



WS-SecurityPolicy 1.3

OASIS Committee Draft 03

12 November 2008

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-03.doc>
(Authoritative)
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-03.pdf>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-03.html>

Previous Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-02.doc>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-02.pdf>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cd/ws-securitypolicy-1.3-spec-cd-02.html>

Latest Version:

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

Technical Committee:

OASIS Web Services Secure Exchange TC

Chair(s):

Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):

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

Related work:

N/A

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the

“Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/ws-sx>.

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

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

Notices

Copyright © OASIS® 1993–2008. All Rights Reserved.

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

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

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

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

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

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

Table of Contents

1	Introduction	7
1.1	Example	7
1.2	Namespaces	8
1.3	Schema Files	9
1.4	Terminology	9
1.4.1	Notational Conventions	9
1.5	Normative References	10
1.6	Non-Normative References	13
2	Security Policy Model	14
2.1	Security Assertion Model	14
2.2	Nested Policy Assertions	15
2.3	Security Binding Abstraction	15
3	Policy Considerations	17
3.1	Nested Policy	17
3.2	Policy Subjects	17
4	Protection Assertions	19
4.1	Integrity Assertions	19
4.1.1	SignedParts Assertion	19
4.1.2	SignedElements Assertion	20
4.2	Confidentiality Assertions	21
4.2.1	EncryptedParts Assertion	21
4.2.2	EncryptedElements Assertion	22
4.2.3	ContentEncryptedElements Assertion	23
4.3	Required Elements Assertion	23
4.3.1	RequiredElements Assertion	24
4.3.2	RequiredParts Assertion	24
5	Token Assertions	26
5.1	Token Inclusion	26
5.1.1	Token Inclusion Values	26
5.1.2	Token Inclusion and Token References	27
5.2	Token Issuer and Required Claims	27
5.2.1	Token Issuer	27
5.2.2	Token Issuer Name	27
5.2.3	Required Claims	27
5.2.4	Processing Rules and Token Matching	28
5.3	Token Properties	28
5.3.1	[Derived Keys] Property	28
5.3.2	[Explicit Derived Keys] Property	28
5.3.3	[Implied Derived Keys] Property	28
5.4	Token Assertion Types	28
5.4.1	UsernameToken Assertion	28

5.4.2	ICreatessuedToken Assertion	30
5.4.3	X509Token Assertion	32
5.4.4	KerberosToken Assertion.....	34
5.4.5	SpnegoContextToken Assertion	36
5.4.6	SecurityContextToken Assertion	37
5.4.7	SecureConversationToken Assertion	38
5.4.8	SamlToken Assertion	42
5.4.9	RelToken Assertion.....	44
5.4.10	HttpsToken Assertion.....	45
5.4.11	KeyValueToken Assertion.....	46
6	Security Binding Properties	49
6.1	[Algorithm Suite] Property	49
6.2	[Timestamp] Property	51
6.3	[Protection Order] Property	51
6.4	[Signature Protection] Property.....	51
6.5	[Token Protection] Property.....	51
6.6	[Entire Header and Body Signatures] Property	52
6.7	[Security Header Layout] Property.....	52
6.7.1	Strict Layout Rules for WSS 1.0	52
7	Security Binding Assertions	54
7.1	AlgorithmSuite Assertion	54
7.2	Layout Assertion.....	56
7.3	TransportBinding Assertion	57
7.4	SymmetricBinding Assertion	58
7.5	AsymmetricBinding Assertion	60
8	Supporting Tokens	63
8.1	SupportingTokens Assertion.....	64
8.2	SignedSupportingTokens Assertion	65
8.3	EndorsingSupportingTokens Assertion	67
8.4	SignedEndorsingSupportingTokens Assertion	69
8.5	SignedEncryptedSupportingTokens Assertion.....	71
8.6	EncryptedSupportingTokens Assertion	71
8.7	EndorsingEncryptedSupportingTokens Assertion	71
8.8	SignedEndorsingEncryptedSupportingTokens Assertion.....	71
8.9	Interaction between [Token Protection] property and supporting token assertions	71
8.10	Example	72
9	WSS: SOAP Message Security Options	73
9.1	Wss10 Assertion	74
9.2	Wss11 Assertion	75
10	WS-Trust Options.....	77
10.1	Trust13 Assertion	78
11	Guidance on creating new assertions and assertion extensibility	80
11.1	General Design Points	80

11.2 Detailed Design Guidance	80
12 Security Considerations	82
13 Conformance	83
A. Assertions and WS-PolicyAttachment	84
A.1 Endpoint Policy Subject Assertions	84
A.1.1 Security Binding Assertions	84
A.1.2 Token Assertions	84
A.1.3 WSS: SOAP Message Security 1.0 Assertions	84
A.1.4 WSS: SOAP Message Security 1.1 Assertions	84
A.1.5 Trust 1.0 Assertions	84
A.2 Operation Policy Subject Assertions	84
A.2.1 Security Binding Assertions	84
A.2.2 Supporting Token Assertions	84
A.3 Message Policy Subject Assertions	85
A.3.1 Supporting Token Assertions	85
A.3.2 Protection Assertions	85
A.4 Assertions With Undefined Policy Subject	85
A.4.1 General Assertions	85
A.4.2 Token Usage Assertions	85
A.4.3 Token Assertions	85
B. Issued Token Policy	87
C. Strict Security Header Layout Examples	89
C.1 Transport Binding	89
C.1.1 Policy	89
C.1.2 Initiator to Recipient Messages	90
C.1.3 Recipient to Initiator Messages	91
C.2 Symmetric Binding	92
C.2.1 Policy	93
C.2.2 Initiator to Recipient Messages	94
C.2.3 Recipient to Initiator Messages	98
C.3 Asymmetric Binding	101
C.3.1 Policy	101
C.3.2 Initiator to Recipient Messages	103
C.3.3 Recipient to Initiator Messages	107
D. Signed and Encrypted Elements in the Security Header	111
D.1 Elements signed by the message signature	111
D.2 Elements signed by all endorsing signatures	111
D.3 Elements signed by a specific endorsing signature	111
D.4 Elements that are encrypted	111
E. Acknowledgements	112

1 Introduction

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

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

1.1 Example

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

Table 1: Example security policy.

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

```

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

```

46

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

59 1.2 Namespaces

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

```

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

```

63

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

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

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

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

67 1.3 Schema Files

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

70 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy-1.2.xsd>
71 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy-1.3.xsd>

72 1.4 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

89 1.4.1 Notational Conventions

90 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD
91 NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described
92 in [[RFC2119](#)].

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

- 94 • The syntax appears as an XML instance, but values in italics indicate data types instead of literal
95 values.
- 96 • Characters are appended to elements and attributes to indicate cardinality:
 - 97 ○ "?" (0 or 1)
 - 98 ○ "*" (0 or more)
 - 99 ○ "+" (1 or more)
- 100 • The character "|" is used to indicate a choice between alternatives.
- 101 • The characters "(" and ")" are used to indicate that contained items are to be treated as a group
102 with respect to cardinality or choice.

- 103 • The characters "[" and "]" are used to call out references and property names.
- 104 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be
- 105 added at the indicated extension points but MUST NOT contradict the semantics of the parent
- 106 and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver
- 107 SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated
- 108 below.
- 109 • XML namespace prefixes (see Table 2) are used to indicate the namespace of the element being
- 110 defined.

- 111
- 112 Elements and Attributes defined by this specification are referred to in the text of this document using
- 113 XPath 1.0 expressions. Extensibility points are referred to using an extended version of this syntax:
- 114 • An element extensibility point is referred to using {any} in place of the element name. This
 - 115 indicates that any element name can be used, from any namespace other than the namespace of
 - 116 this specification.
 - 117 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
 - 118 indicates that any attribute name can be used, from any namespace other than the namespace of
 - 119 this specification.

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

121 In this document reference is made to the `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires`

122 elements in a utility schema ([http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd)

123 [1.0.xsd](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

124 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp

125 element could reference it (as is done here).

126

127 WS-SecurityPolicy is designed to work with the general Web Services framework including WSDL service

128 descriptions, UDDI businessServices and bindingTemplates and SOAP message structure and message

129 processing model, and WS-SecurityPolicy SHOULD be applicable to any version of SOAP. The current

130 SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit

131 the applicability of this specification to a single version of SOAP.

