



WS-SecurityPolicy 1.2

Committee Draft 01, 4 December 2006

Artifact Identifier:

ws-securitypolicy-1.2-spec-cd-01

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	18
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	21
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	41
6.6	[Entire Header and Body Signatures] Property	41
6.7	[Security Header Layout] Property	41
6.7.1	Strict Layout Rules for WSS 1.0	41
7	Security Binding Assertions	44
7.1	AlgorithmSuite Assertion	44
7.2	Layout Assertion	46
7.3	TransportBinding Assertion	47
7.4	SymmetricBinding Assertion	48
7.5	AsymmetricBinding Assertion	50
8	Supporting Tokens	53
8.1	SupportingTokens Assertion	54
8.2	SignedSupportingTokens Assertion	55
8.3	EndorsingSupportingTokens Assertion	57
8.4	SignedEndorsingSupportingTokens Assertion	59
8.5	SignedEncryptedSupportingTokens Assertion	61
8.6	EndorsingEncryptedSupportingTokens Assertion	61
8.7	SignedEndorsingEncryptedSupportingTokens Assertion	61
8.8	Interaction between [Token Protection] property and supporting token assertions	61
8.9	Example	61
9	WSS: SOAP Message Security Options	63
9.1	Wss10 Assertion	64
9.2	Wss11 Assertion	65
10	WS-Trust Options	67
10.1	Trust10 Assertion	68
11	Guidance on creating new assertions and assertion extensibility	69
11.1	General Design Points	69
11.2	Detailed Design Guidance	69
12	Security Considerations	71
A.	Assertions and WS-PolicyAttachment	72
A.1	Endpoint Policy Subject Assertions	72
A.1.1	Security Binding Assertions	72
A.1.2	Token Assertions	72
A.1.3	WSS: SOAP Message Security 1.0 Assertions	72

A.1.4 WSS: SOAP Message Security 1.1 Assertions	72
A.1.5 Trust 1.0 Assertions	72
A.2 Operation Policy Subject Assertions	72
A.2.1 Security Binding Assertions	72
A.2.2 Supporting Token Assertions	72
A.3 Message Policy Subject Assertions	73
A.3.1 Supporting Token Assertions	73
A.3.2 Protection Assertions	73
A.4 Assertions With Undefined Policy Subject	73
A.4.1 General Assertions	73
A.4.2 Token Usage Assertions	73
A.4.3 Token Assertions	73
B. Issued Token Policy	75
C. Strict Security Header Layout Examples	77
C.1 Transport Binding	77
C.1.1 Policy	77
C.1.2 Initiator to Recipient Messages	78
C.1.3 Recipient to Initiator Messages	79
C.2 Symmetric Binding	80
C.2.1 Policy	81
C.2.2 Initiator to Recipient Messages	82
C.2.3 Recipient to Initiator Messages	86
C.3 Asymmetric Binding	89
C.3.1 Policy	89
C.3.2 Initiator to Recipient Messages	91
C.3.3 Recipient to Initiator Messages	95
D. Signed and Encrypted Elements in the Security Header	99
D.1 Elements signed by the message signature	99
D.2 Elements signed by all endorsing signatures	99
D.3 Elements signed by a specific endorsing signature	99
D.4 Elements that are encrypted	99
E. Acknowledgements	100

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 1997
151		
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	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
158		
159		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
160		
161	[WS-Policy]	W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
162		
163		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
164		
165	[WS-PolicyAttachment]	W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
166		
167		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
168		
169		
170	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.3", September 2006
171		http://docs.oasis-open.org/ws-sx/ws-trust/200512
172		
173	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.3", September 2006
174		
175		http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
176		
177	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
178		
179		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
180		
181		
182	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
183		
184		http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
185		
186		
187	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004
188		
189		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
190		
191		
192	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
193		
194		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
195		

196		
197	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
198		
199		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
200		
201		
202	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
203		
204		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
205		
206		
207	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
208		
209		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
210		
211		
212	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
213		
214		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
215		
216	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
217		
218		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
219		
220		
221	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
222		
223		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
224		
225	[WSS:RELTTokenProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile 1.1", February 2006
226		
227		http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
228		
229		
230	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
231		
232		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
233		
234	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002.
235		
236		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
237		
238	[XPath]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999.
239		
240		http://www.w3.org/TR/1999/REC-xpath-19991116
241		

242 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second
243 Edition", 28 October 2004.
244 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
245

246 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second
247 Edition", 28 October 2004.
248 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
249
250

251 **1.6 Non-Normative References**

252 None.
253

254 **2 Security Policy Model**

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

259
260 The primary goal of this specification is to define an initial set of patterns or sets of assertions that
261 represent common ways to describe how messages are secured on a communication path. The intent is
262 to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging
263 transport security, but to be specific enough to ensure interoperability based on assertion matching.

264
265 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for
266 selecting policy alternatives and the attachment mechanism for associating policy assertions with web
267 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters
268 or attributes. This enables first-level, QName based assertion matching without security domain-specific
269 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed
270 set of policy alternatives that are shared by the two parties attempting to establish a secure
271 communication path.

272
273 In general, assertions defined in this specification allow additional attributes, based on schemas, to be
274 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not
275 match based on these attributes. Attributes specified on the assertion element that are not defined in this
276 specification or in WS-Policy are to be treated as informational properties.

277 **2.1 Security Assertion Model**

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

285
286 To indicate the scope of protection, assertions identify message parts that are to be protected in a
287 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

288
289 The general aspects of security includes the relationships between or characteristics of the environment
290 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality
291 protection and which are supporting, the applicable algorithms to use, etc.

292
293 The security binding assertion is a logical grouping which defines how the general aspects are used to
294 protect the indicated parts. For example, that an asymmetric token is used with a digital signature to
295 provide integrity protection, and that parts are encrypted with a symmetric key which is then encrypted

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

298

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

303 **2.2 Nested Policy Assertions**

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

306 **2.3 Security Binding Abstraction**

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

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

312

313 A binding defines the following security characteristics:

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

322

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

329

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

- 333 • TransportBinding (Section 7.3)
- 334 • SymmetricBinding (Section 7.4)
- 335 • AsymmetricBinding (Section 7.5)

336 3 Policy Considerations

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

339 3.1 Nested Policy

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

342

343 3.2 Policy Subjects

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

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

348 [Message Policy Subject]

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

351

352 wsdl:message

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

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

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

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

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

361 [Operation Policy Subject]

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

363 wsdl:portType/wsdl:operation

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

366 wsdl:binding/wsdl:operation

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

369

370

371 [Endpoint Policy Subject]

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

374 wsdl:portType

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

377 wsdl:binding

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

380 wsdl:port

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

383 4 Protection Assertions

384 The following assertions are used to identify *what* is being protected and the level of protection provided.
385 These assertions SHOULD apply to [Message Policy Subject]. These assertions MAY apply to [Endpoint
386 Policy Subject] or [Operation Policy Subject]. Where they apply to [Operation Policy Subject] they apply to
387 all messages of that operation. Where they apply to [Endpoint Policy Subject] they apply to all operations
388 of that endpoint.

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

391 4.1 Integrity Assertions

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

395 4.1.1 SignedParts Assertion

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

401
402 There MAY be multiple SignedParts assertions present. Multiple SignedParts assertions present within a
403 policy alternative are equivalent to a single SignedParts assertion containing the union of all specified
404 message parts. Note that this assertion does not require that a given part appear in a message, just that if
405 such a part appears, it requires integrity protection.

406 Syntax

```
407 <sp:SignedParts xmlns:sp="..." ... >  
408   <sp:Body />?  
409   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
410   ...  
411 </sp:SignedParts>
```

412
413 The following describes the attributes and elements listed in the schema outlined above:

414 /sp:SignedParts

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

418 /sp:SignedParts/sp:Body

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

421 /sp:SignedParts/sp:Header

422 Presence of this optional element indicates a specific SOAP header, it's attributes and content (or
423 set of such headers) needs to be protected. There may be multiple sp:Header elements within a
424 single sp:SignedParts element. If multiple SOAP headers with the same local name but different

425 namespace names are to be integrity protected multiple sp:Header elements are needed, either
426 as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts assertions.
427 This element only applies to SOAP header elements targeted to the same actor/role as the
428 Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
429 SOAP Header elements targeted to a different actor/role, that may be accomplished using the
430 sp:SignedElements assertion.

431 /sp:SignedParts/sp:Header/@Name

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

435 /sp:SignedParts/sp:Header/@Namespace

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

437 4.1.2 SignedElements Assertion

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

443

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

447 Syntax

```
448 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
449 <sp:XPath>xs:string</sp:XPath>+  
450 ...  
451 </sp:SignedElements>
```

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

453 /sp:SignedElements

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

455 /sp:SignedElements/@XPathVersion

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

458 /sp:SignedElements/sp:XPath

459 This element contains a string specifying an XPath expression that identifies the nodes to be
460 integrity protected. The XPath expression is evaluated against the S:Envelope element node of
461 the message. Multiple instances of this element may appear within this assertion and should be
462 treated as separate references in a signature when message security is used.

463 4.2 Confidentiality Assertions

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

467 4.2.1 EncryptedParts Assertion

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

472

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

477 Syntax

```
478 <sp:EncryptedParts xmlns:sp="..." ... >  
479   <sp:Body/>?  
480   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
481   ...  
482 </sp:EncryptedParts>
```

483

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

485 /sp:EncryptedParts

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

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

489 /sp:EncryptedParts/sp:Body

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

494 /sp:EncryptedParts/sp:Header

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

503 /sp:EncryptedParts/sp:Header/@Name

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

507 /sp:EncryptedParts/sp:Header/@Namespace

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

510 4.2.2 EncryptedElements Assertion

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

516

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

520 Syntax

```
521 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
522   <sp:XPath>xs:string</sp:XPath>+  
523   ...  
524 </sp:EncryptedElements>
```

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

526 /sp:EncryptedElements

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

529 /sp:EncryptedElements/@XPathVersion

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

532 /sp:EncryptedElements/sp:XPath

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

537 4.2.3 ContentEncryptedElements Assertion

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

543

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

547 Syntax

```
548 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
549   <sp:XPath>xs:string</sp:XPath>+  
550   ...  
551 </sp:ContentEncryptedElements>
```

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

553 /sp:ContentEncryptedElements

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

556 /sp:ContentEncryptedElements/@XPathVersion
557 This optional attribute contains a URI which indicates the version of XPath to use.
558 /sp:ContentEncryptedElements/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 4.3 Required Elements Assertion

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

566

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

570 4.3.1 RequiredElements Assertion

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

573

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

577 Syntax

```
578 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
579   <sp:XPath>xs:string</sp:XPath> +  
580   ...  
581 </sp:RequiredElements>
```

582

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

584 /sp:RequiredElements

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

586 /sp:RequiredElements/@XPathVersion

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

589 /sp:RequiredElements/sp:XPath

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

594 4.3.2 RequiredParts Assertion

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

598

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

602 **Syntax**

```
603 <sp:RequiredParts XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
604   <sp:Header Name = "..." Namespace= "..." /> +  
605 </sp:RequiredParts>
```

606

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

608 /sp:RequiredParts/sp:Header

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

610 /sp:RequiredParts/sp:Header/@Name

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

613 /sp:RequiredParts/sp:Header/@Namespace

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

616 5 Token Assertions

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

621 5.1 Token Inclusion

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

627 5.1.1 Token Inclusion Values

628 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.

629
630 Note: In examples, the namespace URI is replaced with "...". For example,
631 `.../IncludeToken/Never` is actually `http://docs.oasis-open.org/ws-sx/ws-`
632 `securitypolicy/200512/IncludeToken/Never`. Other token inclusion URI values MAY be defined but are out-
633 of-scope of this specification.

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

636 5.1.2 Token Inclusion and Token References

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

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

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

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

653 5.2 Token Properties

654 5.2.1 [Derived Keys] Property

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

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

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

663 5.2.2 [Explicit Derived Keys] Property

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

667 5.2.3 [Implicit Derived Keys] Property

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

671 5.3 Token Assertion Types

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

673 5.3.1 UsernameToken Assertion

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

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

- 676 1. When transport security is not used.
- 677 2. When a plaintext password is used.
- 678 3. When a weak password hash is used.
- 679 4. When the username needs to be protected, e.g. for privacy reasons.

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

683

684 Syntax

```
685 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
686   <wsp:Policy xmlns:wsp="...">  
687     (  
688       <sp:NoPassword ... /> |  
689       <sp:HashPassword ... />  
690     ) ?  
691     (  
692       <sp:RequireDerivedKeys /> |  
693       <sp:RequireImplicitDerivedKeys ... /> |  
694       <sp:RequireExplicitDerivedKeys ... />  
695     ) ?  
696     (  
697       <sp:WssUsernameToken10 ... /> |  
698       <sp:WssUsernameToken11 ... />  
699     ) ?  
700     ...  
701   </wsp:Policy> ?  
702   ...  
703 </sp:UsernameToken>
```

704

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

706 /sp:UsernameToken

707 This identifies a UsernameToken assertion.

708 /sp:UsernameToken/@sp:IncludeToken

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

710 /sp:UsernameToken/wsp:Policy

711 This optional element identifies additional requirements for use of the sp:UsernameToken
712 assertion.

713 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

716 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

721 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

724 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 725 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
 726 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.
 727 /sp:UsernameToken/wsp:Policy/sp:RequireImplicitDerivedKeys
 728 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
 729 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.
 730 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10
 731 This optional element is a policy assertion that indicates that a Username token should be used
 732 as defined in [WSS:UsernameTokenProfile1.0].
 733 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11
 734 This optional element is a policy assertion that indicates that a Username token should be used
 735 as defined in [WSS:UsernameTokenProfile1.1].

736 5.3.2 IssuedToken Assertion

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

741 Syntax

```

742 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
743   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
744   <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
745     ...
746   </sp:RequestSecurityTokenTemplate>
747   <wsp:Policy xmlns:wsp="...">
748     (
749       <sp:RequireDerivedKeys ... /> |
750       <sp:RequireImplicitDerivedKeys ... /> |
751       <sp:RequireExplicitDerivedKeys ... />
752     ) ?
753   <sp:RequireExternalReference ... /> ?
754   <sp:RequireInternalReference ... /> ?
755   ...
756 </wsp:Policy> ?
757   ...
758 </sp:IssuedToken>
  
```

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

760 /sp:IssuedToken

761 This identifies an IssuedToken assertion.

762 /sp:IssuedToken/@sp:IncludeToken

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

764 /sp:IssuedToken/sp:Issuer

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

767 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

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

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

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

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

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

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

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

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

799 **5.3.3 X509Token Assertion**

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

801 **Syntax**

```

802 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
803   <wsp:Policy xmlns:wsp="...">
804     (
805       <sp:RequireDerivedKeys ... /> |
806       <sp:RequireExplicitDerivedKeys ... /> |
807       <sp:RequireImplicitDerivedKeys ... />
808     ) ?
809   <sp:RequireKeyIdentifierReference ... /> ?
810   <sp:RequireIssuerSerialReference ... /> ?
811   <sp:RequireEmbeddedTokenReference ... /> ?
812   <sp:RequireThumbprintReference ... /> ?
813   (
814     <sp:WssX509V3Token10 ... /> |
815     <sp:WssX509Pkcs7Token10 ... /> |
816     <sp:WssX509PkiPathV1Token10 ... /> |
817     <sp:WssX509V1Token11 ... /> |
818     <sp:WssX509V3Token11 ... /> |
819     <sp:WssX509Pkcs7Token11 ... /> |
820     <sp:WssX509PkiPathV1Token11 ... />
821   ) ?
822   ...
823 </wsp:Policy> ?
824   ...
825 </sp:X509Token>

```

826
827 The following describes the attributes and elements listed in the schema outlined above:
828 /sp:X509Token

829 This identifies an X509Token assertion.

830 /sp:X509Token/@sp:IncludeToken

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

832 /sp:X509Token/wsp:Policy

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

834 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

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

837 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

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

840 /sp:X509Token/wsp:Policy/sp:RequireImplicitDerivedKeys

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

843 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

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

846 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

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

849 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

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

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

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

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

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

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

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

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

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

876 5.3.4 KerberosToken Assertion

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

878 Syntax

```

879 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
880 <wsp:Policy xmlns:wsp="...">
881   (
882     <sp:RequireDerivedKeys ... /> |
883     <sp:RequireImplicitDerivedKeys ... /> |
884     <sp:RequireExplicitDerivedKeys ... />
885   ) ?
886   <sp:RequireKeyIdentifierReference ... /> ?
887   (
888     <sp:WssKerberosV5ApReqToken11 ... /> |
889     <sp:WssGssKerberosV5ApReqToken11 ... />
890   ) ?
891   ...
892 </wsp:Policy> ?
893   ...
894 </sp:KerberosToken>
  
```

896

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

898 /sp:KerberosToken

899 This identifies a KerberosV5ApReqToken assertion.

900 /sp:KerberosToken/@sp:IncludeToken

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

902 /sp:KerberosToken/wsp:Policy

903 This optional element identifies additional requirements for use of the sp:KerberosToken
904 assertion.

905 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

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

908 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

911 /sp:KerberosToken/wsp:Policy/sp:RequireImplicitDerivedKeys

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

914 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

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

917 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

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

920 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

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

923 5.3.5 SpnegoContextToken Assertion

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

926 Syntax

```

927 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
928   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> ?
929   <wsp:Policy xmlns:wsp="...">
930     (
931       <sp:RequireDerivedKeys ... /> |
932       <sp:RequireImplicitDerivedKeys ... /> |
933       <sp:RequireExplicitDerivedKeys ... />
934     ) ?
935     ...
936   </wsp:Policy> ?
937   ...
938 </sp:SpnegoContextToken>

```

939

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

941 /sp:SpnegoContextToken

942 This identifies a SpnegoContextToken assertion.

943 /sp:SpnegoContextToken/@sp:IncludeToken

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

945 /sp:SpnegoContextToken/sp:Issuer

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

947 Spnego Context Token.

948 /sp:SpnegoContextToken/wsp:Policy

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

950 assertion.

