Nokia Corporation
3554 Jeff Hodges, Neustar, Inc.
3555 Will Hopkins, BEA Systems, Inc.
3556 Alex Hristov, Otecia Incorporated
3557 John Hughes, PA Consulting
3558 Diane Jordan, IBM
3559 Venugopal K, Sun Microsystems
3560 Chris Kaler, Microsoft Corporation
3561 Dana Kaufman, Forum Systems, Inc.
3562 Paul Knight, Nortel Networks Limited
3563 Ramanathan Krishnamurthy, IONA Technologies
3564 Christopher Kurt, Microsoft Corporation
3565 Kelvin Lawrence, IBM
3566 Hubert Le Van Gong, Sun Microsystems
3567 Jong Lee, BEA Systems, Inc.
3568 Rich Levinson, Oracle Corporation
3569 Tommy Lindberg, Dajeil Ltd.
3570 Mark Little, JBoss Inc.
3571 Hal Lockhart, BEA Systems, Inc.
3572 Mike Lyons, Layer 7 Technologies Inc.
3573 Eve Maler, Sun Microsystems
3574 Ashok Malhotra, Oracle Corporation
3575 Anand Mani, CrimsonLogic Pte Ltd
3576 Jonathan Marsh, Microsoft Corporation
3577 Robin Martherus, Oracle Corporation
3578 Miko Matsumura, Infravio, Inc.
3579 Gary McAfee, IBM
3580 Michael McIntosh, IBM

3581 John Merrells, Sxip Networks SRL
3582 Jeff Mischkinsky, Oracle Corporation
3583 Prateek Mishra, Oracle Corporation
3584 Bob Morgan, Internet2
3585 Vamsi Motukuru, Oracle Corporation
3586 Raajmohan Na, EDS
3587 Anthony Nadalin, IBM
3588 Andrew Nash, Reactivity, Inc.
3589 Eric Newcomer, IONA Technologies
3590 Duane Nickull, Adobe Systems
3591 Toshihiro Nishimura, Fujitsu Limited
3592 Rob Philpott, RSA Security
3593 Denis Pilipchuk, BEA Systems, Inc.
3594 Darren Platt, Ping Identity Corporation
3595 Martin Raepple, SAP AG
3596 Nick Ragouzis, Enosis Group LLC
3597 Prakash Reddy, CA
3598 Alain Regnier, Ricoh Company, Ltd.
3599 Irving Reid, Hewlett-Packard
3600 Bruce Rich, IBM
3601 Tom Rutt, Fujitsu Limited
3602 Maneesh Sahu, Actional Corporation
3603 Frank Siebenlist, Argonne National Laboratory
3604 Joe Smith, Apani Networks
3605 Davanum Srinivas, WSO2
3606 Yakov Sverdlov, CA
3607 Gene Thurston, AmberPoint
3608 Victor Valle, IBM
3609 Asir Vedamuthu, Microsoft Corporation
3610 Greg Whitehead, Hewlett-Packard
3611 Ron Williams, IBM
3612 Corinna Witt, BEA Systems, Inc.
3613 Kyle Young, Microsoft Corporation

3614

F. Revision History

3615 [optional; should not be included in OASIS Standards]

3616

Revision	Date	Editor	Changes Made
0.1	12-06-2005	Anthony Nadalin	Initial Conversion to OASIS Template
0.2	01-09-2006	Martin Gudgin	Updated TOC. Typos. Namespaces.
0.3.	01-17-2006	Marc Goodner	Updated artifact identifier per AIR guidelines, corrected version number, 2005 updated to 2006, added resolution of i012, changed status to editor draft
0.4	02-20-2006	Marc Goodner	Corrected section numbers after section reordering caused by OASIS template (Issue 21)
0.5	03-27-2006	Martin Gudgin	Issue 3 - Lines 229-236 Issue 9 - Lines 1620-1644, 1675, 1681, 1685-1708 Issue 23 - Lines 1300 and 1302 Issue 25 - Lines 1921-1932 Issue 26 - Line 1343 Issue 27 - Lines 766-783 Issue 29 - 1704-1707 Issue 32 - Line 828 (Text was removed not added) Issue 49 - Lines 1273, 1277-1278 Issue 50 - Lines 557-560
0.6	04-27-2006	Martin Gudgin	Issue 16 – Section 4.1.1 Issue 18 – Sections 1.6, 6.1, 7.1 Issue 30 – Section 11 Issue 51 – Sections 5.2.1, 5.2.2, 5.2.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5, 5.3.6, 5.3.7, 5.3.8, 5.3.9 Issue 53 – Section 5.3.7
0.7	06-19-2006	Martin Gudgin	Issue 31 – Updates to Section 5.3.1 Issue 33 – Added new Appendix D Issue 48 – Updates to Sections 7.4, 7.5 and Appendix A. Issue 69 – Updates to Sections 5.3 and 7.2. Issue 75 – Updates to Section 5.3.10
0.8	08-28-2006	Martin Gudgin Marc Goodner	Issue 71 – Updates to section 2.3 Issue 74 – Added sections 8.5, 8.6 and 8.7.