132 1.5 Normative References

133	[RFC2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.
134		http://www.ietf.org/rfc/rfc2119.txt
135		http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
136		
137	[SOAP]	W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.
138		http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
139		
140	[SOAP12]	W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003.
141		http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
142		
143		
144	[SOAPNorm]	W3C Working Group Note, "SOAP Version 1.2 Message Normalization", 8 October 2003.
145		http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/
146		
147		

148	[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
149		
150		
151		http://www.ietf.org/rfc/rfc3986.txt
152		
153	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
154		
155		http://www.ietf.org/rfc/rfc2068.txt
156		
157	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
158		http://www.ietf.org/rfc/rfc2246.txt
159		
160	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
161		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
162		
163	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
164		
165		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
166		
167	[WS-Policy]	W3C Recommendation, "Web Services Policy 1.5 - Framework", 04 September 2007.
168		
169		http://www.w3.org/TR/2007/REC-ws-policy-20070904/
170		W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
171		
172		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
173		
174	[WS-PolicyAttachment]	W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04 September 2007.
175		
176		http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
177		W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
178		
179		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
180		
181		
182	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.4", 2008
183		http://docs.oasis-open.org/ws-sx/ws-trust/200802
184		OASIS Standard, "WS-Trust 1.3", March 2007
185		http://docs.oasis-open.org/ws-sx/ws-trust/200512
186		
187	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.4", July 2008
188		http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
189		
190	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
191		

192		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
193		
194		
195	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
196		
197		http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
198		
199		
200	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004
201		
202		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
203		
204		
205	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
206		
207		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
208		
209		
210	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
211		
212		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
213		
214		
215	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
216		
217		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
218		
219		
220	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
221		
222		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
223		
224		
225	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
226		
227		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
228		
229	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
230		
231		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
232		
233		
234	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
235		
236		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
237		

238	[WSS:RELTOKENProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile 1.1", February 2006
239		
240		http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
241		
242		
243	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1", February 2006
244		
245		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
246		
247		
248	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
249		
250		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
251		
252	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002.
253		
254		http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/
255		
256		W3C Recommendation, D. Eastlake et al. XML Signature Syntax and Processing (Second Edition). 10 June 2008.
257		
258		http://www.w3.org/TR/2008/REC-xmlsig-core-20080610/
259		
260		
261	[XPath]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999.
262		
263		http://www.w3.org/TR/1999/REC-xpath-19991116
264		
265	[XPath 2.0 Filter]	W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November 2002.
266		
267		http://www.w3.org/TR/2002/REC-xmlsig-filter2-20021108/
268		
269	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition", 28 October 2004.
270		
271		http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
272		
273	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition", 28 October 2004.
274		
275		http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
276		

277 1.6 Non-Normative References

278 None.

279

280

2 Security Policy Model

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

285

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

290

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

301

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

2.1 Security Assertion Model

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

313

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

316

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

320

321

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

327

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

332

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

335 **2.2 Nested Policy Assertions**

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

342 **2.3 Security Binding Abstraction**

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

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

348

349 A binding defines the following security characteristics:

- 350 • The minimum set of tokens that will be used and how they are bound to messages. Note that
351 services might accept messages containing more tokens than those specified in policy.
- 352 • Any necessary key transport mechanisms
- 353 • Any REQUIRED message elements (e.g. timestamps) in the `wsse:Security` header.
- 354 • The content and ordering of elements in the `wsse:Security` header. Elements not specified in
355 the binding are not allowed.
- 356 • Various parameters, including those describing the algorithms to be used for canonicalization,
357 signing and encryption.

358

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

365

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

- 369 • TransportBinding (Section 7.3)
- 370 • SymmetricBinding (Section 7.4)
- 371 • AsymmetricBinding (Section 7.5)

372 3 Policy Considerations

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

375 3.1 Nested Policy

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

378

379 3.2 Policy Subjects

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

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

384 [Message Policy Subject]

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

387

388 wsdl:message

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

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

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

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

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

397 [Operation Policy Subject]

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

399 wsdl:portType/wsdl:operation

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

402 wsdl:binding/wsdl:operation

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

405

406

407 [Endpoint Policy Subject]

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

410 wsdl:portType

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

413 wsdl:binding

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

416 wsdl:port

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

419 4 Protection Assertions

420 The following assertions are used to identify *what* is being protected and the level of protection provided.
421 These assertions SHOULD apply to [Message Policy Subject]. These assertions MAY apply to [Endpoint
422 Policy Subject] or [Operation Policy Subject]. Where they apply to [Operation Policy Subject] they apply to
423 all messages of that operation. Where they apply to [Endpoint Policy Subject] they apply to all operations
424 of that endpoint.

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

427 4.1 Integrity Assertions

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

431 4.1.1 SignedParts Assertion

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

437
438 There MAY be multiple SignedParts assertions present. Multiple SignedParts assertions present within a
439 policy alternative are equivalent to a single SignedParts assertion containing the union of all specified
440 message parts. Note that this assertion does not require that a given part appear in a message, just that if
441 such a part appears, it requires integrity protection.

442 Syntax

```
443 <sp:SignedParts xmlns:sp="..." ... >  
444   <sp:Body />?  
445   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
446   <sp:Attachments>  
447     <sp13:ContentSignatureTransform /> ?  
448     <sp13:AttachmentCompleteSignatureTransform /> ?  
449   </sp:Attachments> ?  
450   ...  
451 </sp:SignedParts>
```

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

454 /sp:SignedParts

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

458 /sp:SignedParts/sp:Body

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

461 /sp:SignedParts/sp:Header

462 Presence of this OPTIONAL element indicates a specific SOAP header, its attributes and content
463 (or set of such headers) needs to be protected. There may be multiple sp:Header elements within
464 a single sp:SignedParts element. If multiple SOAP headers with the same local name but
465 different namespace names are to be integrity protected multiple sp:Header elements are
466 needed, either as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts
467 assertions.

468 This element only applies to SOAP header elements targeted to the same actor/role as the
469 Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
470 SOAP Header elements targeted to a different actor/role, that may be accomplished using the
471 sp:SignedElements assertion.

472 /sp:SignedParts/sp:Header/@Name

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

476 /sp:SignedParts/sp:Header/@Namespace

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

479 /sp:SignedParts/sp:Attachments

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

484 /sp:SignedParts/sp:Attachments/sp13:ContentSignatureTransform

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

487 /sp:SignedParts/sp:Attachments/sp13:AttachmentCompleteSignatureTransform

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

490 This is the default if neither sp13:ContentSignatureTransform or
491 sp13:AttachmentCompleteSignatureTransform are specified.

492 4.1.2 SignedElements Assertion

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

498

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

502 Syntax

```
503 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
504 <sp:XPath>xs:string</sp:XPath>+  
505 <sp13:Xpath2 Filter="xs:string">xs:string</sp13:Xpath2>+  
506 ...  
507 </sp:SignedElements>
```

508 The following describes the attributes and elements listed in the schema outlined above:
509 /sp:SignedElements
510 This assertion specifies the parts of the message that need integrity protection.
511 /sp:SignedElements/@XPathVersion
512 This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
513 attribute is provided, then XPath 1.0 is assumed.
514 /sp:SignedElements/sp:XPath
515 This element contains a string specifying an XPath expression that identifies the nodes to be
516 integrity protected. The XPath expression is evaluated against the S:Envelope element node of
517 the message. Multiple instances of this element MAY appear within this assertion and SHOULD
518 be treated as separate references in a signature when message security is used.
519 /sp:SignedElements/sp:XPath2
520 This element contains a string specifying an XPath 2 expression that identifies the nodes to be
521 integrity protected. The XPath expression is evaluated against the S:Envelope element node of
522 the message. Multiple instances of this element MAY appear within this assertion and SHOULD
523 be treated as separate references in a signature when message security is used.
524 /sp:SignedElements/sp:XPath2@Filter
525 This REQUIRED attribute contains a string to specify an [XPath Filter 2.0] transform to apply.

526 4.2 Confidentiality Assertions

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

530 4.2.1 EncryptedParts Assertion

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

536
537 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present
538 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all
539 specified message parts. Note that this assertion does not require that a given part appear in a message,
540 just that if such a part appears, it requires confidentiality protection.