951 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

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

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

954 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

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

957 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImplicitDerivedKeys

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

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

960 5.3.6 SecurityContextToken Assertion

961 This element represents a requirement for a SecurityContextToken token.

962 Syntax

```

963 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
964   <wsp:Policy xmlns:wsp="...">
965     (
966       <sp:RequireDerivedKeys ... /> |
967       <sp:RequireImplicitDerivedKeys ... /> |
968       <sp:RequireExplicitDerivedKeys ... />
969     ) ?
970     <sp:RequireExternalUriReference ... /> ?
971     <sp:SC200502SecurityContextToken ... /> ?
972     ...
973   </wsp:Policy> ?
974   ...
975 </sp:SecurityContextToken>

```

976

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

978 /sp:SecurityContextToken

979 This identifies a SecurityContextToken assertion.

980 /sp:SecurityContextToken/@sp:IncludeToken

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

982 /sp:SecurityContextToken/wsp:Policy

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

984 assertion.

985 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

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

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

988 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

991 /sp:SecurityContextToken/wsp:Policy/sp:RequireImplicitDerivedKeys

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

994 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

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

997 /sp:SecurityContextToken/wsp:Policy/sp:SC200502SecurityContextToken

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

1000

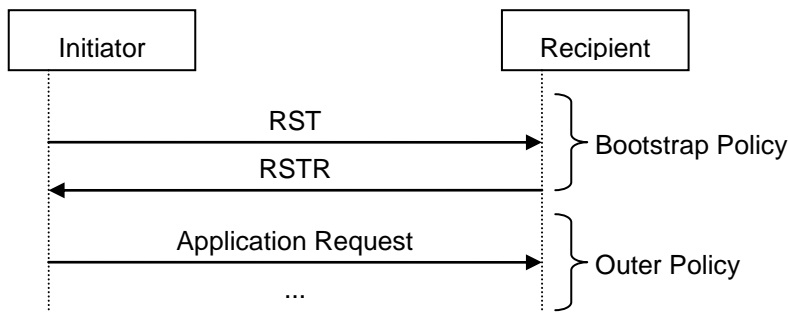
1001 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
1002 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
1003 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
1004 sp:SecureConversationToken or the sp:IssuedToken assertion should be used instead.

1005 5.3.7 SecureConversationToken Assertion

1006 This element represents a requirement for a Security Context Token retrieved from the indicated issuer
1007 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
1008 service endpoint address.

1009

1010 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1011 the target service and may not have a separate port (with separate policy), this assertion SHOULD
1012 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1013 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1014 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1015 below.



1016

1017 **Syntax**


```

1018 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1019   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
1020   <wsp:Policy xmlns:wsp="...">
1021     (
1022       <sp:RequireDerivedKeys ... /> |
1023       <sp:RequireImplicitDerivedKeys ... /> |
1024       <sp:RequireExplicitDerivedKeys ... />
1025     ) ?
1026     <sp:RequireExternalUriReference ... /> ?
1027     <sp:SC200502SecurityContextToken ... /> ?
1028     <sp:BootstrapPolicy ... > ?
1029     <wsp:Policy> ... </wsp:Policy>
1030   </sp:BootstrapPolicy>
1031 </wsp:Policy> ?
1032   ...
1033 </sp:SecureConversationToken>

```

1034

1035 The following describes the attributes and elements listed in the schema outlined above:

1036 /sp:SecureConversationToken

1037 This identifies a SecureConversationToken assertion.

1038 /sp:SecureConversationToken/@sp:IncludeToken

1039 This optional attribute identifies the token inclusion value for this token assertion.

1040 /sp:SecureConversationToken/sp:Issuer

1041 This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for the
1042 Security Context Token.

1043 /sp:SecureConversationToken/wsp:Policy

1044 This optional element identifies additional requirements for use of the
1045 sp:SecureConversationToken assertion.

1046 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1047 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1048 and [Implicit Derived Keys] properties for this token to 'true'.

1049 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1050 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1051 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

1052 /sp:SecureConversationToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1053 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
1054 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1055 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1056 This optional element is a policy assertion that indicates that an external URI reference is
1057 required when referencing this token.

1058 /sp:SecureConversationToken/wsp:Policy/sp:SC200502SecurityContextToken

1059 This optional element is a policy assertion that indicates that a Security Context Token should be
1060 used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1061 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1062 This optional element is a policy assertion that contains the policy indicating the requirements for
1063 obtaining the Security Context Token.

1064 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1065 This element contains the security binding requirements for obtaining the Security Context Token. It
1066 will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with protection
1067 assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that are to be
1068 protected.

1069 **Example**

```
1070 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
1071   <sp:SymmetricBinding>  
1072     <wsp:Policy>  
1073       <sp:ProtectionToken>  
1074         <wsp:Policy>  
1075           <sp:SecureConversationToken>  
1076             <sp:Issuer>  
1077               <wsa:Address>http://example.org/sts</wsa:Address>  
1078             </sp:Issuer>  
1079           <wsp:Policy>  
1080             <sp:SC10SecurityContextToken />  
1081           <sp:BootstrapPolicy>  
1082             <wsp:Policy>  
1083               <sp:AsymmetricBinding>  
1084                 <wsp:Policy>  
1085                   <sp:InitiatorToken>  
1086                     ...  
1087                   </sp:InitiatorToken>  
1088                   <sp:RecipientToken>  
1089                     ...  
1090                   </sp:RecipientToken>  
1091                 </wsp:Policy>  
1092               </sp:AsymmetricBinding>  
1093             <sp:SignedParts>  
1094               ...  
1095             </sp:SignedParts>  
1096           </wsp:Policy>  
1097         </sp:BootstrapPolicy>  
1098       </wsp:Policy>  
1099     </sp:SecureConversationToken>  
1100   </wsp:Policy>  
1101 </sp:ProtectionToken>  
1102   ...  
1103 </wsp:Policy>  
1104 </sp:SymmetricBinding>  
1105 <sp:SignedParts>  
1106   ...  
1107 </sp:SignedParts>  
1108   ...  
1109 </wsp:Policy>
```

1111 **5.3.8 SamlToken Assertion**

1112 This element represents a requirement for a SAML token.

1113 **Syntax**

```

1114 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1115   <wsp:Policy xmlns:wsp="...">
1116     (
1117       <sp:RequireDerivedKeys ... /> |
1118       <sp:RequireImplicitDerivedKeys ... /> |
1119       <sp:RequireExplicitDerivedKeys ... />
1120     ) ?
1121     <sp:RequireKeyIdentifierReference ... /> ?
1122     (
1123       <sp:WssSamlV11Token10 ... /> |
1124       <sp:WssSamlV11Token11 ... /> |
1125       <sp:WssSamlV20Token11 ... />
1126     ) ?
1127     ...
1128   </wsp:Policy> ?
1129   ...
1130 </sp:SamlToken>

```

1131

1132 The following describes the attributes and elements listed in the schema outlined above:

1133 /sp:SamlToken

1134 This identifies a SamlToken assertion.

1135 /sp:SamlToken/@sp:IncludeToken

1136 This optional attribute identifies the token inclusion value for this token assertion.

1137 /sp:SamlToken/wsp:Policy

1138 This optional element identifies additional requirements for use of the sp:SamlToken assertion.

1139 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

1140 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1141 and [Implicit Derived Keys] properties for this token to 'true'.

1142 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1143 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
 1144 properties for this token to 'true' and the [Implicit Derived Keys] property for this token to 'false'.

1145 /sp:SamlToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1146 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
 1147 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1148 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1149 This optional element is a policy assertion that indicates that a key identifier reference is required
 1150 when referencing this token.

1151 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10

1152 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1153 used as defined in [\[WSS:SAMLTokenProfile1.0\]](#).

1154 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11

1155 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1156 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1157 /sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11

1158 This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be
 1159 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1160

1161 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1162 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1163 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion should
1164 be used instead.

1165 5.3.9 RelToken Assertion

1166 This element represents a requirement for a REL token.

1167 Syntax

```
1168 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1169 <wsp:Policy xmlns:wsp="...">  
1170 (   
1171 <sp:RequireDerivedKeys ... /> |   
1172 <sp:RequireImplicitDerivedKeys ... /> |   
1173 <sp:RequireExplicitDerivedKeys ... />   
1174 ) ?   
1175 <sp:RequireKeyIdentifierReference ... /> ?   
1176 (   
1177 <sp:WssRelV10Token10 ... /> |   
1178 <sp:WssRelV20Token10 ... /> |   
1179 <sp:WssRelV10Token11 ... /> |   
1180 <sp:WssRelV20Token11 ... />   
1181 ) ?   
1182 ...   
1183 </wsp:Policy> ?   
1184 ...   
1185 </sp:RelToken>
```

1186
1187 The following describes the attributes and elements listed in the schema outlined above:

1188 /sp:RelToken

1189 This identifies a RelToken assertion.

1190 /sp:RelToken/@sp:IncludeToken

1191 This optional attribute identifies the token inclusion value for this token assertion.

1192 /sp:RelToken/wsp:Policy

1193 This optional element identifies additional requirements for use of the sp:RelToken assertion.

1194 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1195 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1196 and [Implicit Derived Keys] property for this token to 'true'.

1197 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1200 /sp:RelToken/wsp:Policy/sp:RequireImplicitDerivedKeys

1201 This optional element is a policy assertion that sets the [Derived Keys] and [Implicit Derived Keys]
1202 properties for this token to 'true' and the [Explicit Derived Keys] property for this token to 'false'.

1203 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1204 This optional element is a policy assertion that indicates that a key identifier reference is required
1205 when referencing this token.

1206 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1207 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1208 used as defined in [\[WSS:RELTOKENPROFILE1.0\]](#).

1209 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1210 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1211 used as defined in [\[WSS:RELTOKENPROFILE1.0\]](#).

1212 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1213 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1214 used as defined in [\[WSS:RELTOKENPROFILE1.1\]](#).

1215 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1216 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1217 used as defined in [\[WSS:RELTOKENPROFILE1.1\]](#).

1218

1219 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
1220 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1221 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion should
1222 be used instead.

1223 5.3.10 HttpsToken Assertion

1224 This element represents a requirement for a transport binding to support the use of HTTPS.

1225 Syntax

```
1226 <sp:HttpsToken xmlns:sp="..." ... >  
1227   <wsp:Policy xmlns:wsp="...">  
1228     (  
1229       <sp:HttpBasicAuthentication /> |  
1230       <sp:HttpDigestAuthentication /> |  
1231       <sp:RequireClientCertificate /> |  
1232       ...  
1233     )?  
1234     ...  
1235   </wsp:Policy> ?  
1236   ...  
1237 </sp:HttpsToken>
```

1238 The following describes the attributes and elements listed in the schema outlined above:

1239 /sp:HttpsToken

1240 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1241 supported.

1242 /sp:HttpsToken/wsp:Policy

1243 This optional element identifies additional requirements for use of the sp:HttpsToken assertion.

1244 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1245 This optional element is a policy assertion that indicates that the client MUST use HTTP Basic
1246 Authentication [\[RFC2068\]](#) to authenticate to the service.

1247 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1248 This optional element is a policy assertion that indicates that the client MUST use HTTP Digest
1249 Authentication [\[RFC2068\]](#) to authenticate to the service.

1250 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1251 This optional element is a policy assertion that indicates that the client MUST provide a certificate
1252 when negotiating the HTTPS session.

1253 6 Security Binding Properties

1254 This section defines the various properties or conditions of a security binding, their semantics, values and
1255 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are
1256 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that
1257 populates a value of a property appears in a policy, that property is set to the value indicated by the
1258 assertion. The security binding then uses the value of the property to control its behavior. The properties
1259 listed here are common to the various security bindings described in Section 7. Assertions that define
1260 values for these properties are defined in Section 7. The following properties are used by the security
1261 binding assertions.

1262 6.1 [Algorithm Suite] Property

1263 This property specifies the algorithm suite required for performing cryptographic operations with
1264 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and
1265 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This
1266 property defines the set of available algorithms. The value of this property is typically referenced by a
1267 security binding and is used to specify the algorithms used for all message level cryptographic operations
1268 performed under the security binding.

1269 Note: In some cases, this property MAY be referenced under a context other than a security binding and
1270 used to control the algorithms used under that context. For example, supporting token assertions define
1271 such a context. In such contexts, the specified algorithms still apply to message level cryptographic
1272 operations.

1273 An algorithm suite defines values for each of the following operations and properties:

- 1274 • [Sym Sig] Symmetric Key Signature
- 1275 • [Asym Sig] Signature with an asymmetric key
- 1276 • [Dig] Digest
- 1277 • [Enc] Encryption
- 1278 • [Sym KW] Symmetric Key Wrap
- 1279 • [Asym KW] Asymmetric Key Wrap
- 1280 • [Comp Key] Computed key
- 1281 • [Enc KD] Encryption key derivation
- 1282 • [Sig KD] Signature key derivation
- 1283 • [Min SKL] Minimum symmetric key length
- 1284 • [Max SKL] Maximum symmetric key length
- 1285 • [Min AKL] Minimum asymmetric key length
- 1286 • [Max AKL] Maximum asymmetric key length

1287

1288 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/xmllenc#sha256
Sha512	http://www.w3.org/2001/04/xmllenc#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
STRT10	http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-soap-message-security-1.0#STR-Transform
AbsXPath	http://docs.oasis-open.org/...TBD.../AbsXPath

1289

1290 The tables below show all the base algorithm suites defined by this specification. This table defines

1291 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

1292 This table defines additional properties whose values can be specified along with the default value for that
1293 property.

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

1294 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
Basic192Sha256Rsa15	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Sha256Rsa15	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesSha256Rsa15	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

1295 6.2 [Timestamp] Property

1296 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`
1297 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected
1298 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be
1299 present. The default value for this property is 'false'.

1300 6.3 [Protection Order] Property

1301 This property indicates the order in which integrity and confidentiality are applied to the message, in
1302 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.

1303 The default value for this property is 'SignBeforeEncrypting'.

1304 6.4 [Signature Protection] Property

1305 This boolean property specifies whether the signature must be encrypted. If the value is 'true', the primary
1306 signature MUST be encrypted and any signature confirmation elements MUST also be encrypted. If the
1307 value is 'false', the primary signature MUST NOT be encrypted and any signature confirmation elements
1308 MUST NOT be encrypted. The default value for this property is 'false'.

1309 6.5 [Token Protection] Property

1310 This boolean property specifies whether signatures must cover the token used to generate that signature.
1311 If the value is 'true', then each token used to generate a signature MUST be covered by that signature. If
1312 the value is 'false', then the token MUST NOT be covered by the signature. Note that in cases where
1313 derived keys are used the 'main' token, and NOT the derived key token, is covered by the signature. It is
1314 recommended that assertions that define values for this property apply to [Endpoint Policy Subject]. The
1315 default value for this property is 'false'.

1316 6.6 [Entire Header and Body Signatures] Property

1317 This boolean property specifies whether signature digests over the SOAP body and SOAP headers must
1318 only cover the entire body and entire header elements. If the value is 'true', then each digest over the
1319 SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In
1320 addition each digest over a SOAP header MUST be over an actual header element and not a descendant
1321 of a header element. This restriction does not specifically apply to the wsse:Security header. However
1322 signature digests over child elements of the wsse:Security header MUST be over the entire child element
1323 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a
1324 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to
1325 'true' mitigates against some possible re-writing attacks. It is recommended that assertions that define
1326 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

1327 6.7 [Security Header Layout] Property

1328 This property indicates which layout rules to apply when adding items to the security header. The
1329 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.

1330

1331 6.7.1 Strict Layout Rules for WSS 1.0

- 1332 1. Tokens that are included in the message MUST be declared before use. For example:
- 1333 a. A local signing token MUST occur before the signature that uses it.
 - 1334 b. A local token serving as the source token for a derived key token MUST occur before that
 - 1335 derived key token.

- 1336 c. A local encryption token MUST occur before the reference list that points to
1337 xenc:EncryptedData elements that use it.
- 1338 d. If the same token is used for both signing and encryption, then it should appear before
1339 the ds:Signature and xenc:ReferenceList elements in the security header that are
1340 generated using the token.
- 1341 2. Signed elements inside the security header MUST occur before the signature that signs them.
1342 For example:
- 1343 a. A timestamp MUST occur before the signature that signs it.
- 1344 b. A Username token (usually in encrypted form) MUST occur before the signature that
1345 signs it.
- 1346 c. A primary signature MUST occur before the supporting token signature that signs the
1347 primary signature's signature value element.
- 1348 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1349 has the same order requirements as the source plain text element, unless requirement 4
1350 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1351 supporting token signature per 2.c above and an encrypted token has the same ordering
1352 requirements as the unencrypted token.
- 1353 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1354 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1355 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1356 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1357 Layout Rules for WSS 1.1
- 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 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1375 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1376 has the same order requirements as the source plain text element, unless requirement 4
1377 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1378 supporting token signature per 2.c above and an encrypted token has the same ordering
1379 requirements as the unencrypted token.
- 1380 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1381 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1382 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1383 However, the xenc:ReferenceList is not required to appear before independently encrypted
1384 tokens such as the xenc:EncryptedKey token as defined in WSS.