			<p>Updated Section 8, Appendix A.1.2, A.2.2 and A.3.1 accordingly.</p> <p>Issue 79 – Updates to Appendix A.4.1</p> <p>Issue 80 – Updates to section 8</p> <p>Issue 82 – Updates to section 1.5</p> <p>Issue 84 – Updates to sections 3.1.2, 3.1.3</p> <p>Issue 85 – Updates to line 111, section 1.4.1</p> <p>Issue 88 – Updates to sections 4.1.2, 4.2.2 and 4.3.1</p> <p>Issue 89 – Removed Appendix F</p> <p>Issue 91 – Updated Appendix C.3.2 and C.3.3</p> <p>Issue 92 – Updates to Section 10 and 10.1</p> <p>Issue 94 – Edits to Section 3.1.3</p> <p>Issue 95 – Edits to Sections 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.5, 5.3.6, 5.3.7, 5.3.8, 5.3.9, 5.3.10, 7.1, 7.2, 7.3, 7.4, 7.5, 8.1, 8.2, 8.3, 8.4</p> <p>Issue 97 – Edits to Section 4.2.1</p> <p>Issue 98 – Edit to Section 4.1.1</p> <p>Issue 4 – Section 1.5 and throughout</p> <p>Editorial nits, corrected some hyperlinks and references, some xml examples and document location.</p> <p>Updated notational conventions inline with other SX specs.</p> <p>Added TC acknowledgements.</p>
0.9	09-01-2006	Marc Goodner	<p>Accepted previous changes.</p> <p>More editorial nits.</p> <p>Updated notational convention regarding extensibility points may exist in exemplar and not in the text.</p> <p>Issue 8 – throughout</p> <p>Issue 109 – 5.3.2 and appendix B</p> <p>Removed section 3.1 content, added reference to WS-Policy section on nested policy.</p> <p>Updated references to section 3.1 to refer to WS-Policy.</p>
0.10	10-04-2006	Marc Goodner	<p>i083 – updated appendix diagrams</p> <p>i086 – added section 4.2.3</p> <p>i096 – applied changes to section A</p> <p>i110 – added section 4.3.2</p> <p>Editorial nits: Corrected WSS11 ns in ns prefix table; Corrected examples in Section 8.9, C.1.1, C.2.1 to show proper nesting of X509v3 assertion; fixed incomplete exemplar in Section 7.5, updated C.3.2 point four to refer to all supporting tokens in section 8</p>

0.11	11-13-2006	Marc Goodner	<p>i090 – Updated section 6.7.1, added section 6.7.2</p> <p>i114 – Added normative SOAP 1.2 message normalization reference in section 1.5</p> <p>i115 – Updated Appendix D, 5.3.1</p> <p>i116 – Updated Appendix D, 8.3 and 9.</p> <p>i117 – Updated 8.5, 8.6, and 8.7 with text advising to use element encr for protecting supporting tokens</p> <p>Editorial nits: Improper caps in 4.3.2 exemplar, further corrections to exemplar in 7.5, corrected missing “Body” id in symmetric and asymmetric examples in appendix C</p>
0.12	12-04-2006	Marc Goodner	<p>i118 – Updated example in section 1.1</p> <p>i123 – Added reference to wsse:InvalidSecurity in section 9</p> <p>Editorial – Nits throughout, updated style to amtch OASIS template, corrected reference to schema file, added note that section 12 is also non-normative, included note that example 1.1 does not show attachment point, corrected policy in C.3.1 to agree with subsequent examples, corrected figure in C.3.2 (Sig2 should not point to ST1), further housecleaning in 7.5, updated EncryptBeforeSigning to agree with 7.5, rephrased D.4 point 3</p>

3617