541 Syntax

```
542 <sp:EncryptedParts xmlns:sp="..." ... >  
543   <sp:Body />?  
544   <sp:Header Name="xs:NCName" ? Namespace="xs:anyURI" ... />?  
545   <sp:Attachments />?  
546   ...  
547 </sp:EncryptedParts>
```

548
549 The following describes the attributes and elements listed in the schema outlined above:
550 /sp:EncryptedParts

551 This assertion specifies the parts of the message that need confidentiality protection. The single
552 child element of this assertion specifies the set of message parts using an extensible dialect.
553 If no child elements are specified, the body of the message MUST be confidentiality protected.
554 /sp:EncryptedParts/sp:Body
555 Presence of this OPTIONAL empty element indicates that the entire body of the message needs
556 to be confidentiality protected. In the case where mechanisms from WSS: SOAP Message
557 Security are used to satisfy this assertion, then the soap:Body element is encrypted using the
558 #Content encryption type.

559 /sp:EncryptedParts/sp:Header
560 Presence of this OPTIONAL element indicates that a specific SOAP header (or set of such
561 headers) needs to be protected. There may be multiple sp:Header elements within a single Parts
562 element. Each header or set of headers MUST be encrypted. Such encryption will encrypt such
563 elements using WSS 1.1 Encrypted Headers. As such, if WSS 1.1 Encrypted Headers are not
564 supported by a service, then this element cannot be used to specify headers that require
565 encryption using message level security. If multiple SOAP headers with the same local name but
566 different namespace names are to be encrypted then multiple sp:Header elements are needed,
567 either as part of a single sp:EncryptedParts assertion or as part of separate sp:EncryptedParts
568 assertions.

569 /sp:EncryptedParts/sp:Header/@Name
570 This OPTIONAL attribute indicates the local name of the SOAP header to be confidentiality
571 protected. If this attribute is not specified, all SOAP headers whose namespace matches the
572 Namespace attribute are to be protected.

573 /sp:EncryptedParts/sp:Header/@Namespace
574 This REQUIRED attribute indicates the namespace of the SOAP header(s) to be confidentiality
575 protected.

576 /sp:EncryptedParts/sp:Attachments
577 Presence of this OPTIONAL empty element indicates that all SwA (SOAP Messages with
578 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message
579 Security is used to accomplish this, all message parts other than the part containing the primary
580 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security
581 [WSS:SwAProfile1.1].

582 4.2.2 EncryptedElements Assertion

583 The EncryptedElements assertion is used to specify arbitrary elements in the message that require
584 confidentiality protection. This assertion can be satisfied using WSS: SOAP Message Security
585 mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the
586 message over a secure transport protocol like HTTPS. The binding specific token properties detail the
587 exact mechanism by which the protection is provided.
588

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

592 Syntax

```
593 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
594   <sp:XPath>xs:string</sp:XPath>+
595   ...
596 </sp:EncryptedElements>
```

597 The following describes the attributes and elements listed in the schema outlined above:
598 /sp:EncryptedElements
599 This assertion specifies the parts of the message that need confidentiality protection. Any such
600 elements are subject to #Element encryption.
601 /sp:EncryptedElements/@XPathVersion
602 This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
603 attribute is provided, then XPath 1.0 is assumed.
604 /sp:EncryptedElements/sp:XPath
605 This element contains a string specifying an XPath expression that identifies the nodes to be
606 confidentiality protected. The XPath expression is evaluated against the S:Envelope element
607 node of the message. Multiple instances of this element MAY appear within this assertion and
608 SHOULD be treated as separate references.

609 **4.2.3 ContentEncryptedElements Assertion**

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

615
616 There MAY be multiple ContentEncryptedElements assertions present. Multiple
617 ContentEncryptedElements assertions present within a policy alternative are equivalent to a single
618 ContentEncryptedElements assertion containing the union of all specified XPath expressions.

619 **Syntax**

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

624 The following describes the attributes and elements listed in the schema outlined above:
625 /sp:ContentEncryptedElements
626 This assertion specifies the parts of the message that need confidentiality protection. Any such
627 elements are subject to #Content encryption.
628 /sp:ContentEncryptedElements/@XPathVersion
629 This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no
630 attribute is provided, then XPath 1.0 is assumed.
631 /sp:ContentEncryptedElements/sp:XPath
632 This element contains a string specifying an XPath expression that identifies the nodes to be
633 confidentiality protected. The XPath expression is evaluated against the S:Envelope element
634 node of the message. Multiple instances of this element MAY appear within this assertion and
635 SHOULD be treated as separate references.

636 **4.3 Required Elements Assertion**

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

639

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

643 4.3.1 RequiredElements Assertion

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

646

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

650 Syntax

```
651 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
652 <sp:XPath>xs:string</sp:XPath> +  
653 ...  
654 </sp:RequiredElements>
```

655

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

657 /sp:RequiredElements

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

659 /sp:RequiredElements/@XPathVersion

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

662 /sp:RequiredElements/sp:XPath

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

667 4.3.2 RequiredParts Assertion

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

671

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

675 Syntax

```
676 <sp:RequiredParts XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
677 <sp:Header Name="..." Namespace="..." /> +  
678 </sp:RequiredParts>
```

679

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

681 /sp:RequiredParts/sp:Header

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

683 /sp:RequiredParts/sp:Header/@Name

684 This REQUIRED attribute indicates the local name of the SOAPHeader that needs to be present
685 in the message.
686 /sp:RequiredParts/sp:Header/@Namespace
687 This REQUIRED attribute indicates the namespace of the SOAP header that needs to be present
688 in the message.

689

5 Token Assertions

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

5.1 Token Inclusion

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

5.1.1 Token Inclusion Values

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

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

702

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

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

709 **5.1.2 Token Inclusion and Token References**

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

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

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

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

726 **5.2 Token Issuer and Required Claims**

727 **5.2.1 Token Issuer**

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

732 **5.2.2 Token Issuer Name**

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

737
738 It is out of scope of this specification how the relationship between the issuer's logical name and the
739 physical manifestation of the issuer in the security token is defined.

740 While both sp:Issuer and sp:IssuerName elements are OPTIONAL they are also mutually exclusive and
741 cannot be specified both at the same time.

742 **5.2.3 Required Claims**

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

746
747 This element indicates the REQUIRED claims that the security token must contain in order to satisfy the
748 requirements of the token assertion.

749
750 Individual token assertions MAY further limit what claims MAY be specified for that specific token
751 assertion.

752 5.2.4 Processing Rules and Token Matching

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

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

- 760
761 1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and
762 T2 is issued by issuer B and contains claims C3 and C4.
- 763 2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is
764 also issued by issuer A and contains claim C2 and T3 is issued by issuer B and
765 contains claims C3 and C4.
- 766 3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2,
767 T2 is issued by issuer B and contains claim C3 and T3 is also issued by issuer B and
768 contains claim C4.
- 769 4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is
770 also issued by issuer A and contains claim C2, T3 is issued by issuer B and contains
771 claim C3 and T4 is also issued by issuer B and contains claim C4.

772 5.3 Token Properties

773 5.3.1 [Derived Keys] Property

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

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

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

782 5.3.2 [Explicit Derived Keys] Property

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

786 5.3.3 [Implied Derived Keys] Property

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

790 5.4 Token Assertion Types

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

792 5.4.1 UsernameToken Assertion

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

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

- 795 1. When transport security is not used.
- 796 2. When a plaintext password is used.
- 797 3. When a weak password hash is used.
- 798 4. When the username needs to be protected, e.g. for privacy reasons.