- 1385
1386
5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security 1.1](#)] MUST obey rule 1 above.

1387 7 Security Binding Assertions

1388 The appropriate representation of the different facets of security mechanisms requires distilling the
1389 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy
1390 scope of assertions defined in this section is the policy scope of their containing element.

1391 7.1 AlgorithmSuite Assertion

1392 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]
1393 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

1394 Syntax

```
1395 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1396   <wsp:Policy xmlns:wsp="...">  
1397     (<sp:Basic256 ... /> |  
1398     <sp:Basic192 ... /> |  
1399     <sp:Basic128 ... /> |  
1400     <sp:TripleDes ... /> |  
1401     <sp:Basic256Rsa15 ... /> |  
1402     <sp:Basic192Rsa15 ... /> |  
1403     <sp:Basic128Rsa15 ... /> |  
1404     <sp:TripleDesRsa15 ... /> |  
1405     <sp:Basic256Sha256 ... /> |  
1406     <sp:Basic192Sha256 ... /> |  
1407     <sp:Basic128Sha256 ... /> |  
1408     <sp:TripleDesSha256 ... /> |  
1409     <sp:Basic256Sha256Rsa15 ... /> |  
1410     <sp:Basic192Sha256Rsa15 ... /> |  
1411     <sp:Basic128Sha256Rsa15 ... /> |  
1412     <sp:TripleDesSha256Rsa15 ... /> |  
1413     ...)  
1414     <sp:InclusiveC14N ... /> ?  
1415     <sp:SOAPNormalization10 ... /> ?  
1416     <sp:STRTransform10 ... /> ?  
1417     (<sp:XPath10 ... /> |  
1418     <sp:XPathFilter20 ... /> |  
1419     <sp:AbsXPath ... /> |  
1420     ...)?  
1421     ...  
1422   </wsp:Policy>  
1423   ...  
1424 </sp:AlgorithmSuite>
```

1425
1426 The following describes the attributes and elements listed in the schema outlined above:

1427 /sp:AlgorithmSuite

1428 This identifies an AlgorithmSuite assertion.

1429 /sp:AlgorithmSuite/wsp:Policy

1430 This element contains one or more policy assertions that indicate the specific algorithm suite to use.

1431 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1432 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1433 to 'Basic256'.

1434 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

1435 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1436 to 'Basic192'.

1437 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1438 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1439 to 'Basic128'.

1440 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1441 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1442 to 'TripleDes'.

1443 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1444 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1445 to 'Basic256Rsa15'.

1446 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1447 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1448 to 'Basic192Rsa15'.

1449 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1450 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1451 to 'Basic128Rsa15'.

1452 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1453 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1454 to 'TripleDesRsa15'.

1455 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1456 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1457 to 'Basic256Sha256'.

1458 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1459 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1460 to 'Basic192Sha256'.

1461 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1462 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1463 to 'Basic128Sha256'.

1464 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1465 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1466 to 'TripleDesSha256'.

1467 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1468 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1469 to 'Basic256Sha256Rsa15'.

1470 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1471 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1472 to 'Basic192Sha256Rsa15'.

1473 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1474 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1475 to 'Basic128Sha256Rsa15'.

1476 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1477 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1478 to 'TripleDesSha256Rsa15'.

1479 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1480 This optional element is a policy assertion that indicates that the [C14N] property of an algorithm
1481 suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N] property is
1482 'ExcC14N'.

1483 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1484 This optional element is a policy assertion that indicates that the [SOAP Norm] property is set to
1485 'SNT'.

1486 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1487 This optional element is a policy assertion that indicates that the [STR Transform] property is set
1488 to 'STRT10'.

1489 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1490 This optional element is a policy assertion that indicates that the [XPath] property is set to 'XPath'.

1491 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1492 This optional element is a policy assertion that indicates that the [XPath] property is set to
1493 'XPath20'.

1494 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1495 This optional element is a policy assertion that indicates that the [XPath] property is set to
1496 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1497

1498 7.2 Layout Assertion

1499 This assertion indicates a requirement for a particular security header layout as defined under the
1500 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1501 containing assertion.

1502 Syntax

```
1503 <sp:Layout xmlns:sp="..." ... >  
1504   <wsp:Policy xmlns:wsp="...">  
1505     <sp:Strict ... /> |  
1506     <sp:Lax ... /> |  
1507     <sp:LaxTsFirst ... /> |  
1508     <sp:LaxTsLast ... /> |  
1509     ...  
1510   </wsp:Policy>  
1511   ...  
1512 </sp:Layout>
```

1513

1514 The following describes the attributes and elements listed in the schema outlined above:

1515 /sp:Layout

1516 This identifies a Layout assertion.

1517 /sp:Layout/wsp:Policy

1518 This element contains one or more policy assertions that indicate the specific security header layout
1519 to use.

1520 /sp:Layout/wsp:Policy/sp:Strict

1521 This optional element is a policy assertion that indicates that the [Security Header Layout]
1522 property is set to 'Strict'.

1523 /sp:Layout/wsp:Policy/sp:Lax

1524 This optional element is a policy assertion that indicates that the [Security Header Layout]
1525 property is set to 'Lax'.

1526 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1527 This optional element is a policy assertion that indicates that the [Security Header Layout]
1528 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1529 'true' by the presence of an sp:IncludeTimestamp assertion.

1530 /sp:Layout/wsp:Policy/sp:LaxTsLast

1531 This optional element is a policy assertion that indicates that the [Security Header Layout]
1532 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
1533 'true' by the presence of an sp:IncludeTimestamp assertion.

1534 7.3 TransportBinding Assertion

1535 The TransportBinding assertion is used in scenarios in which message protection and security correlation
1536 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
1537 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
1538 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
1539 MUST apply to [Endpoint Policy Subject].

1540 Syntax

```
1541 <sp:TransportBinding xmlns:sp="..." ... >  
1542   <wsp:Policy xmlns:wsp="...">  
1543     <sp:TransportToken ... >  
1544       <wsp:Policy> ... </wsp:Policy>  
1545     ...  
1546   </sp:TransportToken>  
1547   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1548   <sp:Layout ... > ... </sp:Layout> ?  
1549   <sp:IncludeTimestamp ... /> ?  
1550   ...  
1551 </wsp:Policy>  
1552   ...  
1553 </sp:TransportBinding>
```

1554
1555 The following describes the attributes and elements listed in the schema outlined above:

1556 /sp:TransportBinding

1557 This identifies a TransportBinding assertion.

1558 /sp:TransportBinding/wsp:Policy

1559 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
1560 assertion.

1561 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1562 This required element is a policy assertion that indicates a requirement for a Transport Token.
1563 The specified token populates the [Transport Token] property and indicates how the transport is
1564 secured.

1565 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1566 This indicates a nested policy that identifies the type of Transport Token to use.

1567 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

1568 This required element is a policy assertion that indicates a value that populates the [Algorithm
1569 Suite] property. See Section 6.1 for more details.

1570 /sp:TransportBinding/wsp:Policy/sp:Layout

1571 This optional element is a policy assertion that indicates a value that populates the [Security
1572 Header Layout] property. See Section 6.7 for more details.

1573 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

1574 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1575 'true'.

1576 7.4 SymmetricBinding Assertion

1577 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
1578 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
1579 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
1580 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
1581 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
1582 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
1583 properties and is used as the basis for both encryption and signature in both directions. This assertion
1584 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

1585 Syntax

```
1586 <sp:SymmetricBinding xmlns:sp="..." ... >  
1587   <wsp:Policy xmlns:wsp="...">  
1588     (  
1589       <sp:EncryptionToken ... >  
1590         <wsp:Policy> ... </wsp:Policy>  
1591       </sp:EncryptionToken>  
1592       <sp:SignatureToken ... >  
1593         <wsp:Policy> ... </wsp:Policy>  
1594       </sp:SignatureToken>  
1595     ) | (  
1596       <sp:ProtectionToken ... >  
1597         <wsp:Policy> ... </wsp:Policy>  
1598       </sp:ProtectionToken>  
1599     )  
1600   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1601   <sp:Layout ... > ... </sp:Layout> ?  
1602   <sp:IncludeTimestamp ... /> ?  
1603   <sp:EncryptBeforeSigning ... /> ?  
1604   <sp:EncryptSignature ... /> ?  
1605   <sp:ProtectTokens ... /> ?  
1606   <sp:OnlySignEntireHeadersAndBody ... /> ?  
1607   ...  
1608 </wsp:Policy>  
1609   ...  
1610 </sp:SymmetricBinding>
```

1611
1612 The following describes the attributes and elements listed in the schema outlined above:

1613 /sp:SymmetricBinding

1614 This identifies a SymmetricBinding assertion.

1615 /sp:SymmetricBinding/wsp:Policy

1616 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
1617 assertion.

1618 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken

1619 This optional element is a policy assertion that indicates a requirement for an Encryption Token.
1620 The specified token populates the [Encryption Token] property and is used for encryption. It is an
1621 error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

1622 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
1623 The policy contained here MUST identify exactly one token to use for encryption.

1624 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
1625 This optional element is a policy assertion that indicates a requirement for a Signature Token.
1626 The specified token populates the [Signature Token] property and is used for the message
1627 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
1628 specified.

1629 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
1630 The policy contained here MUST identify exactly one token to use for signatures.

1631 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
1632 This optional element is a policy assertion that indicates a requirement for a Protection Token.
1633 The specified token populates the [Encryption Token] and [Signature Token properties] and is
1634 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
1635 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
1636 specified.

1637 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
1638 The policy contained here MUST identify exactly one token to use for protection.

1639 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
1640 This required element is a policy assertion that indicates a value that populates the [Algorithm
1641 Suite] property. See Section 6.1 for more details.

1642 /sp:SymmetricBinding/wsp:Policy/sp:Layout
1643 This optional element is a policy assertion that indicates a value that populates the [Security
1644 Header Layout] property. See Section 6.7 for more details.

1645 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
1646 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1647 'true'.

1648 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
1649 This optional element is a policy assertion that indicates that the [Protection Order] property is set
1650 to 'EncryptBeforeSigning'.

1651 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
1652 This optional element is a policy assertion that indicates that the [Signature Protection] property is
1653 set to 'true'.

1654 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
1655 This optional element is a policy assertion that indicates that the [Token Protection] property is
1656 set to 'true'.

1657 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
1658 This optional element is a policy assertion that indicates that the [Entire Header And Body
1659 Signatures] property is set to 'true'.

1660 7.5 AsymmetricBinding Assertion

1661 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means
1662 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly
1663 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and
1664 signature. However it is also common practice to use distinct keys for encryption and signature, because
1665 of their different lifecycles.

1666
1667 This binding enables either of these practices by means of four binding specific token properties: [Initiator
1668 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption
1669 Token].

1670
1671 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator
1672 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient
1673 Encryption Token] will both refer to the same token.

1674
1675 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator
1676 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient
1677 Encryption Token] will refer to different tokens.

1678
1679 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the
1680 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response
1681 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the
1682 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for
1683 the message encryption from initiator to the recipient. Note that in each case, the token is associated with
1684 the party (initiator or recipient) who knows the secret.

1685 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy
1686 Subject].

1687 Syntax

```
1688 <sp:AsymmetricBinding xmlns:sp="..." ... >  
1689   <wsp:Policy xmlns:wsp="...">  
1690     (  
1691       <sp:InitiatorToken>  
1692         <wsp:Policy> ... </wsp:Policy>  
1693       </sp:InitiatorToken>  
1694     ) | (  
1695       <sp:InitiatorSignatureToken>  
1696         <wsp:Policy> ... </wsp:Policy>  
1697       </sp:InitiatorSignatureToken>  
1698       <sp:InitiatorEncryptionToken>  
1699         <wsp:Policy> ... </wsp:Policy>  
1700       </sp:InitiatorEncryptionToken>  
1701     )  
1702   ) | (  
1703     <sp:RecipientToken>  
1704       <wsp:Policy> ... </wsp:Policy>  
1705     </sp:RecipientToken>  
1706   ) | (  
1707     <sp:RecipientSignatureToken>  
1708       <wsp:Policy> ... </wsp:Policy>  
1709     </sp:RecipientSignatureToken>  
1710     <sp:RecipientEncryptionToken>  
1711       <wsp:Policy> ... </wsp:Policy>
```

```

1712     </sp:RecipientEncryptionToken>
1713   )
1714   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1715   <sp:Layout ... > ... </sp:Layout> ?
1716   <sp:IncludeTimestamp ... /> ?
1717   <sp:EncryptBeforeSigning ... /> ?
1718   <sp:EncryptSignature ... /> ?
1719   <sp:ProtectTokens ... /> ?
1720   <sp:OnlySignEntireHeadersAndBody ... /> ?
1721   ...
1722 </wsp:Policy>
1723   ...
1724 </sp:AsymmetricBinding>

```

1725

1726 The following describes the attributes and elements listed in the schema outlined above:

1727 /sp:AsymmetricBinding

1728 This identifies a AsymmetricBinding assertion.

1729 /sp:AsymmetricBinding/wsp:Policy

1730 This indicates a nested `wsp:Policy` element that defines the behavior of the AsymmetricBinding
 1731 assertion.

1732 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

1733 This optional element is a policy assertion that indicates a requirement for an Initiator Token. The
 1734 specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
 1735 properties and is used for the message signature from initiator to recipient, and encryption from
 1736 recipient to initiator.

1737 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

1738 The policy contained here MUST identify one or more token assertions.

1739 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

1740 This optional element is a policy assertion that indicates a requirement for an Initiator Signature
 1741 Token. The specified token populates the [Initiator Signature Token] property and is used for the
 1742 message signature from initiator to recipient.

1743 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

1744 The policy contained here MUST identify one or more token assertions.

1745 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

1746 This optional element is a policy assertion that indicates a requirement for an Initiator Encryption
 1747 Token. The specified token populates the [Initiator Encryption Token] property and is used for the
 1748 message encryption from recipient to initiator.

1749 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

1750 The policy contained here MUST identify one or more token assertions.

1751 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

1752 This optional element is a policy assertion that indicates a requirement for a Recipient Token. The
 1753 specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
 1754 property and is used for encryption from initiator to recipient, and for the message signature from
 1755 recipient to initiator.

1756 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

1757 The policy contained here MUST identify one or more token assertions.

1758 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

- 1759 This optional element is a policy assertion that indicates a requirement for a Recipient Signature
1760 Token. The specified token populates the [Recipient Signature Token] property and is used for
1761 the message signature from Recipient to recipient.
- 1762 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
1763 The policy contained here MUST identify one or more token assertions.
- 1764 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
1765 This optional element is a policy assertion that indicates a requirement for a Recipient Encryption
1766 Token. The specified token populates the [Recipient Encryption Token] property and is used for
1767 the message encryption from recipient to Recipient.
- 1768 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
1769 The policy contained here MUST identify one or more token assertions.
- 1770 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
1771 This required element is a policy assertion that indicates a value that populates the [Algorithm
1772 Suite] property. See Section 6.1 for more details.
- 1773 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
1774 This optional element is a policy assertion that indicates a value that populates the [Security
1775 Header Layout] property. See Section 6.7 for more details.
- 1776 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
1777 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1778 'true'.
- 1779 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
1780 This optional element is a policy assertion that indicates that the [Protection Order] property is set
1781 to 'EncryptBeforeSigning'.
- 1782 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
1783 This optional element is a policy assertion that indicates that the [Signature Protection] property is
1784 set to 'true'.
- 1785 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
1786 This optional element is a policy assertion that indicates that the [Token Protection] property is
1787 set to 'true'.
- 1788 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
1789 This optional element is a policy assertion that indicates that the [Entire Header And Body
1790 Signatures] property is set to 'true'.

1791

8 Supporting Tokens

1792 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to
1793 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore
1794 be referred to as the “message signature”. Additional tokens may be specified to augment the claims
1795 provided by the token associated with the “message signature” provided by the Security Binding. This
1796 section defines seven properties related to supporting token requirements which may be referenced by a
1797 Security Binding: [Supporting Tokens], [Signed Supporting Tokens], [Endorsing Supporting Tokens],
1798 [Signed Endorsing Supporting Tokens], [Signed Encrypted Supporting Tokens], [Endorsing Encrypted
1799 Supporting Tokens] and [Signed Endorsing Encrypted Supporting Tokens]. Seven assertions are defined
1800 to populate those properties: SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,
1801 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,
1802 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These
1803 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy
1804 Subject] or [Operation Policy Subject].

1805

1806 Supporting tokens may be specified at a different scope than the binding assertion which provides
1807 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while
1808 the supporting tokens might be defined at the scope of a message. When assertions that populate this
1809 property are defined in overlapping scopes, the sender should merge the requirements by including all
1810 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

1811

1812 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the
1813 tokens should sign and encrypt the various message parts. In such cases ordering of elements (tokens,
1814 signatures, reference lists etc.) in the security header would be used to determine which order signature
1815 and encryptions occurred in.

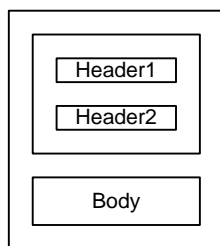
1816

1817 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional
1818 constraints defined by the supporting token assertion. For example, if the supporting token assertion
1819 specifies message parts that need to be encrypted, the specified tokens need to be capable of
1820 encryption.

1821

1822 To illustrate the different ways that supporting tokens may be bound to the message, let’s consider a
1823 message with three components: Header1, Header2, and Body.