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

802

803 Syntax

```
804 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
805 (   
806   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
807   <sp:IssuerName>xs:anyURI</sp:IssuerName>   
808 ) ?   
809 <wst:Claims Dialect="..."> ... </wst:Claims> ?   
810 <wsp:Policy xmlns:wsp="...">   
811   ( (   
812     <sp:NoPassword ... /> |   
813     <sp:HashPassword ... />   
814   ) |   
815   (   
816     <sp13:Created .../> ?   
817     <sp13:Nonce .../> ?   
818   ) ) ?   
819   (   
820     <sp:RequireDerivedKeys /> |   
821     <sp:RequireImpliedDerivedKeys ... /> |   
822     <sp:RequireExplicitDerivedKeys ... />   
823   ) ?   
824   (   
825     <sp:WssUsernameToken10 ... /> |   
826     <sp:WssUsernameToken11 ... />   
827   ) ?   
828   ...   
829 </wsp:Policy>   
830 ...   
831 </sp:UsernameToken>
```

832

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

834 /sp:UsernameToken

835 This identifies a UsernameToken assertion.

836 /sp:UsernameToken/@sp:IncludeToken

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

838 /sp:UsernameToken/sp:Issuer

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

841 /sp:UsernameToken/sp:IssuerName

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

844 /sp:UsernameToken/wst:Claims

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

847 /sp:UsernameToken/wsp:Policy

848 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken
849 assertion.

850 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

853 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

858 /sp13:UsernameToken/wsp:Policy/sp13:Created

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

862 /sp13:UsernameToken/wsp:Policy/sp13:Nonce

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

866 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

869 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

873 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

877 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

880 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

883 5.4.2 ICreatessuedToken Assertion

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

888 Syntax

```

889 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
890 (
891 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
892 <sp:IssuerName>xs:anyURI</sp:IssuerName>
893 ) ?
894 <wst:Claims Dialect="..."> ... </wst:Claims> ?
895 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
896 ...
897 </sp:RequestSecurityTokenTemplate>
898 <wsp:Policy xmlns:wsp="...">
899 (
900 <sp:RequireDerivedKeys ... /> |
901 <sp:RequireImpliedDerivedKeys ... /> |
902 <sp:RequireExplicitDerivedKeys ... />
903 ) ?
904 <sp:RequireExternalReference ... /> ?
905 <sp:RequireInternalReference ... /> ?
906 ...
907 </wsp:Policy>
908 ...
909 </sp:IssuedToken>

```

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

911 /sp:IssuedToken

912 This identifies an IssuedToken assertion.

913 /sp:IssuedToken/@sp:IncludeToken

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

915 /sp:IssuedToken/sp:Issuer

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

918 /sp:IssuedToken/sp:IssuerName

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

921 /sp:IssuedToken/wst:Claims

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

924 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

929 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

930 This OPTIONAL attribute contains a WS-Trust specification namespace URI identifying the
931 version of WS-Trust referenced by the contents of this element. For example, when using Trust
932 1.3 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200512> should be used and when using
933 Trust 1.4 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200802> should be used.

934 /sp:IssuedToken/wsp:Policy

935 This REQUIRED element identifies additional requirements for use of the sp:IssuedToken
936 assertion.

937 /sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys

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

940 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

944 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

948 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

949 This OPTIONAL element is a policy assertion that indicates whether an internal reference is
950 REQUIRED when referencing this token.

951 Note: This reference will be supplied by the issuer of the token.

952 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

953 This OPTIONAL element is a policy assertion that indicates whether an external reference is
954 REQUIRED when referencing this token.

955 Note: This reference will be supplied by the issuer of the token.

956 Note: The IssuedToken MAY or MAY NOT be associated with key material and such key material may be
957 symmetric or asymmetric. The Binding assertion will imply the type of key associated with this token.

958 Services MAY also include information in the sp:RequestSecurityTokenTemplate element to
959 explicitly define the expected key type. See [Appendix B](#) for details of the

960 sp:RequestSecurityTokenTemplate element.

961 5.4.3 X509Token Assertion

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

963 Syntax

```
964 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
965   (  
966     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
967     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
968   ) ?  
969   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```



```

970 <wsp:Policy xmlns:wsp="...">
971   (
972     <sp:RequireDerivedKeys ... /> |
973     <sp:RequireExplicitDerivedKeys ... /> |
974     <sp:RequireImpliedDerivedKeys ... />
975   ) ?
976 <sp:RequireKeyIdentifierReference ... /> ?
977 <sp:RequireIssuerSerialReference ... /> ?
978 <sp:RequireEmbeddedTokenReference ... /> ?
979 <sp:RequireThumbprintReference ... /> ?
980   (
981     <sp:WssX509V3Token10 ... /> |
982     <sp:WssX509Pkcs7Token10 ... /> |
983     <sp:WssX509PkiPathV1Token10 ... /> |
984     <sp:WssX509V1Token11 ... /> |
985     <sp:WssX509V3Token11 ... /> |
986     <sp:WssX509Pkcs7Token11 ... /> |
987     <sp:WssX509PkiPathV1Token11 ... />
988   ) ?
989   ...
990 </wsp:Policy>
991 ...
992 </sp:X509Token>

```

- 993
- 994 The following describes the attributes and elements listed in the schema outlined above:
- 995 /sp:X509Token
- 996 This identifies an X509Token assertion.
- 997 /sp:X509Token/@sp:IncludeToken
- 998 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 999 /sp:X509Token/sp:Issuer
- 1000 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
- 1001 of the sp:X509Token.
- 1002 /sp:X509Token/sp:IssuerName
- 1003 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:X509Token
- 1004 issuer.
- 1005 /sp:X509Token/wst:Claims
- 1006 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
- 1007 order to satisfy the token assertion requirements.
- 1008 /sp:X509Token/wsp:Policy
- 1009 This REQUIRED element identifies additional requirements for use of the sp:X509Token
- 1010 assertion.
- 1011 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys
- 1012 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
- 1013 and [Implied Derived Keys] properties for this token to 'true'.
- 1014 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1015 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
- 1016 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
- 1017 'false'.
- 1018 /sp:X509Token/wsp:Policy/sp:RequireImpliedDerivedKeys

1019 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1020 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1021 'false'.

1022 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

1023 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1024 REQUIRED when referencing this token.

1025 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

1026 This OPTIONAL element is a policy assertion that indicates that an issuer serial reference is
 1027 REQUIRED when referencing this token.

1028 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

1029 This OPTIONAL element is a policy assertion that indicates that an embedded token reference is
 1030 REQUIRED when referencing this token.

1031 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1032 This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is
 1033 REQUIRED when referencing this token.

1034 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1035 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
 1036 be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1037 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1038 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
 1039 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1040 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1041 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
 1042 token should be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1043 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1044 This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should
 1045 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1046 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1047 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
 1048 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1049 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1050 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
 1051 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1052 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1053 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
 1054 token should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1055 **5.4.4 KerberosToken Assertion**

1056 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

1057 **Syntax**

```
1058 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1059 (
1060   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1061   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1062 ) ?
```

```

1063 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1064 <wsp:Policy xmlns:wsp="...">
1065   (
1066     <sp:RequireDerivedKeys ... /> |
1067     <sp:RequireImpliedDerivedKeys ... /> |
1068     <sp:RequireExplicitDerivedKeys ... />
1069   ) ?
1070 <sp:RequireKeyIdentifierReference ... /> ?
1071   (
1072     <sp:WssKerberosV5ApReqToken11 ... /> |
1073     <sp:WssGssKerberosV5ApReqToken11 ... />
1074   ) ?
1075
1076   ...
1077 </wsp:Policy>
1078   ...
1079 </sp:KerberosToken>

```

1080

1081 The following describes the attributes and elements listed in the schema outlined above:

1082 /sp:KerberosToken

1083 This identifies a KerberosV5ApReqToken assertion.

1084 /sp:KerberosToken/@sp:IncludeToken

1085 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1086 /sp:KerberosToken/sp:Issuer

1087 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1088 of the sp:KerberosToken.

1089 /sp:KerberosToken/sp:IssuerName

1090 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken
1091 issuer.

1092 /sp:KerberosToken/wst:Claims

1093 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1094 order to satisfy the token assertion requirements.

1095 /sp:KerberosToken/wsp:Policy

1096 This REQUIRED element identifies additional requirements for use of the sp:KerberosToken
1097 assertion.

1098 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1099 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1100 and [Implied Derived Keys] properties for this token to 'true'.

1101 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1102 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1103 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1104 'false'.

1105 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1106 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1107 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1108 'false'.

1109 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1110 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1111 REQUIRED when referencing this token.

1112 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

1113 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ
1114 token should be used as defined in [WSS:KerberosTokenProfile1.1].

1115 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1116 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-
1117 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

1118 5.4.5 SpnegoContextToken Assertion

1119 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
1120 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1121 Syntax

```
1122 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1123   (  
1124     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1125     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1126   ) ?  
1127   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1128   <wsp:Policy xmlns:wsp="...">  
1129     (  
1130       <sp:RequireDerivedKeys ... /> |  
1131       <sp:RequireImpliedDerivedKeys ... /> |  
1132       <sp:RequireExplicitDerivedKeys ... />  
1133     ) ?  
1134     <sp:MustNotSendCancel ... /> ?  
1135     <sp:MustNotSendAmend ... /> ?  
1136     <sp:MustNotSendRenew ... /> ?  
1137     ...  
1138   </wsp:Policy>  
1139   ...  
1140 </sp:SpnegoContextToken>
```

1141

1142 The following describes the attributes and elements listed in the schema outlined above:

1143 /sp:SpnegoContextToken

1144 This identifies a SpnegoContextToken assertion.

1145 /sp:SpnegoContextToken/@sp:IncludeToken

1146 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1147 /sp:SpnegoContextToken/sp:Issuer

1148 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
1149 for the Spnego Context Token.

1150 /sp:SpnegoContextToken/sp:IssuerName

1151 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
1152 sp:SpnegoContextToken issuer.

1153 /sp:SpnegoContextToken/wst:Claims

1154 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1155 order to satisfy the token assertion requirements.

1156 /sp:SpnegoContextToken/wsp:Policy

1157 This REQUIRED element identifies additional requirements for use of the
 1158 sp:SpnegoContextToken assertion.

1159 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1160 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1161 and [Implied Derived Keys] properties for this token to 'true'.

1162 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1163 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1164 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1165 'false'.

1166 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1167 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1168 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1169 'false'.

1170 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1171 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1172 token does not support SCT/Cancel RST messages. If this assertion is missing it means that
 1173 SCT/Cancel RST messages are supported by the STS.

1174 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1175 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1176 token does not support SCT/Amend RST messages. If this assertion is missing it means that
 1177 SCT/Amend RST messages are supported by the STS.

1178 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1179 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1180 token does not support SCT/Renew RST messages. If this assertion is missing it means that
 1181 SCT/Renew RST messages are supported by the STS.

1182 5.4.6 SecurityContextToken Assertion

1183 This element represents a requirement for a SecurityContextToken token.

1184 Syntax

```

1185 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1186 (
1187   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1188   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1189 ) ?
1190 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1191 <wsp:Policy xmlns:wsp="...">
1192   (
1193     <sp:RequireDerivedKeys ... /> |
1194     <sp:RequireImpliedDerivedKeys ... /> |
1195     <sp:RequireExplicitDerivedKeys ... />
1196   ) ?
1197   <sp:RequireExternalUriReference ... /> ?
1198   <sp:SC13SecurityContextToken... /> ?
1199   ...
1200 </wsp:Policy>
1201 ...
1202 </sp:SecurityContextToken>

```

1203

1204 The following describes the attributes and elements listed in the schema outlined above:

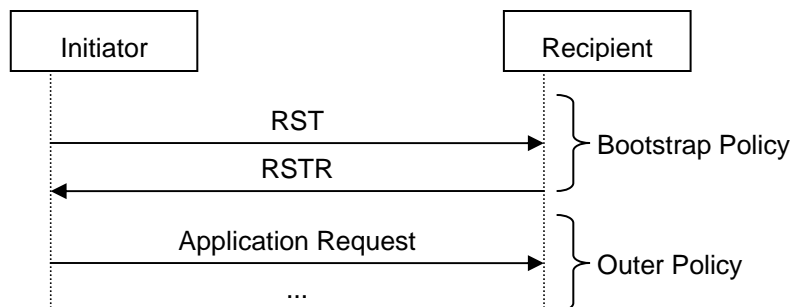
1205 /sp:SecurityContextToken

- 1206 This identifies a SecurityContextToken assertion.
- 1207 /sp:SecurityContextToken/@sp:IncludeToken
- 1208 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 1209 /sp:SecurityContextToken/sp:Issuer
- 1210 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
- 1211 of the sp:SecurityContextToken.
- 1212 /sp:SecurityContextToken/sp:IssuerName
- 1213 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
- 1214 sp:SecurityContextToken issuer.
- 1215 /sp:SecurityContextToken/wst:Claims
- 1216 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
- 1217 order to satisfy the token assertion requirements.
- 1218 /sp:SecurityContextToken/wsp:Policy
- 1219 This REQUIRED element identifies additional requirements for use of the
- 1220 sp:SecurityContextToken assertion.
- 1221 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys
- 1222 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
- 1223 and [Implied Derived Keys] properties for this token to 'true'.
- 1224 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1225 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
- 1226 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
- 1227 'false'.
- 1228 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1229 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
- 1230 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
- 1231 'false'.
- 1232 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference
- 1233 This OPTIONAL element is a policy assertion that indicates that an external URI reference is
- 1234 REQUIRED when referencing this token.
- 1235 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken
- 1236 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
- 1237 be used as defined in [[WS-SecureConversation](#)].
- 1238
- 1239 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
- 1240 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
- 1241 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
- 1242 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

1243 **5.4.7 SecureConversationToken Assertion**

- 1244 This element represents a requirement for a Security Context Token retrieved from the indicated issuer
- 1245 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
- 1246 service endpoint address.
- 1247

1248 Note: This assertion describes the token accepted by the target service. Because this token is issued by
 1249 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD
 1250 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
 1251 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
 1252 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
 1253 below.



1254

1255

1256 If the bootstrap policy assertion is used to indicate the security binding and policy in effect when
 1257 requesting a secure conversation token from the target service, then subsequent Amend, Renew and
 1258 Cancel messages MUST comply with the following rules.

1259 **Amending Context**

1260 To amend an existing secure conversation token, a requestor uses the context amending mechanism as
 1261 described by the WS-SecureConversation specification. The message exchange MUST be secured
 1262 using the existing (to be amended) SCT in accordance with the target service (outer) policy, combined
 1263 with endorsing supporting tokens carrying the new claims to be associated with the amended context with
 1264 the inclusion mode set to:

1265 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1266 See the EndorsingSupportingTokens Assertion section for more details on the usage of the endorsing
 1267 supporting tokens.

1268 **Renewing Context**

1269 To renew an existing secure conversation token, a requestor uses the context renewal mechanism as
 1270 described by the WS-SecureConversation specification. The message exchange MUST be secured
 1271 according to the requirements of the bootstrap policy assertion, combined with the existing (to be
 1272 renewed) SCT used as an endorsing supporting token with the inclusion mode set to:

1273 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1274 See the EndorsingSupportingTokens Assertion section for more details on the usage of endorsing
 1275 support tokens.

1276 **Canceling Context**

1277 To cancel an existing secure conversation token, a requestor uses the context cancelling mechanism as
 1278 described by the WS-SecureConversation specification. The message exchange MUST be secured
 1279 using the existing (to be cancelled) SCT in accordance with the target service (outer) policy.

1280 **Handling Policy Alternatives**

1281 If there are policy alternatives present in either the bootstrap policy assertion or the target service (outer)
 1282 policy assertion, the following rules MUST be followed.

- 1283 • The policy alternative used as a basis for the context renewal MUST be the same as the policy
 1284 alternative which was previously used for the context issuance.

- If the target service (outer) policy has policy alternatives and SecureConversationToken assertion appears in multiple alternatives as follows:

1287 Policy

1288 Policy-alternative-1

1289 SecureConversationToken-assertion-1

1290 Policy-alternative-2

1291 SecureConversationToken-assertion-2

1292 The policy alternative used as basis for context amend and cancel MUST be the same as the policy
1293 alternative that was used to obtain the context. This means that Policy-alternative-1 above cannot be
1294 used to amend and cancel SecureConversationToken-assertion-2 and vice-versa.

- If the target service (outer) policy has policy alternatives that are outside the SecureConversationToken assertion as follows:

1297 Policy

1298 SecureConversationToken-assertion-1

1299 Policy-alternative-1

1300 Policy-alternative-2

1301 Any policy alternative can be used to amend or cancel the context. This means that either Policy-
1302 alternative-1 or Policy-alternative-2 can be used to amend or cancel SecureConversationToken-
1303 assertion-1.

1304

1305 Syntax