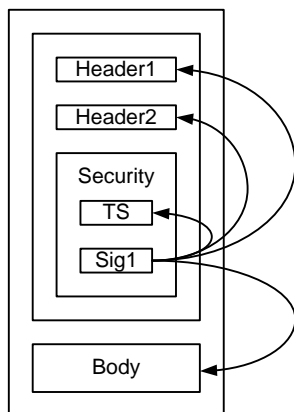
1824



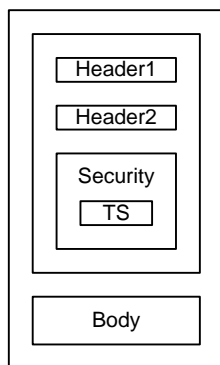
1825

1826 Even before any supporting tokens are added, each binding requires that the message is signed using a
1827 token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts

1828 of the message including the message timestamp (TS) facilitate replay detection. The signature is then
1829 included as part of the Security header as illustrated below:
1830



1831
1832 Note: if required, the initiator may also include in the Security header the token used as the basis for the
1833 message signature (Sig1), not shown in the diagram.
1834 If transport security is used, only the message timestamp (TS) is included in the Security header as
1835 illustrated below:



1836

1837 8.1 SupportingTokens Assertion

1838 Supporting tokens are included in the security header and may optionally include additional message
1839 parts to sign and/or encrypt.

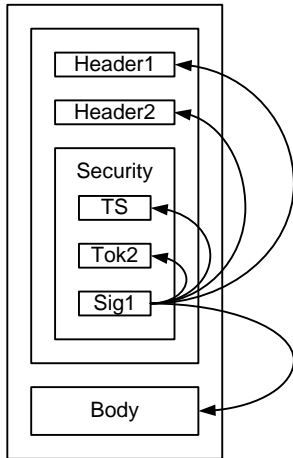
1840 Syntax

```
1841 <sp:SupportingTokens xmlns:sp="..." ... >  
1842   <wsp:Policy xmlns:wsp="...">  
1843     [Token Assertion]+  
1844     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?  
1845     (  
1846       <sp:SignedParts ... > ... </sp:SignedParts> |  
1847       <sp:SignedElements ... > ... </sp:SignedElements> |  
1848       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |  
1849       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |  
1850     ) *  
1851     ...  
1852   </wsp:Policy>  
1853   ...  
1854 </sp:SupportingTokens>
```

1856 The following describes the attributes and elements listed in the schema outlined above:
1857 /sp:SupportingTokens
1858 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting Tokens]
1859 property.
1860 /sp:SupportingTokens/wsp:Policy
1861 This describes additional requirements for satisfying the SupportingTokens assertion.
1862 /sp:SupportingTokens/wsp:Policy/[Token Assertion]
1863 The policy MUST identify one or more token assertions.
1864 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite
1865 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
1866 describes the algorithms to use for cryptographic operations performed with the tokens identified
1867 by this policy assertion.
1868 /sp:SupportingTokens/wsp:Policy/sp:SignedParts
1869 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
1870 describes additional message parts that MUST be included in the signature generated with the
1871 token identified by this policy assertion.
1872 /sp:SupportingTokens/wsp:Policy/sp:SignedElements
1873 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
1874 describes additional message elements that MUST be included in the signature generated with
1875 the token identified by this policy assertion.
1876 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts
1877 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
1878 describes additional message parts that MUST be encrypted using the token identified by this
1879 policy assertion.
1880 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements
1881 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
1882 describes additional message elements that MUST be encrypted using the token identified by this
1883 policy assertion.

1884 **8.2 SignedSupportingTokens Assertion**

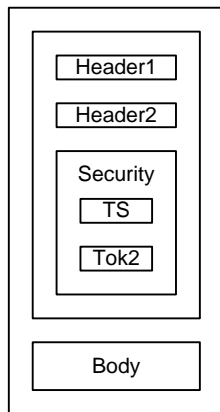
1885 Signed tokens are included in the “message signature” as defined above and may optionally include
1886 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
1887 (Tok2) is signed by the message signature (Sig1):
1888



1889

1890 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

1891



1892

1893 **Syntax**

```

1894 <sp:SignedSupportingTokens xmlns:sp="..." ... >
1895   <wsp:Policy xmlns:wsp="...">
1896     [Token Assertion]+
1897     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
1898     (
1899       <sp:SignedParts ... > ... </sp:SignedParts> |
1900       <sp:SignedElements ... > ... </sp:SignedElements> |
1901       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
1902       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
1903     ) *
1904     ...
1905   </wsp:Policy>
1906   ...
1907 </sp:SignedSupportingTokens>

```

1908

1909 The following describes the attributes and elements listed in the schema outlined above:

1910 /sp:SignedSupportingTokens

1911 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed Supporting Tokens] property.

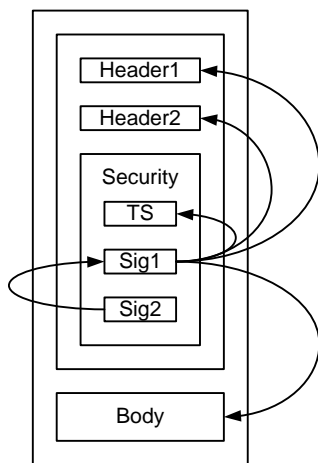
1913 /sp:SignedSupportingTokens/wsp:Policy

1914 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

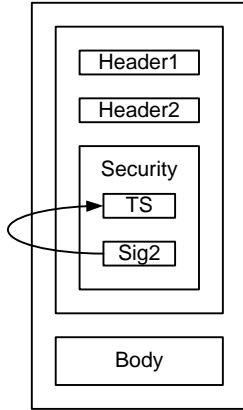
- 1915 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 1916 The policy MUST identify one or more token assertions.
- 1917 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 1918 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
 1919 describes the algorithms to use for cryptographic operations performed with the tokens identified
 1920 by this policy assertion.
- 1921 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 1922 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
 1923 describes additional message parts that MUST be included in the signature generated with the
 1924 token identified by this policy assertion.
- 1925 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 1926 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
 1927 describes additional message elements that MUST be included in the signature generated with
 1928 the token identified by this policy assertion.
- 1929 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 1930 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
 1931 describes additional message parts that MUST be encrypted using the token identified by this
 1932 policy assertion.
- 1933 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 1934 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
 1935 describes additional message elements that MUST be encrypted using the token identified by this
 1936 policy assertion.

1937 8.3 EndorsingSupportingTokens Assertion

1938 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 1939 produced from the message signature and may optionally include additional message parts to sign and/or
 1940 encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message signature
 1941 (Sig1):
 1942



1943
 1944 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 1945 below:
 1946



1947

1948 **Syntax**

```

1949 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
1950   <wsp:Policy xmlns:wsp="...">
1951     [Token Assertion]+
1952     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
1953     (
1954       <sp:SignedParts ... > ... </sp:SignedParts> |
1955       <sp:SignedElements ... > ... </sp:SignedElements> |
1956       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
1957       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
1958     ) *
1959     ...
1960   </wsp:Policy>
1961   ...
1962 </sp:EndorsingSupportingTokens>

```

1963

1964 The following describes the attributes and elements listed in the schema outlined above:

1965 /sp:EndorsingSupportingTokens

1966 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
1967 [Endorsing Supporting Tokens] property.

1968 /sp:EndorsingSupportingTokens/wsp:Policy

1969 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

1970 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

1971 The policy MUST identify one or more token assertions.

1972 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

1973 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
1974 describes the algorithms to use for cryptographic operations performed with the tokens identified
1975 by this policy assertion.

1976 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

1977 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
1978 describes additional message parts that MUST be included in the signature generated with the
1979 token identified by this policy assertion.

1980 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

1981 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
1982 describes additional message elements that MUST be included in the signature generated with
1983 the token identified by this policy assertion.

1984 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

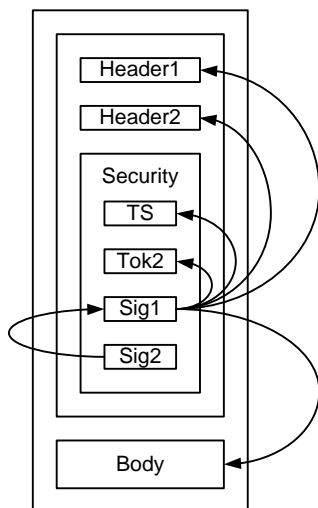
1985 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
1986 describes additional message parts that MUST be encrypted using the token identified by this
1987 policy assertion.

1988 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

1989 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
1990 describes additional message elements that MUST be encrypted using the token identified by this
1991 policy assertion.

1992 8.4 SignedEndorsingSupportingTokens Assertion

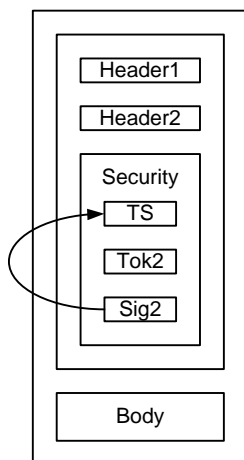
1993 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
1994 and are themselves signed by that message signature, that is both tokens (the token used for the
1995 message signature and the signed endorsing token) sign each other. This assertion may optionally
1996 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
1997 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
1998 message signature (Sig1):
1999



2000

2001 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
2002 should cover the message timestamp as illustrated below:

2003



2004

2005 **Syntax**

2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019

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

2020

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

2022

/sp:SignedEndorsingSupportingTokens

2023
2024

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

2025

/sp:SignedEndorsingSupportingTokens/wsp:Policy

2026

This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2027

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

2028

The policy MUST identify one or more token assertions.

2029

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

2030
2031
2032

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

2033

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

2034
2035
2036

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

2037

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

2038
2039
2040

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

2041

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

2042
2043
2044

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

2045

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

2046
2047
2048

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

2049 **8.5 SignedEncryptedSupportingTokens Assertion**

2050 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also
2051 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2052 encrypting the supporting tokens.

2053 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of
2054 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2055 sp:SignedSupportingTokens assertion.

2056 **8.6 EndorsingEncryptedSupportingTokens Assertion**

2057 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also
2058 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2059 encrypting the supporting tokens.

2060 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of
2061 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2062 sp:EndorsingSupportingTokens assertion.

2063 **8.7 SignedEndorsingEncryptedSupportingTokens Assertion**

2064 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section
2065 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD
2066 be used for encrypting the supporting tokens.

2067 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of
2068 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per
2069 the sp:SignedEndorsingSupportingTokens assertion.

2070 **8.8 Interaction between [Token Protection] property and supporting 2071 token assertions**

2072 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that
2073 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2074 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or
2075 the [Signature Token] in the Symmetric binding case, covers that token.
- 2076 • Endorsing signatures cover the main signature and the endorsing token.
- 2077 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the
2078 message signature and once by the endorsing signature.

2079 In addition, signed supporting tokens are covered by the message signature, although this is independent
2080 of [Token Protection].

2081 **8.9 Example**

2082 Example policy containing supporting token assertions:

2083

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

```

2084 <wsp:Policy xmlns:wsp="...">
2085   <sp:SymmetricBinding xmlns:sp="...">
2086     <wsp:Policy>
2087       <sp:ProtectionToken>
2088         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2089           <sp:Issuer>...</sp:Issuer>
2090           <sp:RequestSecurityTokenTemplate>
2091             ...
2092           </sp:RequestSecurityTokenTemplate>
2093         </sp:IssuedToken>
2094       </sp:ProtectionToken>
2095       <sp:AlgorithmSuite>
2096         <wsp:Policy>
2097           <sp:Basic256 />
2098         </wsp:Policy>
2099       </sp:AlgorithmSuite>
2100       ...
2101     </wsp:Policy>
2102   </sp:SymmetricBinding>
2103   ...
2104   <sp:SignedSupportingTokens>
2105     <wsp:Policy>
2106       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2107     </wsp:Policy>
2108   </sp:SignedSupportingTokens>
2109   <sp:SignedEndorsingSupportingTokens>
2110     <wsp:Policy>
2111       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2112         <wsp:Policy>
2113           <sp:WssX509v3Token10 />
2114         </wsp:Policy>
2115       </sp:X509Token>
2116     </wsp:Policy>
2117   </sp:SignedEndorsingSupportingTokens>
2118   ...
2119 </wsp:Policy>

```

2120 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be
2121 included in the security header and covered by the message signature. The
2122 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the
2123 security header and covered by the message signature. In addition, a signature over the message
2124 signature based on the key material associated with the X509 certificate must be included in the security
2125 header.

2126 9 WSS: SOAP Message Security Options

2127 There are several optional aspects to the WSS: SOAP Message Security specification that are
2128 independent of the trust and token taxonomies. This section describes another class of properties and
2129 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The
2130 assertions defined here MUST apply to [Endpoint Policy Subject].

2131 The properties and assertions dealing with token references defined in this section indicate whether the
2132 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and
2133 recipient MAY send a fault if such references are encountered.

2134

2135 Note: This approach is chosen because:

2136 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used
2137 in a single reference.

2138 B) In a multi-message exchange, a token may be referenced using different mechanisms depending
2139 on which of a series of messages is being secured.

2140

2141 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a
2142 `wsse:InvalidSecurity` fault.

2143

2144 **WSS: SOAP Message Security 1.0 Properties**

2145 **[Direct References]**

2146 This property indicates whether the initiator and recipient MUST be able to process direct token
2147 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations
2148 MUST be able to process such references.

2149

2150 **[Key Identifier References]**

2151 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific
2152 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to
2153 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST
2154 NOT generate such references and that the initiator and recipient MAY send a fault if such references are
2155 encountered. This property has a default value of 'false'.

2156

2157 **[Issuer Serial References]**

2158 This boolean property indicates whether the initiator and recipient MUST be able to process references
2159 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST
2160 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT
2161 generate such references and that the initiator and recipient MAY send a fault if such references are
2162 encountered. This property has a default value of 'false'.

2163

2164 **[External URI References]**

2165 This boolean property indicates whether the initiator and recipient MUST be able to process references to
2166 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST
2167 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2168 generate such references and that the initiator and recipient MAY send a fault if such references are
2169 encountered. This property has a default value of 'false'.

2170 **[Embedded Token References]**

2171 This boolean property indicates whether the initiator and recipient MUST be able to process references
2172 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to
2173 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2174 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2175 This property has a default value of 'false'.

2176

2177 **WSS: SOAP Message Security 1.1 Properties**

2178 **[Thumbprint References]**

2179 This boolean property indicates whether the initiator and recipient MUST be able to process references
2180 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to
2181 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2182 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2183 This property has a default value of 'false'.

2184

2185 **[EncryptedKey References]**

2186 This boolean property indicates whether the initiator and recipient MUST be able to process references
2187 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to
2188 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2189 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2190 This property has a default value of 'false'.

2191

2192 **[Signature Confirmation]**

2193 This boolean property specifies whether `wss11:SignatureConfirmation` elements should be used
2194 as defined in WSS: Soap Message Security 1.1. If the value is 'true',
2195 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If
2196 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property
2197 applies to all signatures that are included in the security header. This property has a default value of
2198 'false'.

2199 **9.1 Wss10 Assertion**

2200 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are
2201 supported.

2202 **Syntax**

```
2203 <sp:Wss10 xmlns:sp="..." ... >  
2204   <wsp:Policy xmlns:wsp="...">  
2205     <sp:MustSupportRefKeyIdentifier ... /> ?  
2206     <sp:MustSupportRefIssuerSerial ... /> ?  
2207     <sp:MustSupportRefExternalURI ... /> ?  
2208     <sp:MustSupportRefEmbeddedToken ... /> ?  
2209     ...  
2210   </wsp:Policy>  
2211   ...  
2212 </sp:Wss10>
```

2213

2214 The following describes the attributes and elements listed in the schema outlined above:

2215 /sp:Wss10
 2216 This identifies a WSS10 assertion.
 2217 /sp:Wss10/wsp:Policy
 2218
 2219 This indicates a policy that controls WSS: SOAP Message Security 1.0
 2220 options./sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2221 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2222 is set to 'true'.
 2223 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial
 2224 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2225 set to 'true'.
 2226 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI
 2227 This optional element is a policy assertion indicates that the [External URI References] property is
 2228 set to 'true'.
 2229 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken
 2230 This optional element is a policy assertion indicates that the [Embedded Token References]
 2231 property is set to 'true'.

2232 9.2 Wss11 Assertion

2233 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
 2234 supported.

2235 Syntax

```

2236 <sp:Wss11 xmlns:sp="..." ... >
2237   <wsp:Policy xmlns:wsp="...">
2238     <sp:MustSupportRefKeyIdentifier ... /> ?
2239     <sp:MustSupportRefIssuerSerial ... /> ?
2240     <sp:MustSupportRefExternalURI ... /> ?
2241     <sp:MustSupportRefEmbeddedToken ... /> ?
2242     <sp:MustSupportRefThumbprint ... /> ?
2243     <sp:MustSupportRefEncryptedKey ... /> ?
2244     <sp:RequireSignatureConfirmation ... /> ?
2245     ...
2246   </wsp:Policy>
2247 </sp:Wss11>
  
```

2248
 2249 The following describes the attributes and elements listed in the schema outlined above:

2250 /sp:Wss11
 2251 This identifies an WSS11 assertion.
 2252 /sp:Wss11/wsp:Policy
 2253 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.
 2254 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2255 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2256 is set to 'true'.
 2257 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial
 2258 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2259 set to 'true'.