```
1306 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1307 (  
1308 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1309 <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1310 ) ?  
1311 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1312 <wsp:Policy xmlns:wsp="...">  
1313 (  
1314 <sp:RequireDerivedKeys ... /> |  
1315 <sp:RequireImpliedDerivedKeys ... /> |  
1316 <sp:RequireExplicitDerivedKeys ... />  
1317 ) ?  
1318 <sp:RequireExternalUriReference ... /> ?  
1319 <sp:SC13SecurityContextToken ... /> ?  
1320 <sp:MustNotSendCancel ... /> ?  
1321 <sp:MustNotSendAmend ... /> ?  
1322 <sp:MustNotSendRenew ... /> ?  
1323 <sp:BootstrapPolicy ... >  
1324 <wsp:Policy> ... </wsp:Policy>  
1325 </sp:BootstrapPolicy> ?  
1326 </wsp:Policy>  
1327 ...  
1328 </sp:SecureConversationToken>
```

1329

1330 The following describes the attributes and elements listed in the schema outlined above:

1331 /sp:SecureConversationToken

1332 This identifies a SecureConversationToken assertion.

1333 /sp:SecureConversationToken/@sp:IncludeToken

1334 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1335 /sp:SecureConversationToken/sp:Issuer
1336 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
1337 for the Security Context Token.

1338 /sp:SecureConversationToken/sp:IssuerName
1339 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
1340 sp:SecureConversationToken issuer.

1341 /sp:SpnegoContextToken/wst:Claims
1342 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1343 order to satisfy the token assertion requirements.

1344 /sp:SecureConversationToken/wsp:Policy
1345 This REQUIRED element identifies additional requirements for use of the
1346 sp:SecureConversationToken assertion.

1347 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys
1348 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1349 and [Implied Derived Keys] properties for this token to 'true'.

1350 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys
1351 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1352 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1353 'false'.

1354 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys
1355 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1356 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1357 'false'.

1358 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference
1359 This OPTIONAL element is a policy assertion that indicates that an external URI reference is
1360 REQUIRED when referencing this token.

1361 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken
1362 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
1363 be used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1364 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel
1365 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1366 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it
1367 means that SCT/Cancel RST messages are supported by the STS.

1368 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend
1369 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1370 conversation token does not support SCT/Amend RST messages. If this assertion is missing it
1371 means that SCT/Amend RST messages are supported by the STS.

1372 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew
1373 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
1374 conversation token does not support SCT/Renew RST messages. If this assertion is missing it
1375 means that SCT/Renew RST messages are supported by the STS.

1376 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy
1377 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements
1378 for obtaining the Security Context Token.

1379 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy
1380 This element contains the security binding requirements for obtaining the Security Context Token.
1381 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
1382 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
1383 are to be protected.

1384 Example

```
1385 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
1386   <sp:SymmetricBinding>  
1387     <wsp:Policy>  
1388       <sp:ProtectionToken>  
1389         <wsp:Policy>  
1390           <sp:SecureConversationToken>  
1391             <sp:Issuer>  
1392               <wsa:Address>http://example.org/sts</wsa:Address>  
1393             </sp:Issuer>  
1394           <wsp:Policy>  
1395             <sp:SC13SecurityContextToken />  
1396           <sp:BootstrapPolicy>  
1397             <wsp:Policy>  
1398               <sp:AsymmetricBinding>  
1399                 <wsp:Policy>  
1400                   <sp:InitiatorToken>  
1401                     ...  
1402                   </sp:InitiatorToken>  
1403                   <sp:RecipientToken>  
1404                     ...  
1405                   </sp:RecipientToken>  
1406                 </wsp:Policy>  
1407               </sp:AsymmetricBinding>  
1408             <sp:SignedParts>  
1409               ...  
1410             </sp:SignedParts>  
1411             ...  
1412           </wsp:Policy>  
1413         </sp:BootstrapPolicy>  
1414       </wsp:Policy>  
1415     </sp:SecureConversationToken>  
1416   </wsp:Policy>  
1417 </sp:ProtectionToken>  
1418   ...  
1419 </wsp:Policy>  
1420 </sp:SymmetricBinding>  
1421 <sp:SignedParts>  
1422   ...  
1423 </sp:SignedParts>  
1424   ...  
1425 </wsp:Policy>
```

1426 5.4.8 SamlToken Assertion

1427 This element represents a requirement for a SAML token.

1428 Syntax

```
1429 <sp:SamlToken sp:IncludeToken="xs:anyURI" ? xmlns:sp="..." ... >  
1430   (  
1431     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1432     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1433   ) ?  
1434   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```

1435 <wsp:Policy xmlns:wsp="...">
1436   (
1437     <sp:RequireDerivedKeys ... /> |
1438     <sp:RequireImpliedDerivedKeys ... /> |
1439     <sp:RequireExplicitDerivedKeys ... />
1440   ) ?
1441   <sp:RequireKeyIdentifierReference ... /> ?
1442   (
1443     <sp:WssSamlV11Token10 ... /> |
1444     <sp:WssSamlV11Token11 ... /> |
1445     <sp:WssSamlV20Token11 ... />
1446   ) ?
1447   ...
1448 </wsp:Policy>
1449 ...
1450 </sp:SamlToken>

```

- 1451
- 1452 The following describes the attributes and elements listed in the schema outlined above:
- 1453 /sp:SamlToken
- 1454 This identifies a SamlToken assertion.
- 1455 /sp:SamlToken/@sp:IncludeToken
- 1456 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 1457 /sp:SamlToken/sp:Issuer
- 1458 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
- 1459 of the sp:SamlToken.
- 1460 /sp:SamlToken/sp:IssuerName
- 1461 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken
- 1462 issuer.
- 1463 /sp:SamlToken/wst:Claims
- 1464 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
- 1465 order to satisfy the token assertion requirements.
- 1466 /sp:SamlToken/wsp:Policy
- 1467 This REQUIRED element identifies additional requirements for use of the sp:SamlToken
- 1468 assertion.
- 1469 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys
- 1470 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
- 1471 and [Implied Derived Keys] properties for this token to 'true'.
- 1472 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1473 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
- 1474 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
- 1475 'false'.
- 1476 /sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1477 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
- 1478 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
- 1479 'false'.
- 1480 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference
- 1481 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
- 1482 REQUIRED when referencing this token.

1483 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10
1484 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1485 should be used as defined in [WSS:SAMLTOKENPROFILE1.0].

1486 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11
1487 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1488 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1489 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11
1490 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token
1491 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1492
1493 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1494 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1495 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion
1496 SHOULD be used instead.

1497 5.4.9 RelToken Assertion

1498 This element represents a requirement for a REL token.

1499 Syntax

```
1500 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1501 (   
1502   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
1503   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1504 ) ?  
1505 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1506 <wsp:Policy xmlns:wsp="...">  
1507 (   
1508   <sp:RequireDerivedKeys ... /> |   
1509   <sp:RequireImpliedDerivedKeys ... /> |   
1510   <sp:RequireExplicitDerivedKeys ... />  
1511 ) ?  
1512 <sp:RequireKeyIdentifierReference ... /> ?  
1513 (   
1514   <sp:WssRelV10Token10 ... /> |   
1515   <sp:WssRelV20Token10 ... /> |   
1516   <sp:WssRelV10Token11 ... /> |   
1517   <sp:WssRelV20Token11 ... />  
1518 ) ?  
1519 ...  
1520 </wsp:Policy>  
1521 ...  
1522 </sp:RelToken>
```

1523
1524 The following describes the attributes and elements listed in the schema outlined above:

1525 /sp:RelToken

1526 This identifies a RelToken assertion.

1527 /sp:RelToken/@sp:IncludeToken

1528 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1529 /sp:RelToken/sp:Issuer

1530 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1531 of the sp:RelToken.

1532 /sp:RelToken/sp:IssuerName
 1533 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken
 1534 issuer.

1535 /sp:RelToken/wst:Claims
 1536 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1537 order to satisfy the token assertion requirements.

1538 /sp:RelToken/wsp:Policy
 1539 This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

1540 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys
 1541 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1542 and [Implied Derived Keys] property for this token to 'true'.

1543 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1544 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1545 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1546 'false'.

1547 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1548 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1549 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1550 'false'.

1551 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1552 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1553 REQUIRED when referencing this token.

1554 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10
 1555 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1556 be used as defined in [\[WSS:RELTokenProfile1.0\]](#).

1557 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10
 1558 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1559 be used as defined in [\[WSS:RELTokenProfile1.0\]](#).

1560 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11
 1561 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1562 be used as defined in [\[WSS:RELTokenProfile1.1\]](#).

1563 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11
 1564 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1565 be used as defined in [\[WSS:RELTokenProfile1.1\]](#).

1566
 1567 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
 1568 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
 1569 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion
 1570 SHOULD be used instead.

1571 **5.4.10 HttpsToken Assertion**

1572 This element represents a requirement for a transport binding to support the use of HTTPS.

1573 **Syntax**