- 2260 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2261 This optional element is a policy assertion indicates that the [External URI References] property is
2262 set to 'true'.
- 2263 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2264 This optional element is a policy assertion indicates that the [Embedded Token References]
2265 property is set to 'true'.
- 2266 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2267 This optional element is a policy assertion indicates that the [Thumbprint References] property is
2268 set to 'true'.
- 2269 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2270 This optional element is a policy assertion indicates that the [EncryptedKey References] property
2271 is set to 'true'.
- 2272 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2273 This optional element is a policy assertion indicates that the [Signature Confirmation] property is
2274 set to 'true'.

2275 10 WS-Trust Options

2276 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically
2277 with client and server challenges and entropy behaviors. These assertions relate to interactions with a
2278 Security Token Service and may augment the behaviors defined by the Binding Property Assertions
2279 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2280

2281 **WS-Trust 1.0 Properties**

2282 **[Client Challenge]**

2283 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a
2284 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of
2285 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of
2286 messages exchanged by the client and service in satisfying the RST. This property has a default value of
2287 'false'.

2288

2289 **[Server Challenge]**

2290 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a
2291 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of
2292 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server may
2293 increase the number of messages exchanged by the client and service in order to accommodate the
2294 `wst:SignChallengeResponse` element sent by the client to the server in response to the
2295 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the
2296 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2297

2298 **[Client Entropy]**

2299 This boolean property indicates whether client entropy is required to be used as key material for a
2300 requested proof token. A value of 'true' indicates that client entropy is required. A value of 'false' indicates
2301 that client entropy is not required. This property has a default value of 'false'.

2302

2303 **[Server Entropy]**

2304 This boolean property indicates whether server entropy is required to be used as key material for a
2305 requested proof token. A value of 'true' indicates that server entropy is required. A value of 'false'
2306 indicates that server entropy is not required. This property has a default value of 'false'.

2307 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy
2308 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm
2309 Suite] property.

2310

2311 **[Issued Tokens]**

2312 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in
2313 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'
2314 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of
2315 'false'.

2316 **[Collection]**

2317 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A
2318 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and
2319 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that
2320 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default
2321 value of 'false'.
2322

2323 10.1 Trust10 Assertion

2324 The Trust10 assertion allows you to specify which WS-Trust 1.0 options are supported.

2325 Syntax

```
2326 <sp:Trust10 xmlns:sp="..." ... >  
2327   <wsp:Policy xmlns:wsp="...">  
2328     <sp:MustSupportClientChallenge ... />?  
2329     <sp:MustSupportServerChallenge ... />?  
2330     <sp:RequireClientEntropy ... />?  
2331     <sp:RequireServerEntropy ... />?  
2332     <sp:MustSupportIssuedTokens ... />?  
2333     <sp:RequireRequestSecurityTokenCollection />?  
2334     ...  
2335   </wsp:Policy>  
2336   ...  
2337 </sp:Trust10 ... >
```

2338
2339 The following describes the attributes and elements listed in the schema outlined above:

2340 /sp:Trust10

2341 This identifies a Trust10 assertion.

2342 /sp:Trust10/wsp:Policy

2343 This indicates a policy that controls WS-Trust 1.0 options.

2344 /sp:Trust10/wsp:Policy/sp:MustSupportClientChallenge

2345 This optional element is a policy assertion indicates that the [Client Challenge] property is set to
2346 'true'.

2347 /sp:Trust10/wsp:Policy/sp:MustSupportServerChallenge

2348 This optional element is a policy assertion indicates that the [Server Challenge] property is set to
2349 'true'.

2350 /sp:Trust10/wsp:Policy/sp:RequireClientEntropy

2351 This optional element is a policy assertion indicates that the [Client Entropy] property is set to
2352 'true'.

2353 /sp:Trust10/wsp:Policy/sp:RequireServerEntropy

2354 This optional element is a policy assertion indicates that the [Server Entropy] property is set to
2355 'true'.

2356 /sp:Trust10/wsp:Policy/sp:MustSupportIssuedTokens

2357 This optional element is a policy assertion indicates that the [Issued Tokens] property is set to
2358 'true'.

2359 /sp:Trust10/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2360 This optional element is a policy assertion that indicates that the [Collection] property is set to
2361 'true'.

2362 11 Guidance on creating new assertions and assertion 2363 extensibility

2364 This non-normative appendix provides guidance for designers of new assertions intended for use with this
2365 specification.

2366 11.1 General Design Points

- 2367 • Prefer Distinct QNames
- 2368 • Parameterize using nested policy where possible.
- 2369 • Parameterize using attributes and/or child elements where necessary.

2370 11.2 Detailed Design Guidance

2371 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per
2372 WS-Policy is performed by matching element QNames. Matching does not take into account attributes
2373 that are present on the assertion element. Nor does it take into account child elements except for
2374 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions
2375 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2376
2377 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken
2378 into account. In general, multiple assertions with distinct QNames are preferably to a single assertion that
2379 uses attributes and/or content to distinguish different cases. For example, given two possible assertion
2380 designs;

```
2381  
2382 Design 1  
2383  
2384 <A1/>  
2385 <A2/>  
2386 <A3/>  
2387  
2388 Design 2.  
2389  
2390 <A Parameter='1' />  
2391 <A Parameter='2' />  
2392 <A Parameter='3' />  
2393
```

2394 then design 1. would generally be preferred because it allows the policy matching logic to provide more
2395 accurate matches between policies.

2396
2397 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct
2398 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These
2399 distinct token assertions make policy matching much more useful as less false positives are generated
2400 when performing policy matching.

2401
2402 There are cases where using attributes or child elements as parameters in assertion design is
2403 reasonable. Examples include cases when implementations are expected to understand all the values for
2404 a given parameter and when encoding the parameter information into the assertion QName would result
2405 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2406 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2407 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2408 was encoded into the assertion QNames, each existing token assertion would require five variants, one
2409 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2410
2411 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2412 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2413 assertion is parameterized by the nested token version assertions. Policy matching can use these
2414 parameters to find matches between policies where the broad token type is support by both parties but
2415 they might not support the same specific versions.

2416
2417 Note, when designing assertions for new token types such assertions SHOULD allow the
2418 sp:IncludeToken attribute and SHOULD allow nested policy.

2419

2420 **12 Security Considerations**

2421 It is strongly recommended that policies and assertions be signed to prevent tampering.

2422 It is recommended that policies should not be accepted unless they are signed and have an associated
2423 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely
2424 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that
2425 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2426
2427 It should be noted that the mechanisms described in this document could be secured as part of a SOAP
2428 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using
2429 object-specific security mechanisms.

2430
2431 It is recommended that policies not specify two (or more) SignedSupportingTokens or
2432 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are
2433 subject to modification which may be undetectable.

2434
2435 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest
2436 of the policy in order to combat certain XML substitution attacks.

2437 **A. Assertions and WS-PolicyAttachment**

2438 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per
2439 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical
2440 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any
2441 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy
2442 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

2443 **A.1 Endpoint Policy Subject Assertions**

2444 **A.1.1 Security Binding Assertions**

- 2445 [TransportBinding Assertion](#) (Section 7.3)
- 2446 [SymmetricBinding Assertion](#) (Section 7.4)
- 2447 [AsymmetricBinding Assertion](#) (Section 7.5)

2448 **A.1.2 Token Assertions**

- 2449 [SupportingTokens Assertion](#) (Section 8.1)
- 2450 [SignedSupportingTokens Assertion](#) (Section 8.2)
- 2451 [EndorsingSupportingTokens Assertion](#) (Section 8.3)
- 2452 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)
- 2453 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)
- 2454 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)
- 2455 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

2456 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

- 2457 [Wss10 Assertion](#) (Section 9.1)

2458 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

- 2459 [Wss11 Assertion](#) (Section 9.2)

2460 **A.1.5 Trust 1.0 Assertions**

- 2461 [Trust10 Assertion](#) (Section 10.1)

2462 **A.2 Operation Policy Subject Assertions**

2463 **A.2.1 Security Binding Assertions**

- 2464 [SymmetricBinding Assertion](#) (Section 7.4)
- 2465 [AsymmetricBinding Assertion](#) (Section 7.5)

2466 **A.2.2 Supporting Token Assertions**

- 2467 [SupportingTokens Assertion](#) (Section 8.1)
- 2468 [SignedSupportingTokens Assertion](#) (Section 8.2)

2469	EndorsingSupportingTokens Assertion	(Section 8.3)
2470	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2471	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2472	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2473	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2474 **A.3 Message Policy Subject Assertions**

2475 **A.3.1 Supporting Token Assertions**

2476	SupportingTokens Assertion	(Section 8.1)
2477	SignedSupportingTokens Assertion	(Section 8.2)
2478	EndorsingSupportingTokens Assertion	(Section 8.3)
2479	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2480	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2481	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2482	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2483 **A.3.2 Protection Assertions**

2484	SignedParts Assertion	(Section 4.1.1)
2485	SignedElements Assertion	(Section 4.1.2)
2486	EncryptedParts Assertion	(Section 4.2.1)
2487	EncryptedElements Assertion	(Section 4.2.2)
2488	ContentEncryptedElements Assertion	(Section 4.2.3)
2489	RequiredElements Assertion	(Section 4.3.1)
2490	RequiredParts Assertion	(Section 4.3.2)

2491 **A.4 Assertions With Undefined Policy Subject**

2492 The assertions listed in this section do not have a defined policy subject because they appear nested
 2493 inside some other assertion which does have a defined policy subject. This list is derived from nested
 2494 assertions in the specification that have independent sections. It is not a complete list of nested
 2495 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
 2496 additional nested assertions.

2497 **A.4.1 General Assertions**

2498	AlgorithmSuite Assertion	(Section 7.1)
2499	Layout Assertion	(Section 7.2)

2500 **A.4.2 Token Usage Assertions**

2501 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
 2502 assertions.

2503 **A.4.3 Token Assertions**

2504	UsernameToken Assertion	(Section 5.3.1)
------	---	-----------------

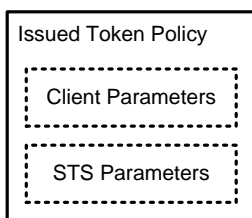
2505	IssuedToken Assertion	(Section 5.3.2)
2506	X509Token Assertion	(Section 5.3.3)
2507	KerberosToken Assertion	(Section 5.3.4)
2508	SpnegoContextToken Assertion	(Section 5.3.5)
2509	SecurityContextToken Assertion	(Section 5.3.6)
2510	SecureConversationToken Assertion	(Section 5.3.7)
2511	SamlToken Assertion	(Section 5.3.8)
2512	RelToken Assertion	(Section 5.3.9)
2513	HttpsToken Assertion	(Section 5.3.10)

2514 B. Issued Token Policy

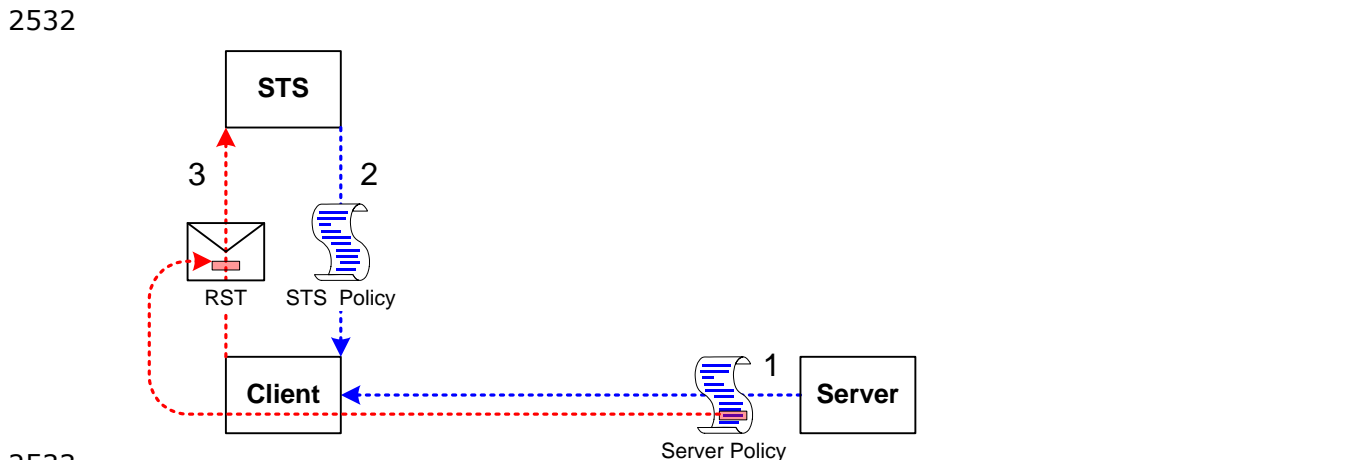
2515 The section provides further detail about behavior associated with the IssuedToken assertion in section
2516 5.3.2.

2517
2518 The issued token security model involves a three-party setup. There's a target Server, a Client, and a
2519 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from
2520 STS to Client. Policy may be embedded inside an Issued Token assertion, or acquired out-of-band. There
2521 may be an explicit trust relationship between the Server and the STS. There must be a trust relationship
2522 between the Client and the STS.

2523
2524 The Issued Token policy assertion includes two parts: 1) client-specific parameters that must be
2525 understood and processed by the client and 2) STS specific parameters which are to be processed by the
2526 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



2527
2528 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server
2529 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are
2530 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the
2531 RST request sent by the Client to the STS as illustrated in the figure below.



2533
2534 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to
2535 formulate the RST request and will include any security-specific requirements of the STS.

2536
2537 The Client may augment or replace the contents of the RST made to the STS based on the Client-specific
2538 parameters received from the Issued Token policy assertion contained in the Server policy, from policy it
2539 received for the STS, or any other local parameters.

2540

2541 The Issued Token Policy Assertion contains elements which must be understood by the Client. The
2542 assertion contains one element which contains a list of arbitrary elements which should be sent along to
2543 the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
2544 request sent by the Client to the STS following the protocol defined in WS-Trust.

2545

2546 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)
2547 [Trust\]](#). All items are optional, since the Server and STS may already have a pre-arranged relationship
2548 which specifies some or all of the conditions and constraints for issued tokens.

2549

C. Strict Security Header Layout Examples

2550 The following sections describe the security header layout for specific bindings when applying the 'Strict'
2551 layout rules defined in Section 6.7.

2552 C.1 Transport Binding

2553 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

2554 C.1.1 Policy

2555 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport
2556 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username
2557 token attached to the message, and finally an X509 token attached to the message and endorsing the
2558 message signature. No message protection requirements are described since the transport covers all
2559 message parts.

```
2560 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2561   <sp:TransportBinding>
2562     <wsp:Policy>
2563       <sp:TransportToken>
2564         <wsp:Policy>
2565           <sp:HttpsToken />
2566         </wsp:Policy>
2567       </sp:TransportToken>
2568       <sp:AlgorithmSuite>
2569         <wsp:Policy>
2570           <sp:Basic256 />
2571         </wsp:Policy>
2572       </sp:AlgorithmSuite>
2573       <sp:Layout>
2574         <wsp:Policy>
2575           <sp:Strict />
2576         </wsp:Policy>
2577       </sp:Layout>
2578       <sp:IncludeTimestamp />
2579     </wsp:Policy>
2580   </sp:TransportBinding>
2581   <sp:SignedSupportingTokens>
2582     <wsp:Policy>
2583       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2584     </wsp:Policy>
2585   </sp:SignedSupportingTokens>
2586   <sp:SignedEndorsingSupportingTokens>
2587     <wsp:Policy>
2588       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
2589         <wsp:Policy>
2590           <sp:WssX509v3Token10 />
2591         </wsp:Policy>
2592       </sp:X509Token>
2593     </wsp:Policy>
2594   </sp:SignedEndorsingSupportingTokens>
2595   <sp:Wss11>
2596     <sp:RequireSignatureConfirmation />
2597   </sp:Wss11>
2598 </wsp:Policy>
```

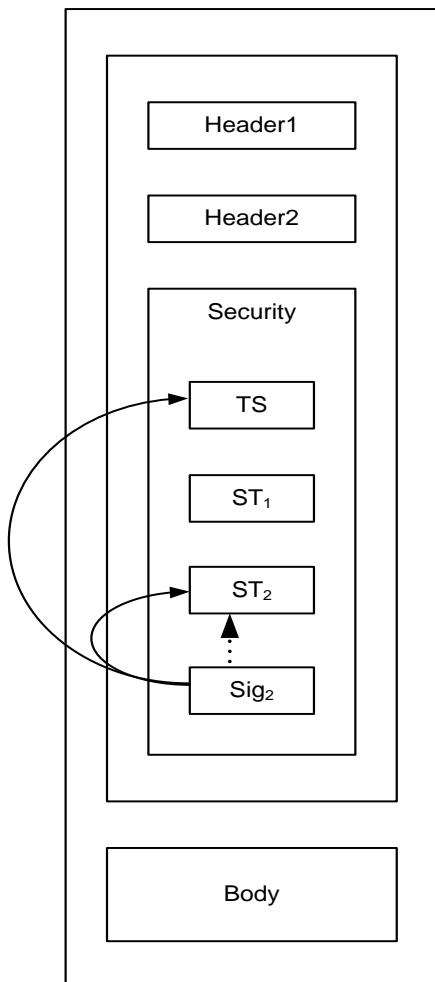
2599 This policy is used as the basis for the examples shown in the subsequent section describing the security
2600 header layout for this binding.

2601 **C.1.2 Initiator to Recipient Messages**

2602 Messages sent from initiator to recipient have the following layout for the security header:

- 2603 1. A `wsu:Timestamp` element.
- 2604 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 2605 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the
2606 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1
2607 above and **SHOULD** cover any other unique identifier for the message in order to prevent
2608 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If
2609 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a
2610 Derived Key Token, based on the supporting token, appears between the supporting token and
2611 the signature.
- 2612 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each
2613 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least
2614 some other unique identifier for the message in order to prevent replays. If [Token Protection] is
2615 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the
2616 supporting token is associated with a symmetric key, then a Derived Key Token, based on the
2617 supporting token, appears before the signature.

2618 The following diagram illustrates the security header layout for the initiator to recipient message:



2619

2620 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The
 2621 arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token labeled ST₂,
 2622 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST₂.
 2623 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering
 2624 of the items in the security header follows the most optimal layout for a receiver to process its contents.

2625 *Example:*

2626 Initiator to recipient message

```

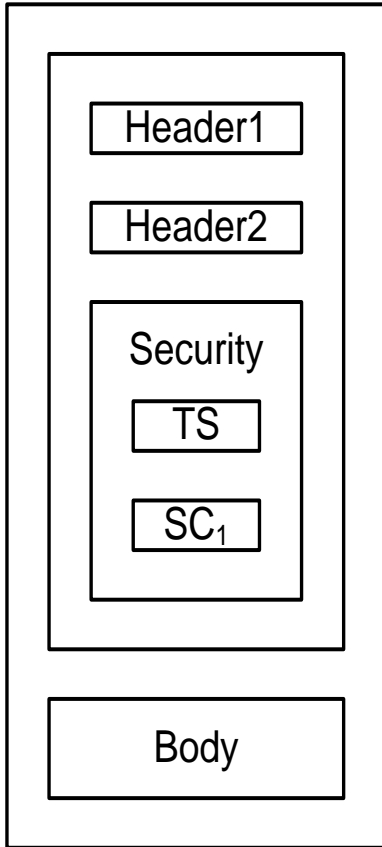
2627 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsp="..." xmlns:ds="...">
2628   <S:Header>
2629     ...
2630     <wsse:Security>
2631       <wsu:Timestamp wsu:Id="timestamp">
2632         <wsu:Created>[datetime]</wsu:Created>
2633         <wsu:Expires>[datetime]</wsu:Expires>
2634       </wsu:Timestamp>
2635       <wsse:UsernameToken wsu:Id='SomeSignedToken' >
2636         ...
2637       </wsse:UsernameToken>
2638       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
2639         ...
2640       </wsse:BinarySecurityToken>
2641       <ds:Signature>
2642         <ds:SignedInfo>
2643           <ds:References>
2644             <ds:Reference URI="#timestamp" />
2645             <ds:Reference URI="#SomeSignedEndorsingToken" />
2646           </ds:References>
2647         </ds:SignedInfo>
2648         <ds:SignatureValue>...</ds:SignatureValue>
2649         <ds:KeyInfo>
2650           <wsse:SecurityTokenReference>
2651             <wsse:Reference URI="#SomeSignedEndorsingToken" />
2652           </wsse:SecurityTokenReference>
2653         </ds:KeyInfo>
2654       </ds:Signature>
2655     ...
2656   </wsse:Security>
2657   ...
2658 </S:Header>
2659 <S:Body>
2660   ...
2661 </S:Body>
2662 </S:Envelope>
  
```

2663 C.1.3 Recipient to Initiator Messages

2664 Messages sent from recipient to initiator have the following layout for the security header:

- 2665 1. A `wsu:Timestamp` element.
- 2666 2. If the [Signature Confirmation] property has a value of 'true', then a
 2667 `wsse11:SignatureConfirmation` element for each signature in the corresponding message
 2668 sent from initiator to recipient. If there are no signatures in the corresponding message from the
 2669 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no Value
 2670 attribute.

2671 The following diagram illustrates the security header layout for the recipient to initiator message:



2672

2673 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 2674 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 2675 message illustrated previously is included. In general, the ordering of the items in the security header
 2676 follows the most optimal layout for a receiver to process its contents.

2677 *Example:*

2678 Recipient to initiator message

```

2679 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
2680   <S:Header>
2681     ...
2682     <wsse:Security>
2683       <wsu:Timestamp wsu:Id="timestamp">
2684         <wsu:Created>[datetime]</wsu:Created>
2685         <wsu:Expires>[datetime]</wsu:Expires>
2686       </wsu:Timestamp>
2687       <wsse11:SignatureConfirmation Value="..." />
2688     ...
2689   </wsse:Security>
2690   ...
2691 </S:Header>
2692 <S:Body>
2693   ...
2694 </S:Body>
2695 </S:Envelope>
  
```

2696 C.2 Symmetric Binding

2697 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

2698 C.2.1 Policy

2699 The following example shows a policy indicating a Symmetric Binding, a symmetric key based
2700 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message
2701 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in
2702 the message signature and the supporting signatures, a username token attached to the message, and
2703 finally an X509 token attached to the message and endorsing the message signature. Minimum message
2704 protection requirements are described as well.

```
2705 <!-- Example Endpoint Policy -->
2706 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2707   <sp:SymmetricBinding>
2708     <wsp:Policy>
2709       <sp:ProtectionToken>
2710         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2711           <sp:Issuer>...</sp:Issuer>
2712           <sp:RequestSecurityTokenTemplate>
2713             ...
2714           </sp:RequestSecurityTokenTemplate>
2715         </sp:IssuedToken>
2716       </sp:ProtectionToken>
2717       <sp:AlgorithmSuite>
2718         <wsp:Policy>
2719           <sp:Basic256 />
2720         </wsp:Policy>
2721       </sp:AlgorithmSuite>
2722       <sp:Layout>
2723         <wsp:Policy>
2724           <sp:Strict />
2725         </wsp:Policy>
2726       </sp:Layout>
2727       <sp:IncludeTimestamp />
2728       <sp:EncryptBeforeSigning />
2729       <sp:EncryptSignature />
2730       <sp:ProtectTokens />
2731     </wsp:Policy>
2732   </sp:SymmetricBinding>
2733   <sp:SignedSupportingTokens>
2734     <wsp:Policy>
2735       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2736     </wsp:Policy>
2737   </sp:SignedSupportingTokens>
2738   <sp:SignedEndorsingSupportingTokens>
2739     <wsp:Policy>
2740       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
2741         <wsp:Policy>
2742           <sp:WssX509v3Token10 />
2743         </wsp:Policy>
2744       </sp:X509Token>
2745     </wsp:Policy>
2746   </sp:SignedEndorsingSupportingTokens>
2747   <sp:Wss11>
2748     <wsp:Policy>
2749       <sp:RequireSignatureConfirmation />
2750     </wsp:Policy>
2751   </sp:Wss11>
2752 </wsp:Policy>
2753
```

```

2754 <!-- Example Message Policy -->
2755 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2756   <sp:SignedParts>
2757     <sp:Header Name="Header1" Namespace="..." />
2758     <sp:Header Name="Header2" Namespace="..." />
2759     <sp:Body/>
2760   </sp:SignedParts>
2761   <sp:EncryptedParts>
2762     <sp:Header Name="Header2" Namespace="..." />
2763     <sp:Body/>
2764   </sp:EncryptedParts>
2765 </wsp:Policy>
2766

```

2767 This policy is used as the basis for the examples shown in the subsequent section describing the security
2768 header layout for this binding.

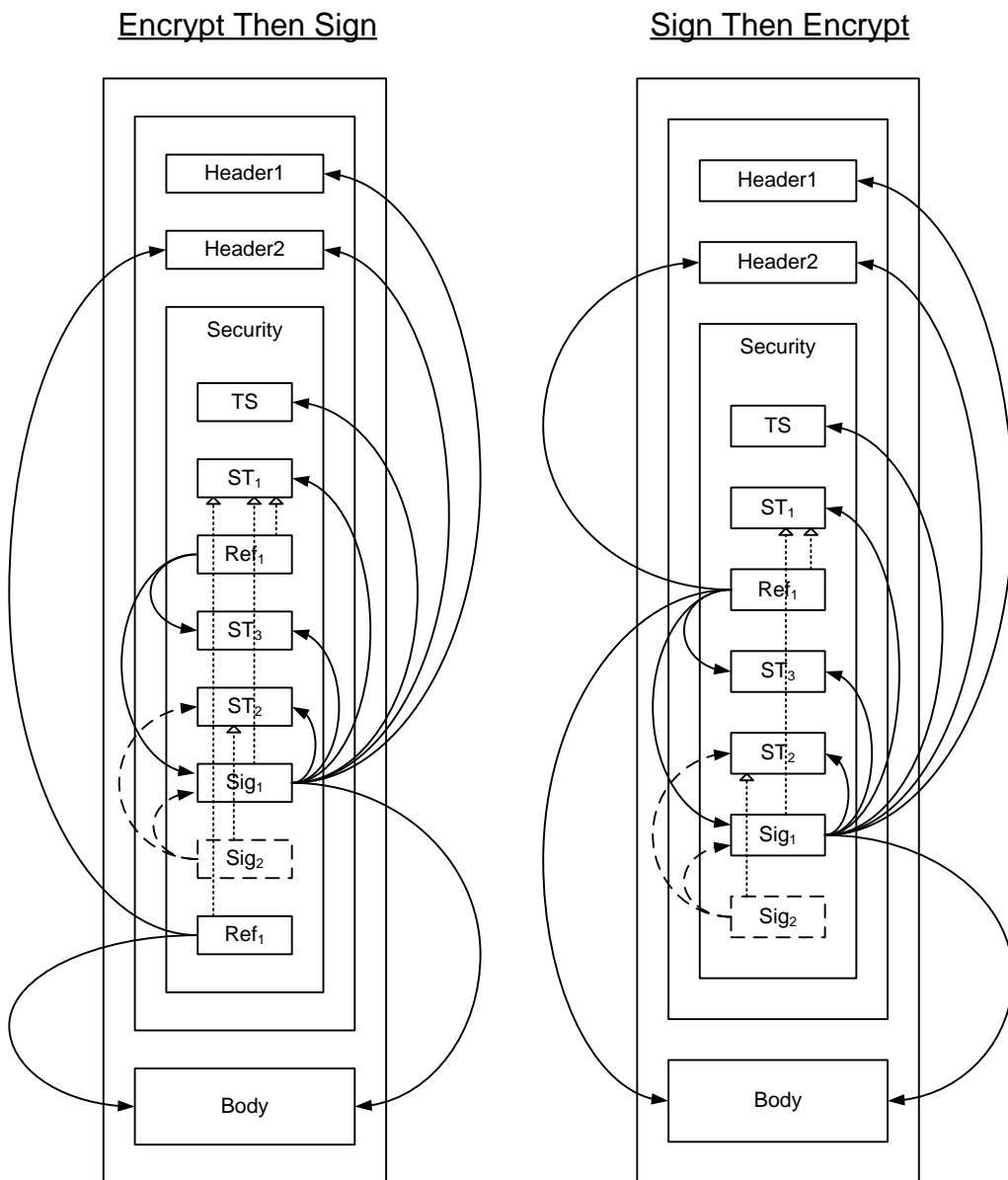
2769 C.2.2 Initiator to Recipient Messages

2770 Messages sent from initiator to recipient have the following layout for the security header:

- 2771 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 2772 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
2773 `.../IncludeToken/Always`, then the [Encryption Token].
- 2774 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
2775 Derived Key Token is used for encryption.
- 2776 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
2777 reference list MUST include a reference to the message signature. If [Protection Order] is
2778 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
2779 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
2780 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 2781 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
2782 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
2783 `.../IncludeToken/Always`.
- 2784 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
2785 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
2786 [Signature Token].
- 2787 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
2788 Derived Key Token is used for signature.
- 2789 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
2790 whether they are included in the message, and any message parts specified in SignedParts
2791 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
2792 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
2793 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 2794 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
2795 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
2796 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
2797 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
2798 endorsing token, appears before the signature.
- 2799 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
2800 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
2801 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
2802 above.

2803

2804 The following diagram illustrates the security header layout for the initiator to recipient message:



2805

2806 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
2807 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
2808 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
2809 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
2810 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
2811 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
2812 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
2813 process its contents.

2814 *Example:*

2815 Initiator to recipient message using EncryptBeforeSigning:

```
2816 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
2817   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
2818   xmlns:xenc="..." xmlns:ds="...">
2819   <S:Header>
2820     <x:Header1 wsu:Id="Header1" >
2821       ...
2822     </x:Header1>
2823
```

```

2824 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
2825   <!-- Plaintext Header2
2826   <x:Header2 wsu:Id="Header2" >
2827     ...
2828   </x:Header2>
2829   -->
2830   ...
2831 </wsse1:EncryptedHeader>
2832 ...
2833 <wsse:Security>
2834   <wsu:Timestamp wsu:Id="Timestamp">
2835     <wsu:Created>...</wsu:Created>
2836     <wsu:Expires>...</wsu:Expires>
2837   </wsu:Timestamp>
2838   <saml:Assertion AssertionId="_SharedSecretToken" ...>
2839     ...
2840   </saml:Assertion>
2841   <xenc:ReferenceList>
2842     <xenc:DataReference URI="#enc_Signature" />
2843     <xenc:DataReference URI="#enc_SomeUsernameToken" />
2844     ...
2845   </xenc:ReferenceList>
2846   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
2847     <!-- Plaintext UsernameToken
2848     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
2849       ...
2850     </wsse:UsernameToken>
2851     -->
2852     ...
2853     <ds:KeyInfo>
2854       <wsse:SecurityTokenReference>
2855         <wsse:Reference URI="#_SharedSecretToken" />
2856       </wsse:SecurityTokenReference>
2857     </ds:KeyInfo>
2858   </xenc:EncryptedData>
2859   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
2860     ...
2861 </wsse:BinarySecurityToken>
2862   <xenc:EncryptedData ID="enc_Signature">
2863     <!-- Plaintext Signature
2864     <ds:Signature Id="Signature">
2865       <ds:SignedInfo>
2866         <ds:References>
2867           <ds:Reference URI="#Timestamp" >...</ds:Reference>
2868           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
2869           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
2870           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
2871           <ds:Reference URI="#Header1" >...</ds:Reference>
2872           <ds:Reference URI="#Header2" >...</ds:Reference>
2873           <ds:Reference URI="#Body" >...</ds:Reference>
2874         </ds:References>
2875       </ds:SignedInfo>
2876     </ds:SignatureValue>...</ds:SignatureValue>
2877     <ds:KeyInfo>
2878       <wsse:SecurityTokenReference>
2879         <wsse:Reference URI="#_SharedSecretToken" />
2880       </wsse:SecurityTokenReference>
2881     </ds:KeyInfo>
2882   </xenc:EncryptedData>
2883   -->
2884   ...
2885   <ds:KeyInfo>
2886     <wsse:SecurityTokenReference>
2887       <wsse:Reference URI="#_SharedSecretToken" />

```

```

2888     </wsse:SecurityTokenReference>
2889   </ds:KeyInfo>
2890 </xenc:EncryptedData>
2891 <ds:Signature>
2892   <ds:SignedInfo>
2893     <ds:References>
2894       <ds:Reference URI="#Signature" >...</ds:Reference>
2895       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
2896     </ds:References>
2897   </ds:SignedInfo>
2898 <ds:SignatureValue>...</ds:SignatureValue>
2899 <ds:KeyInfo>
2900   <wsse:SecurityTokenReference>
2901     <wsse:Reference URI="#SomeSupportingToken" />
2902   </wsse:SecurityTokenReference>
2903 </ds:KeyInfo>
2904 </ds:Signature>
2905 <xenc:ReferenceList>
2906   <xenc:DataReference URI="#enc_Body" />
2907   <xenc:DataReference URI="#enc_Header2" />
2908   ...
2909 </xenc:ReferenceList>
2910 </wsse:Security>
2911 </S:Header>
2912 <S:Body wsu:Id="Body">
2913   <xenc:EncryptedData Id="enc_Body">
2914     ...
2915     <ds:KeyInfo>
2916       <wsse:SecurityTokenReference>
2917         <wsse:Reference URI="#_SharedSecretToken" />
2918       </wsse:SecurityTokenReference>
2919     </ds:KeyInfo>
2920   </xenc:EncryptedData>
2921 </S:Body>
2922 </S:Envelope>

```

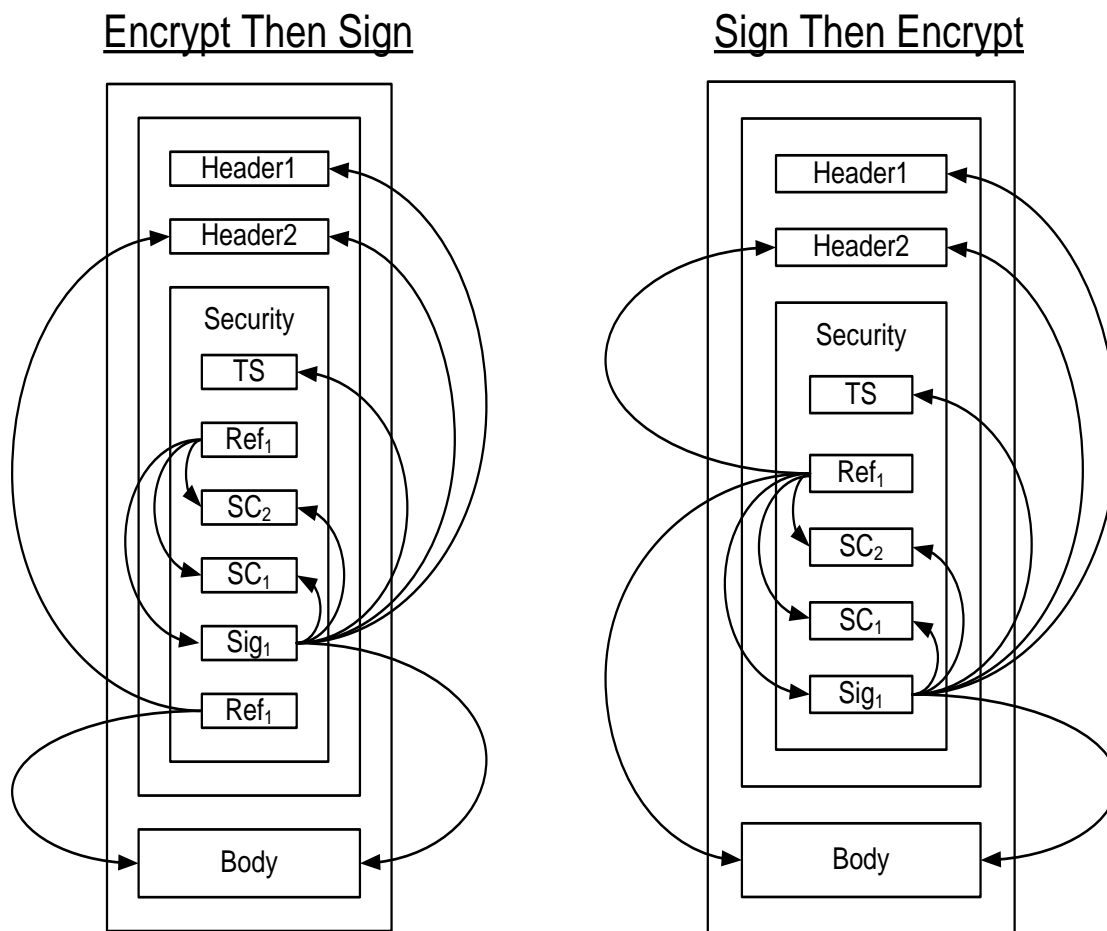
2923 C.2.3 Recipient to Initiator Messages

2924 Messages send from recipient to initiator have the following layout for the security header:

- 2925 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 2926 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the
2927 [Encryption Token].
- 2928 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
2929 Derived Key Token is used for encryption.
- 2930 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
2931 reference list MUST include a reference to the message signature from 6 below, and the
2932 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is
2933 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
2934 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
2935 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2
2936 above.
- 2937 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each
2938 signature in the corresponding message sent from initiator to recipient. If there are no signatures
2939 in the corresponding message from the initiator to the recipient, then a
2940 `wss11:SignatureConfirmation` element with no Value attribute.
- 2941 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
2942 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].