```

1574 <sp:HttpsToken xmlns:sp="..." ... >
1575 (
1576   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1577   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1578 ) ?
1579 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1580 <wsp:Policy xmlns:wsp="...">
1581 (
1582   <sp:HttpBasicAuthentication /> |
1583   <sp:HttpDigestAuthentication /> |
1584   <sp:RequireClientCertificate /> |
1585   ...
1586 )?
1587   ...
1588 </wsp:Policy>
1589   ...
1590 </sp:HttpsToken>

```

1591 The following describes the attributes and elements listed in the schema outlined above:

1592 /sp:HttpsToken

1593 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1594 supported.

1595 /sp:HttpsToken/sp:Issuer

1596 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1597 of the sp:HttpsToken.

1598 /sp:HttpsToken/sp:IssuerName

1599 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken
1600 issuer.

1601 /sp:HttpsToken/wst:Claims

1602 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1603 order to satisfy the token assertion requirements.

1604 /sp:HttpsToken/wsp:Policy

1605 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken
1606 assertion.

1607 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1608 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic
1609 Authentication [[RFC2068](#)] to authenticate to the service.

1610 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1611 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP
1612 Digest Authentication [[RFC2068](#)] to authenticate to the service.

1613 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1614 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a
1615 certificate when negotiating the HTTPS session.

1616 5.4.11 KeyValueToken Assertion

1617 This element represents a requirement for a KeyValue token. The next section defines the KeyValue
1618 security token abstraction for purposes of this token assertion.

1619

1620 This document defines requirements for KeyValueType when used in combination with RSA
1621 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by
1622 introducing new nested assertions besides *sp:RsaKeyValue*.

1623 Syntax

```
1624 <sp:KeyValueType sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1625 <wsp:Policy xmlns:wsp="...">  
1626 <sp:RsaKeyValue ... /> ?  
1627 ...  
1628 </wsp:Policy>  
1629 ...  
1630 </sp:KeyValueType>
```

1631 The following describes the attributes listed in the schema outlined above:

1632 /sp:KeyValueType

1633 This identifies a RsaToken assertion.

1634 /sp:KeyValueType/@sp:IncludeToken

1635 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1636 /sp:KeyValueType/wsp:Policy

1637 This REQUIRED element identifies additional requirements for use of the sp:KeyValueType
1638 assertion.

1639 /sp:KeyValueType/wsp:Policy/sp:RsaKeyValue

1640 This OPTIONAL element is a policy assertion that indicates that the ds:RSAKeyValue element
1641 must be present in the KeyValueType token. This indicates that an RSA key pair must be used.

1642 5.4.11.1 KeyValueType Token

1643 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key
1644 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this
1645 section is to define the KeyValueType token abstraction that represents such key pair referencing mechanism.

1646
1647 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be
1648 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*
1649 element in combination with RSA cryptographic algorithm.

1650
1651 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the
1652 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1653 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1654 <ds:KeyValue>  
1655 <ds:RSAKeyValue>  
1656 <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1657 <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1658 </ds:RSAKeyValue>  
1659 <ds:KeyValue>  
1660 </ds:KeyInfo>
```

1661
1662 When the KeyValueType token is used the corresponding public key value appears directly in the signature or
1663 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValueType token
1664 manifestation outside the *ds:KeyInfo* element.

```
1665 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1666 <SignedInfo>  
1667 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1668 c14n#" />  
1669 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1670 <Reference URI="#_1">  
1671 <Transforms>
```

```

1672     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1673   </Transforms>
1674   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1675   <DigestValue>...</DigestValue>
1676   </Reference>
1677 </SignedInfo>
1678 <SignatureValue>...</SignatureValue>
1679 <KeyInfo>
1680   <KeyValue>
1681     <RSAKeyValue>
1682       <Modulus>...</Modulus>
1683       <Exponent>...</Exponent>
1684     </RSAKeyValue>
1685   </KeyValue>
1686 </KeyInfo>
1687 </Signature>

```

1688
1689 Since there is no representation of the `KeyValue` token outside the `ds:KeyInfo` element and thus no
1690 identifier can be associated with the token, the `KeyValue` token cannot be referenced by using
1691 `wsse:SecurityTokenReference` element. However the `ds:KeyInfo` element representing the `KeyValue`
1692 token can be used whenever a security token can be used as illustrated on the following example:

```

1693 <t:RequestSecurityToken xmlns:t="...">
1694   <t:RequestType>...</t:RequestType>
1695   ...
1696   <t:UseKey>
1697     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1698       <KeyValue>
1699         <RSAKeyValue>
1700           <Modulus>...</Modulus>
1701           <Exponent>...</Exponent>
1702         </RSAKeyValue>
1703       </KeyValue>
1704     </KeyInfo>
1705   </t:UseKey>
1706 </t:RequestSecurityToken>

```


1707

6 Security Binding Properties

1708 This section defines the various properties or conditions of a security binding, their semantics, values and
1709 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are
1710 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that
1711 populates a value of a property appears in a policy, that property is set to the value indicated by the
1712 assertion. The security binding then uses the value of the property to control its behavior. The properties
1713 listed here are common to the various security bindings described in Section 7. Assertions that define
1714 values for these properties are defined in Section 7. The following properties are used by the security
1715 binding assertions.

1716 6.1 [Algorithm Suite] Property

1717 This property specifies the algorithm suite REQUIRED for performing cryptographic operations with
1718 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and
1719 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This
1720 property defines the set of available algorithms. The value of this property is typically referenced by a
1721 security binding and is used to specify the algorithms used for all message level cryptographic operations
1722 performed under the security binding.

1723 Note: In some cases, this property MAY be referenced under a context other than a security binding and
1724 used to control the algorithms used under that context. For example, supporting token assertions define
1725 such a context. In such contexts, the specified algorithms still apply to message level cryptographic
1726 operations.

1727 An algorithm suite defines values for each of the following operations and properties:

- 1728 • [Sym Sig] Symmetric Key Signature
- 1729 • [Asym Sig] Signature with an asymmetric key
- 1730 • [Dig] Digest
- 1731 • [Enc] Encryption
- 1732 • [Sym KW] Symmetric Key Wrap
- 1733 • [Asym KW] Asymmetric Key Wrap
- 1734 • [Comp Key] Computed key
- 1735 • [Enc KD] Encryption key derivation
- 1736 • [Sig KD] Signature key derivation
- 1737 • [Min SKL] Minimum symmetric key length
- 1738 • [Max SKL] Maximum symmetric key length
- 1739 • [Min AKL] Minimum asymmetric key length
- 1740 • [Max AKL] Maximum asymmetric key length

1741

1742 The following table provides abbreviations for the algorithm URI used in the table below:

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

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

1743

1744 The tables below show all the base algorithm suites defined by this specification. This table defines
 1745 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

1746 This table defines additional properties whose values can be specified along with the default value for that
 1747 property.

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

1748 This table defines values for the remaining components for each algorithm suite.

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

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

1749 6.2 [Timestamp] Property

1750 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`
 1751 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected
 1752 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be
 1753 present. The default value for this property is 'false'.

1754 6.3 [Protection Order] Property

1755 This property indicates the order in which integrity and confidentiality are applied to the message, in
 1756 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.

1757 The default value for this property is 'SignBeforeEncrypting'.

1758 6.4 [Signature Protection] Property

1759 This boolean property specifies whether the signature MUST be encrypted. If the value is 'true', the
 1760 primary signature MUST be encrypted and any signature confirmation elements MUST also be encrypted.
 1761 The primary signature element is NOT REQUIRED to be encrypted if the value is 'true' when there is
 1762 nothing in the message that is covered by this signature that is encrypted. If the value is 'false', the
 1763 primary signature MUST NOT be encrypted and any signature confirmation elements MUST NOT be
 1764 encrypted. The default value for this property is 'false'.