- 2943 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This
 2944 Derived Key Token is used for signature.
- 2945 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation
 2946 elements from 5 above, and all the message parts specified in SignedParts assertions in the
 2947 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]
 2948 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token
 2949 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 2950 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message
 2951 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
 2952 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption
 2953 Token].

2954 The following diagram illustrates the security header layout for the recipient to initiator message:



2955

2956 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 2957 The arrows on the left from boxes labeled Ref₁ indicate references to parts encrypted using a key based
 2958 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two
 2959 wssell:SignatureConfirmation elements labeled SC₁ and SC₂ corresponding to the two signatures
 2960 in the initial message illustrated previously is included. In general, the ordering of the items in the security
 2961 header follows the most optimal layout for a receiver to process its contents. The rules used to determine
 2962 this ordering are described in Appendix C.

2963 *Example:*

2964 Recipient to initiator message using EncryptBeforeSigning:

```
2965 <S:Envelope>
2966   <S:Header>
2967     <x:Header1 wsu:Id="Header1" >
2968       ...
2969     </x:Header1>
2970     <wsse11:EncryptedHeader wsu:Id="enc_Header2">
2971       <!-- Plaintext Header2
2972       <x:Header2 wsu:Id="Header2" >
2973         ...
2974       </x:Header2>
2975       -->
2976       ...
2977     </wsse11:EncryptedHeader>
2978     ...
2979     <wsse:Security>
2980       <wsu:Timestamp wsu:Id="Timestamp">
2981         <wsu:Created>...</wsu:Created>
2982         <wsu:Expires>...</wsu:Expires>
2983       </wsu:Timestamp>
2984       <xenc:ReferenceList>
2985         <xenc:DataReference URI="#enc_Signature" />
2986         <xenc:DataReference URI="#enc_SigConf1" />
2987         <xenc:DataReference URI="#enc_SigConf2" />
2988         ...
2989       </xenc:ReferenceList>
2990       <xenc:EncryptedData ID="enc_SigConf1" >
2991         <!-- Plaintext SignatureConfirmation
2992         <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
2993           ...
2994         </wsse11:SignatureConfirmation>
2995         -->
2996         ...
2997       </xenc:EncryptedData>
2998       <xenc:EncryptedData ID="enc_SigConf2" >
2999         <!-- Plaintext SignatureConfirmation
3000         <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3001           ...
3002         </wsse11:SignatureConfirmation>
3003         -->
3004         ...
3005       </xenc:EncryptedData>
```



```

3006 <xenc:EncryptedData Id="enc_Signature">
3007   <!-- Plaintext Signature
3008   <ds:Signature Id="Signature">
3009     <ds:SignedInfo>
3010       <ds:References>
3011         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3012         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3013         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3014         <ds:Reference URI="#Header1" >...</ds:Reference>
3015         <ds:Reference URI="#Header2" >...</ds:Reference>
3016         <ds:Reference URI="#Body" >...</ds:Reference>
3017       </ds:References>
3018     </ds:SignedInfo>
3019     <ds:SignatureValue>...</ds:SignatureValue>
3020     <ds:KeyInfo>
3021       <wsse:SecurityTokenReference>
3022         <wsse:Reference URI="#_SomeIssuedToken" />
3023       </wsse:SecurityTokenReference>
3024     </ds:KeyInfo>
3025   </ds:Signature>
3026 </xenc:EncryptedData>
3027 ...
3028 </xenc:EncryptedData>
3029 ...
3030 <ds:KeyInfo>
3031   <wsse:SecurityTokenReference>
3032     <wsse:Reference URI="#_SomeIssuedToken" />
3033   </wsse:SecurityTokenReference>
3034 </ds:KeyInfo>
3035 <xenc:EncryptedData>
3036 <xenc:ReferenceList>
3037   <xenc:DataReference URI="#enc_Body" />
3038   <xenc:DataReference URI="#enc_Header2" />
3039   ...
3040 </xenc:ReferenceList>
3041 </xenc:EncryptedData>
3042 </wsse:Security>
3043 </S:Header>
3044 <S:Body wsu:Id="Body">
3045   <xenc:EncryptedData Id="enc_Body">
3046     ...
3047     <ds:KeyInfo>
3048       <wsse:SecurityTokenReference>
3049         <wsse:Reference URI="#_SomeIssuedToken" />
3050       </wsse:SecurityTokenReference>
3051     </ds:KeyInfo>
3052   </xenc:EncryptedData>
3053 </S:Body>
3054 </S:Envelope>

```

3055 C.3 Asymmetric Binding

3056 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

3057 C.3.1 Policy

3058 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator
3059 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the
3060 message parts before signing, a requirement to encrypt the message signature, a requirement to include
3061 tokens in the message signature and the supporting signatures, a requirement to include
3062 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3063 an X509 token attached to the message and endorsing the message signature. Minimum message
3064 protection requirements are described as well.

```
3065 <!-- Example Endpoint Policy -->
3066 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3067   <sp:AsymmetricBinding>
3068     <wsp:Policy>
3069       <sp:RecipientToken>
3070         <wsp:Policy>
3071           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3072         </wsp:Policy>
3073       </sp:RecipientToken>
3074       <sp:InitiatorToken>
3075         <wsp:Policy>
3076           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3077         </wsp:Policy>
3078       </sp:InitiatorToken>
3079       <sp:AlgorithmSuite>
3080         <wsp:Policy>
3081           <sp:Basic256 />
3082         </wsp:Policy>
3083       </sp:AlgorithmSuite>
3084       <sp:Layout>
3085         <wsp:Policy>
3086           <sp:Strict />
3087         </wsp:Policy>
3088       </sp:Layout>
3089       <sp:IncludeTimestamp />
3090       <sp:EncryptBeforeSigning />
3091       <sp:EncryptSignature />
3092       <sp:ProtectTokens />
3093     </wsp:Policy>
3094   </sp:AsymmetricBinding>
3095   <sp:SignedEncryptedSupportingTokens>
3096     <wsp:Policy>
3097       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3098     </wsp:Policy>
3099   </sp:SignedEncryptedSupportingTokens>
3100   <sp:SignedEndorsingSupportingTokens>
3101     <wsp:Policy>
3102       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3103         <wsp:Policy>
3104           <sp:WssX509v3Token10 />
3105         </wsp:Policy>
3106       </sp:X509Token>
3107     </wsp:Policy>
3108   </sp:SignedEndorsingSupportingTokens>
3109   <sp:Wss11>
3110     <wsp:Policy>
3111       <sp:RequireSignatureConfirmation />
3112     </wsp:Policy>
3113   </sp:Wss11>
3114 </wsp:Policy>
3115
```

3116

```
3117 <!-- Example Message Policy -->
3118 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3119   <sp:SignedParts>
3120     <sp:Header Name="Header1" Namespace="..." />
3121     <sp:Header Name="Header2" Namespace="..." />
3122     <sp:Body/>
3123   </sp:SignedParts>
3124   <sp:EncryptedParts>
3125     <sp:Header Name="Header2" Namespace="..." />
3126     <sp:Body/>
3127   </sp:EncryptedParts>
3128 </wsp:All>
```

3129
3130 This policy is used as the basis for the examples shown in the subsequent section describing the security
3131 header layout for this binding.

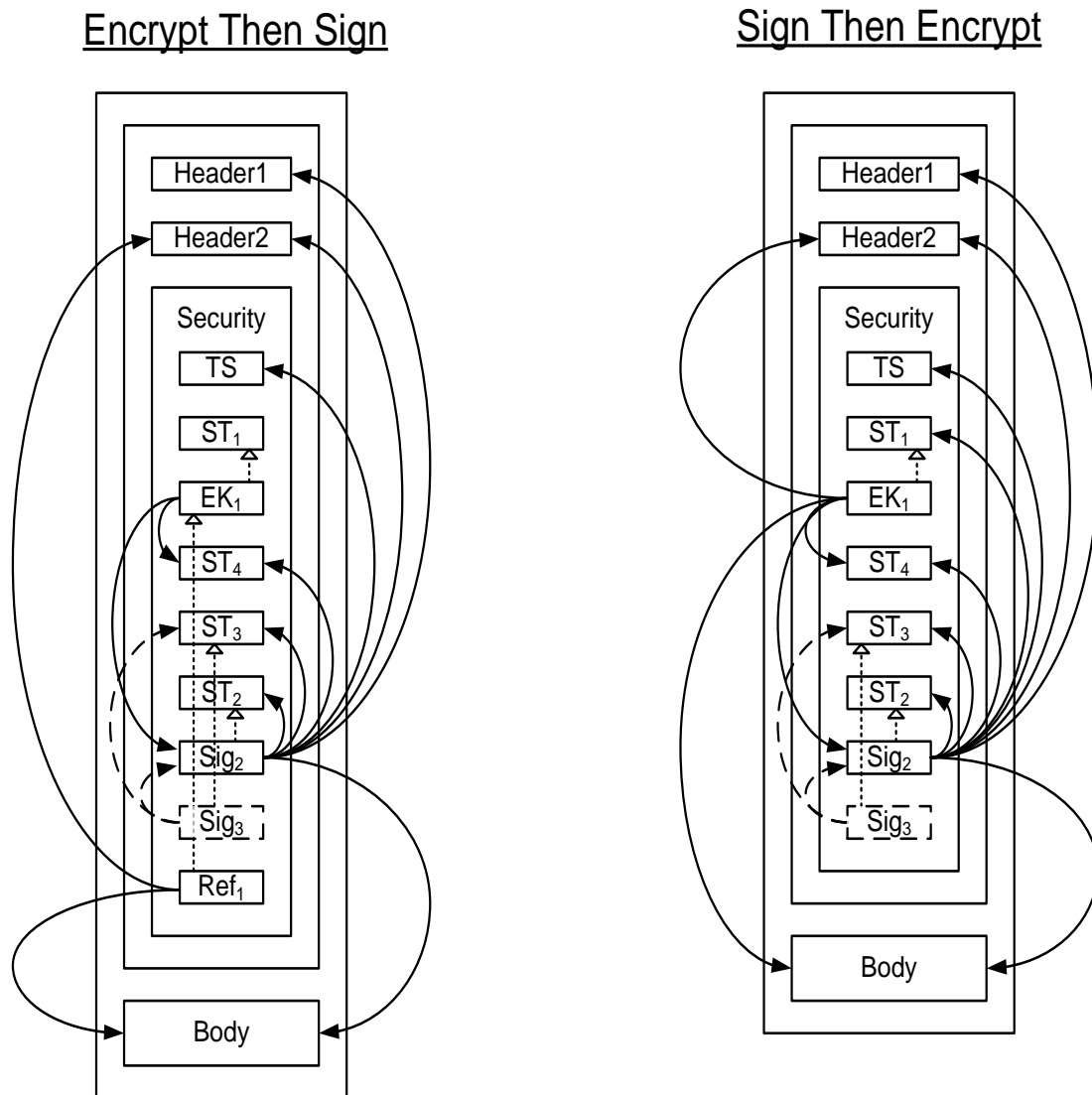
3132 **C.3.2 Initiator to Recipient Messages**

3133 Messages sent from initiator to recipient have the following layout:

- 3134 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3135 2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3136 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
- 3137 3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3138 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3139 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3140 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If
3141 [Signature Protection] is 'true' then the reference list MUST contain a reference to the message
3142 signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message
3143 signature.
- 3144 4. Any tokens from the supporting tokens properties (as defined in section 8) whose
3145 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3146 5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3147 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
- 3148 6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from
3149 1 above, any tokens from 4 above regardless of whether they are included in the message, and
3150 any message parts specified in `SignedParts` assertions in the policy. If [Token Protection] is 'true',
3151 the signature MUST also cover the [Initiator Token] regardless of whether it is included in the
3152 message.
- 3153 7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting
3154 Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a
3155 symmetric key, then a Derived Key Token, based on the supporting token, appears before the
3156 signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token
3157 regardless of whether it is included in the message.
- 3158 8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
3159 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
3160 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
3161 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions
3162 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
3163 element from 3 above.

3164

3165 The following diagram illustrates the security header layout for the initiator to recipient messages:



3166
 3167 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
 3168 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate
 3169 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as
 3170 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references
 3171 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left
 3172 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the
 3173 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as
 3174 the basis for each cryptographic operation. In general, the ordering of the items in the security header
 3175 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
 3176 ordering are described in Appendix C.

3177
 3178 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
 3179 key contains an external reference to the token containing the encryption key. The diagram illustrates
 3180 how one might attach a security token related to the encrypted key for completeness. One possible use-

3181 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3182 wishes to include the encryption token in the message signature.

3183 Initiator to recipient message *Example*

3184 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3185     xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3186 <S:Header>
3187   <x:Header1 wsu:Id="Header1" >
3188     ...
3189   </x:Header1>
3190   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3191     <!-- Plaintext Header2
3192     <x:Header2 wsu:Id="Header2" >
3193       ...
3194     </x:Header2>
3195     -->
3196     ...
3197   </wssell1:EncryptedHeader>
3198   ...
3199   <wsse:Security>
3200     <wsu:Timestamp wsu:Id="Timestamp">
3201       <wsu:Created>...</wsu:Created>
3202       <wsu:Expires>...</wsu:Expires>
3203     </wsu:Timestamp>
3204     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3205       ...
3206     </wsse:BinarySecurityToken>
3207     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3208       ...
3209     <xenc:ReferenceList>
3210       <xenc:DataReference URI="#enc_Signature" />
3211       <xenc:DataReference URI="#enc_SomeUsernameToken" />
3212       ...
3213     </xenc:ReferenceList>
3214   </xenc:EncryptedKey>
3215   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3216     <!-- Plaintext UsernameToken
3217     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3218       ...
3219     </wsse:UsernameToken>
3220     -->
3221     ...
3222   </xenc:EncryptedData>
3223   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3224     ...
3225   </wsse:BinarySecurityToken>
3226   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3227     ...
3228   </wsse:BinarySecurityToken>
3229   <xenc:EncryptedData ID="enc_Signature">
3230     <!-- Plaintext Signature
3231     <ds:Signature Id="Signature">
3232       <ds:SignedInfo>
3233         <ds:References>
3234           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3235           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3236           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3237           <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3238           <ds:Reference URI="#Header1" >...</ds:Reference>
3239           <ds:Reference URI="#Header2" >...</ds:Reference>
3240           <ds:Reference URI="#Body" >...</ds:Reference>
3241         </ds:References>
3242       </ds:SignedInfo>
3243     </ds:SignatureValue>...</ds:SignatureValue>
3244     <ds:KeyInfo>
3245       <wsse:SecurityTokenReference>
3246         <wsse:Reference URI="#InitiatorToken" />
3247       </wsse:SecurityTokenReference>
3248     </ds:KeyInfo>

```

```

3249     </ds:Signature>
3250     -->
3251     ...
3252 </xenc:EncryptedData>
3253 <ds:Signature>
3254   <ds:SignedInfo>
3255     <ds:References>
3256       <ds:Reference URI="#Signature" >...</ds:Reference>
3257       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3258     </ds:References>
3259   </ds:SignedInfo>
3260   <ds:SignatureValue>...</ds:SignatureValue>
3261   <ds:KeyInfo>
3262     <wsse:SecurityTokenReference>
3263       <wsse:Reference URI="#SomeSupportingToken" />
3264     </wsse:SecurityTokenReference>
3265   </ds:KeyInfo>
3266 </ds:Signature>
3267 <xenc:ReferenceList>
3268   <xenc:DataReference URI="#enc_Body" />
3269   <xenc:DataReference URI="#enc_Header2" />
3270   ...
3271 </xenc:ReferenceList>
3272 </wsse:Security>
3273 </S:Header>
3274 <S:Body wsu:Id="Body">
3275   <xenc:EncryptedData Id="enc_Body">
3276     ...
3277     <ds:KeyInfo>
3278       <wsse:SecurityTokenReference>
3279         <wsse:Reference URI="#RecipientEncryptedKey" />
3280       </wsse:SecurityTokenReference>
3281     </ds:KeyInfo>
3282   </xenc:EncryptedData>
3283 </S:Body>
3284 </S:Envelope>

```

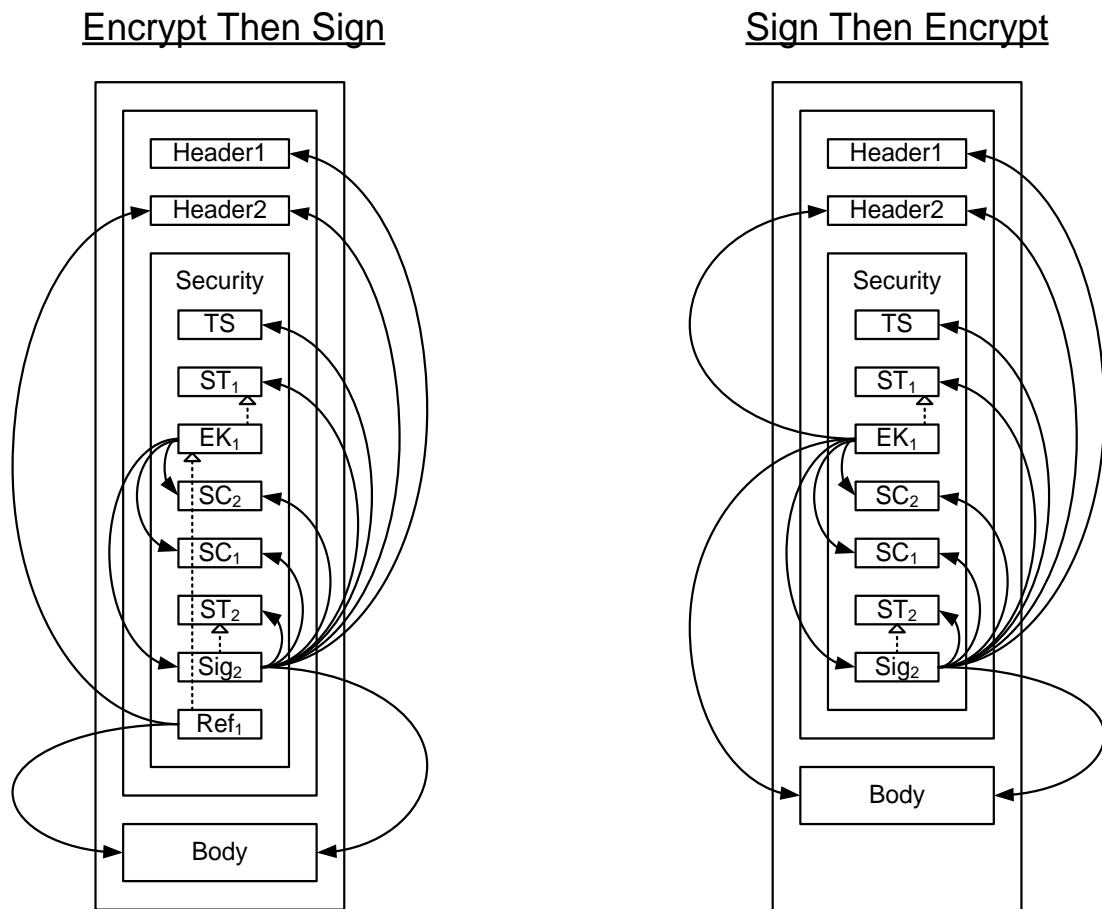
3285 C.3.3 Recipient to Initiator Messages

3286 Messages sent from recipient to initiator have the following layout:

- 3287 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3288 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3289 `.../IncludeToken/Always`, then the [Initiator Token].
- 3290 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3291 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3292 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3293 reference to all the message parts specified in EncryptedParts assertions in the policy. If
3294 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3295 message signature from 6 below, if any and references to the
3296 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3297 4. If [Signature Confirmation] is 'true', then a `wss11:SignatureConfirmation` element for each
3298 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3299 in the corresponding message from the initiator to the recipient, then a
3300 `wss11:SignatureConfirmation` element with no Value attribute.
- 3301 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3302 `.../IncludeToken/Always`, then the [Recipient Token].

- 3303 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],
 3304 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements
 3305 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token
 3306 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3307 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
 3308 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
 3309 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
 3310 reference list includes a reference to all the message parts specified in EncryptedParts assertions
 3311 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
 3312 element from 3 above.

3313
 3314 The following diagram illustrates the security header layout for the recipient to initiator messages:



3315

3316 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
 3317 using the [Recipient Token] labeled ST₂. The arrows on the left from boxes labeled EK₁ indicate
 3318 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on
 3319 the left from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained
 3320 in the encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token
 3321 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements
 3322 labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is
 3323 included. In general, the ordering of the items in the security header follows the most optimal layout for a
 3324 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

3325 Recipient to initiator message *Example*:


```

3326 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3327     xmlns:wssell="..." xmlns:wsse="..."
3328     xmlns:xenc="..." xmlns:ds="...">
3329 <S:Header>
3330     <x:Header1 wsu:Id="Header1" >
3331         ...
3332     </x:Header1>
3333     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3334         <!-- Plaintext Header2
3335         <x:Header2 wsu:Id="Header2" >
3336             ...
3337         </x:Header2>
3338         -->
3339         ...
3340     </wssell:EncryptedHeader>
3341     ...
3342     <wsse:Security>
3343         <wsu:Timestamp wsu:Id="Timestamp">
3344             <wsu:Created>...</wsu:Created>
3345             <wsu:Expires>...</wsu:Expires>
3346         </wsu:Timestamp>
3347         <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3348             ...
3349         </wsse:BinarySecurityToken>
3350         <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3351             ...
3352             <xenc:ReferenceList>
3353                 <xenc:DataReference URI="#enc_Signature" />
3354                 <xenc:DataReference URI="#enc_SigConf1" />
3355                 <xenc:DataReference URI="#enc_SigConf2" />
3356                 ...
3357             </xenc:ReferenceList>
3358         </xenc:EncryptedKey>
3359         <xenc:EncryptedData ID="enc_SigConf2" >
3360             <!-- Plaintext SignatureConfirmation
3361             <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3362                 ...
3363             </wssell:SignatureConfirmation>
3364             -->
3365             ...
3366         </xenc:EncryptedData>
3367         <xenc:EncryptedData ID="enc_SigConf1" >
3368             <!-- Plaintext SignatureConfirmation
3369             <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3370                 ...
3371             </wssell:SignatureConfirmation>
3372             -->
3373             ...
3374         </xenc:EncryptedData>
3375         <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3376             ...
3377         </wsse:BinarySecurityToken>
3378     </wsse:Security>

```

```

3379 <xenc:EncryptedData ID="enc_Signature">
3380 <!-- Plaintext Signature
3381 <ds:Signature Id="Signature">
3382 <ds:SignedInfo>
3383 <ds:References>
3384 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3385 <ds:Reference URI="#SigConf1" >...</ds:Reference>
3386 <ds:Reference URI="#SigConf2" >...</ds:Reference>
3387 <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3388 <ds:Reference URI="#Header1" >...</ds:Reference>
3389 <ds:Reference URI="#Header2" >...</ds:Reference>
3390 <ds:Reference URI="#Body" >...</ds:Reference>
3391 </ds:References>
3392 </ds:SignedInfo>
3393 <ds:SignatureValue>...</ds:SignatureValue>
3394 <ds:KeyInfo>
3395 <wsse:SecurityTokenReference>
3396 <wsse:Reference URI="#RecipientToken" />
3397 </wsse:SecurityTokenReference>
3398 </ds:KeyInfo>
3399 </ds:Signature>
3400 -->
3401 ...
3402 </xenc:EncryptedData>
3403 <xenc:ReferenceList>
3404 <xenc:DataReference URI="#enc_Body" />
3405 <xenc:DataReference URI="#enc_Header2" />
3406 ...
3407 </xenc:ReferenceList>
3408 </wsse:Security>
3409 </S:Header>
3410 <S:Body wsu:Id="Body">
3411 <xenc:EncryptedData Id="enc_Body">
3412 ...
3413 <ds:KeyInfo>
3414 <wsse:SecurityTokenReference>
3415 <wsse:Reference URI="#InitiatorEncryptedKey" />
3416 </wsse:SecurityTokenReference>
3417 </ds:KeyInfo>
3418 </xenc:EncryptedData>
3419 </S:Body>
3420 </S:Envelope>

```

3421 **D. Signed and Encrypted Elements in the Security**
3422 **Header**

3423 This section lists the criteria for when various child elements of the Security header are signed and/or
3424 encrypted at the message level including whether they are signed by the message signature or a
3425 supporting signature. It assumes that there are no `sp:SignedElements` and no
3426 `sp:EncryptedElements` assertions in the policy. If such assertions are present in the policy then
3427 additional child elements of the security header might be signed and/or encrypted.

3428 **D.1 Elements signed by the message signature**

- 3429 1. The `wsu:Timestamp` element (Section 6.2).
3430 2. All `wssell:SignatureConfirmation` elements (Section 9).
3431 3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token],
3432 [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption
3433 Token] when [Token Protection] has a value of 'true' (Section 6.5).
3434 4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed
3435 Endorsing Supporting Tokens] (Section 8.5).

3436 **D.2 Elements signed by all endorsing signatures**

- 3437 1. The `ds:Signature` element that forms the message signature (Section 8.3).
3438 2. The `wsu:Timestamp` element in the case of a transport binding (Section 8.3).

3439 **D.3 Elements signed by a specific endorsing signature**

- 3440 1. Security Tokens corresponding to [Endorsing Supporting Tokens] or [Signed Endorsing
3441 Supporting Tokens] when [Token Protection] has a value of 'true' (Section 8.8).

3442 **D.4 Elements that are encrypted**

- 3443 1. The `ds:Signature` element that forms the message signature when [Signature Protection]
3444 has a value of 'true' (Section 6.4).
3445 2. All `wssell:SignatureConfirmation` elements when [Signature Protection] has a value
3446 of 'true' (Section 6.4).
3447 3. A `wsse:UsernameToken` may be encrypted when a transport binding is not being used
3448 (Section 5.3.1).
3449

3450

E. Acknowledgements

3451 The following individuals have participated in the creation of this specification and are gratefully
3452 acknowledged:

3453 **Original Authors of the initial contribution:**

3454 Giovanni Della-Libera, Microsoft

3455 Martin Gudgin, Microsoft

3456 Phillip Hallam-Baker, VeriSign

3457 Maryann Hondo, IBM

3458 Hans Granqvist, Verisign

3459 Chris Kaler, Microsoft (editor)

3460 Hiroshi Maruyama, IBM

3461 Michael McIntosh, IBM

3462 Anthony Nadalin, IBM (editor)

3463 Nataraj Nagaratnam, IBM

3464 Rob Philpott, RSA Security

3465 Hemma Prafullchandra, VeriSign

3466 John Shewchuk, Microsoft

3467 Doug Walter, Microsoft

3468 Riaz Zolfonoon, RSA Security

3469

3470 **Original Acknowledgements of the initial contribution:**

3471 Vaithialingam B. Balayoghan, Microsoft

3472 Francisco Curbera, IBM

3473 Christopher Ferris, IBM

3474 Cédric Fournet, Microsoft

3475 Andy Gordon, Microsoft

3476 Tomasz Janczuk, Microsoft

3477 David Melgar, IBM

3478 Mike Perks, IBM

3479 Bruce Rich, IBM

3480 Jeffrey Schlimmer, Microsoft

3481 Chris Sharp, IBM

3482 Kent Tamura, IBM

3483 T.R. Vishwanath, Microsoft

3484 Elliot Waingold, Microsoft

3485

3486 **TC Members during the development of this specification:**

3487 Don Adams, Tibco Software Inc.

3488 Jan Alexander, Microsoft Corporation

3489 Steve Anderson, BMC Software

3490 Donal Arundel, IONA Technologies

3491 Howard Bae, Oracle Corporation

3492 Abbie Barbir, Nortel Networks Limited

3493 Charlton Barreto, Adobe Systems

3494 Mighael Botha, Software AG, Inc.

3495 Toufic Boubez, Layer 7 Technologies Inc.

3496 Norman Brickman, Mitre Corporation

3497 Melissa Brumfield, Booz Allen Hamilton

3498 Lloyd Burch, Novell
3499 Scott Cantor, Internet2
3500 Greg Carpenter, Microsoft Corporation
3501 Steve Carter, Novell
3502 Ching-Yun (C.Y.) Chao, IBM
3503 Martin Chapman, Oracle Corporation
3504 Kate Cherry, Lockheed Martin
3505 Henry (Hyenvui) Chung, IBM
3506 Luc Clement, Systinet Corp.
3507 Paul Cotton, Microsoft Corporation
3508 Glen Daniels, Sonic Software Corp.
3509 Peter Davis, Neustar, Inc.
3510 Martijn de Boer, SAP AG
3511 Werner Dittmann, Siemens AG
3512 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
3513 Fred Dushin, IONA Technologies
3514 Petr Dvorak, Systinet Corp.
3515 Colleen Evans, Microsoft Corporation
3516 Ruchith Fernando, WSO2
3517 Mark Fussell, Microsoft Corporation
3518 Vijay Gajjala, Microsoft Corporation
3519 Marc Goodner, Microsoft Corporation
3520 Hans Granqvist, VeriSign
3521 Martin Gudgin, Microsoft Corporation
3522 Tony Gullotta, SOA Software Inc.
3523 Jiandong Guo, Sun Microsystems
3524 Phillip Hallam-Baker, VeriSign
3525 Patrick Harding, Ping Identity Corporation
3526 Heather Hinton, IBM
3527 Frederick Hirsch, Nokia Corporation
3528 Jeff Hodges, Neustar, Inc.
3529 Will Hopkins, BEA Systems, Inc.
3530 Alex Hristov, Otecia Incorporated
3531 John Hughes, PA Consulting
3532 Diane Jordan, IBM
3533 Venugopal K, Sun Microsystems
3534 Chris Kaler, Microsoft Corporation
3535 Dana Kaufman, Forum Systems, Inc.
3536 Paul Knight, Nortel Networks Limited
3537 Ramanathan Krishnamurthy, IONA Technologies
3538 Christopher Kurt, Microsoft Corporation
3539 Kelvin Lawrence, IBM
3540 Hubert Le Van Gong, Sun Microsystems
3541 Jong Lee, BEA Systems, Inc.
3542 Rich Levinson, Oracle Corporation
3543 Tommy Lindberg, Dajeil Ltd.
3544 Mark Little, JBoss Inc.
3545 Hal Lockhart, BEA Systems, Inc.
3546 Mike Lyons, Layer 7 Technologies Inc.
3547 Eve Maler, Sun Microsystems
3548 Ashok Malhotra, Oracle Corporation
3549 Anand Mani, CrimsonLogic Pte Ltd
3550 Jonathan Marsh, Microsoft Corporation
3551 Robin Martherus, Oracle Corporation
3552 Miko Matsumura, Infravio, Inc.
3553 Gary McAfee, IBM
3554 Michael McIntosh, IBM

3555 John Merrells, Sxip Networks SRL
3556 Jeff Mischkinsky, Oracle Corporation
3557 Prateek Mishra, Oracle Corporation
3558 Bob Morgan, Internet2
3559 Vamsi Motukuru, Oracle Corporation
3560 Raajmohan Na, EDS
3561 Anthony Nadalin, IBM
3562 Andrew Nash, Reactivity, Inc.
3563 Eric Newcomer, IONA Technologies
3564 Duane Nickull, Adobe Systems
3565 Toshihiro Nishimura, Fujitsu Limited
3566 Rob Philpott, RSA Security
3567 Denis Pilipchuk, BEA Systems, Inc.
3568 Darren Platt, Ping Identity Corporation
3569 Martin Raepfle, SAP AG
3570 Nick Ragouzis, Enosis Group LLC
3571 Prakash Reddy, CA
3572 Alain Regnier, Ricoh Company, Ltd.
3573 Irving Reid, Hewlett-Packard
3574 Bruce Rich, IBM
3575 Tom Rutt, Fujitsu Limited
3576 Maneesh Sahu, Actional Corporation
3577 Frank Siebenlist, Argonne National Laboratory
3578 Joe Smith, Apani Networks
3579 Davanum Srinivas, WSO2
3580 Yakov Sverdlov, CA
3581 Gene Thurston, AmberPoint
3582 Victor Valle, IBM
3583 Asir Vedamuthu, Microsoft Corporation
3584 Greg Whitehead, Hewlett-Packard
3585 Ron Williams, IBM
3586 Corinna Witt, BEA Systems, Inc.
3587 Kyle Young, Microsoft Corporation
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602