1765 6.5 [Token Protection] Property

1766 This boolean property specifies whether signatures MUST cover the token used to generate that
 1767 signature. If the value is 'true', then each token used to generate a signature MUST be covered by that
 1768 signature. If the value is 'false', then the token MUST NOT be covered by the signature. Note that in
 1769 cases where derived keys are used the 'main' token, and NOT the derived key token, is covered by the
 1770 signature. It is RECOMMENDED that assertions that define values for this property apply to [Endpoint
 1771 Policy Subject]. The default value for this property is 'false'.

1772 6.6 [Entire Header and Body Signatures] Property

1773 This boolean property specifies whether signature digests over the SOAP body and SOAP headers
1774 MUST only cover the entire body and entire header elements. If the value is 'true', then each digest over
1775 the SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In
1776 addition each digest over a SOAP header MUST be over an actual header element and not a descendant
1777 of a header element. This restriction does not specifically apply to the wsse:Security header. However
1778 signature digests over child elements of the wsse:Security header MUST be over the entire child element
1779 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a
1780 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to
1781 'true' mitigates against some possible re-writing attacks. It is RECOMENDED that assertions that define
1782 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

1783 6.7 [Security Header Layout] Property

1784 This property indicates which layout rules to apply when adding items to the security header. The
1785 following table shows which rules are defined by this specification.

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

1786

1787 6.7.1 Strict Layout Rules for WSS 1.0

- 1788 1. Tokens that are included in the message MUST be declared before use. For example:
- 1789 a. A local signing token MUST occur before the signature that uses it.
- 1790 b. A local token serving as the source token for a derived key token MUST occur before that
- 1791 derived key token.
- 1792 c. A local encryption token MUST occur before the reference list that points to
- 1793 xenc:EncryptedData elements that use it.
- 1794 d. If the same token is used for both signing and encryption, then it SHOULD appear before
- 1795 the ds:Signature and xenc:ReferenceList elements in the security header that are
- 1796 generated using the token.
- 1797 2. Signed elements inside the security header MUST occur before the signature that signs them.
- 1798 For example:
- 1799 a. A timestamp MUST occur before the signature that signs it.

- 1800 b. A Username token (usually in encrypted form) MUST occur before the signature that
1801 signs it.
- 1802 c. A primary signature MUST occur before the supporting token signature that signs the
1803 primary signature's signature value element.
- 1804 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1805 has the same order requirements as the source plain text element, unless requirement 4
1806 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1807 supporting token signature per 2.c above and an encrypted token has the same ordering
1808 requirements as the unencrypted token.
- 1809 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1810 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1811 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1812 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1813 Layout Rules for WSS 1.1
- 1814 1. Tokens that are included in the message MUST be declared before use. For example:
- 1815 a. A local signing token MUST occur before the signature that uses it.
- 1816 b. A local token serving as the source token for a derived key token MUST occur before that
1817 derived key token.
- 1818 c. A local encryption token MUST occur before the reference list that points to
1819 xenc:EncryptedData elements that use it.
- 1820 d. If the same token is used for both signing and encryption, then it SHOULD appear before
1821 the ds:Signature and xenc:ReferenceList elements in the security header that are
1822 generated using the token.
- 1823 2. Signed elements inside the security header MUST occur before the signature that signs them.
1824 For example:
- 1825 a. A timestamp MUST occur before the signature that signs it.
- 1826 b. A Username token (usually in encrypted form) MUST occur before the signature that
1827 signs it.
- 1828 c. A primary signature MUST occur before the supporting token signature that signs the
1829 primary signature's signature value element.
- 1830 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1831 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1832 has the same order requirements as the source plain text element, unless requirement 4
1833 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1834 supporting token signature per 2.c above and an encrypted token has the same ordering
1835 requirements as the unencrypted token.
- 1836 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1837 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1838 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1839 However, the xenc:ReferenceList is NOT REQUIRED to appear before independently encrypted
1840 tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1841 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1842 1.1] MUST obey rule 1 above.

1843 7 Security Binding Assertions

1844 The appropriate representation of the different facets of security mechanisms requires distilling the
1845 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy
1846 scope of assertions defined in this section is the policy scope of their containing element.

1847 7.1 AlgorithmSuite Assertion

1848 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]
1849 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

1850 Syntax

```
1851 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1852   <wsp:Policy xmlns:wsp="...">  
1853     (<sp:Basic256 ... /> |  
1854     <sp:Basic192 ... /> |  
1855     <sp:Basic128 ... /> |  
1856     <sp:TripleDes ... /> |  
1857     <sp:Basic256Rsa15 ... /> |  
1858     <sp:Basic192Rsa15 ... /> |  
1859     <sp:Basic128Rsa15 ... /> |  
1860     <sp:TripleDesRsa15 ... /> |  
1861     <sp:Basic256Sha256 ... /> |  
1862     <sp:Basic192Sha256 ... /> |  
1863     <sp:Basic128Sha256 ... /> |  
1864     <sp:TripleDesSha256 ... /> |  
1865     <sp:Basic256Sha256Rsa15 ... /> |  
1866     <sp:Basic192Sha256Rsa15 ... /> |  
1867     <sp:Basic128Sha256Rsa15 ... /> |  
1868     <sp:TripleDesSha256Rsa15 ... /> |  
1869     ...)  
1870     <sp:InclusiveC14N ... /> ?  
1871     <sp:InclusiveC14N11 ... /> ?  
1872     <sp:SOAPNormalization10 ... /> ?  
1873     <sp:STRTransform10 ... /> ?  
1874     (<sp:XPath10 ... /> |  
1875     <sp:XPathFilter20 ... /> |  
1876     <sp:AbsXPath ... /> |  
1877     ...)?  
1878     ...  
1879   </wsp:Policy>  
1880   ...  
1881 </sp:AlgorithmSuite>
```

1882
1883 The following describes the attributes and elements listed in the schema outlined above:

1884 /sp:AlgorithmSuite

1885 This identifies an AlgorithmSuite assertion.

1886 /sp:AlgorithmSuite/wsp:Policy

1887 This REQUIRED element contains one or more policy assertions that indicate the specific
1888 algorithm suite to use.

1889 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1890 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1891 set to 'Basic256'.

- 1892 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192
1893 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1894 set to 'Basic192'.
- 1895 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128
1896 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1897 set to 'Basic128'.
- 1898 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes
1899 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1900 set to 'TripleDes'.
- 1901 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15
1902 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1903 set to 'Basic256Rsa15'.
- 1904 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15
1905 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1906 set to 'Basic192Rsa15'.
- 1907 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15
1908 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1909 set to 'Basic128Rsa15'.
- 1910 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15
1911 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1912 set to 'TripleDesRsa15'.
- 1913 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256
1914 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1915 set to 'Basic256Sha256'.
- 1916 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256
1917 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1918 set to 'Basic192Sha256'.
- 1919 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256
1920 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1921 set to 'Basic128Sha256'.
- 1922 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256
1923 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1924 set to 'TripleDesSha256'.
- 1925 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15
1926 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1927 set to 'Basic256Sha256Rsa15'.
- 1928 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15
1929 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1930 set to 'Basic192Sha256Rsa15'.
- 1931 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15
1932 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1933 set to 'Basic128Sha256Rsa15'.
- 1934 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1935 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1936 set to 'TripleDesSha256Rsa15'.

1937 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1938 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an
1939 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]
1940 property is 'ExC14N'.

1941 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N11

1942
1943 This optional element is a policy assertion that indicates that the
1944 [C14N] property of an algorithm suite is set to 'C14N11'. Note: as
1945 indicated in Section 6.1 the default value of the [C14N] property is
1946 'ExC14N'.

1947 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1948 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set
1949 to 'SNT'.

1950 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1951 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is
1952 set to 'STRT10'.

1953 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1954 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1955 'XPath'.

1956 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1957 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1958 'XPath20'.

1959 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1960 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1961 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1962

1963 7.2 Layout Assertion

1964 This assertion indicates a requirement for a particular security header layout as defined under the
1965 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1966 containing assertion.

1967 Syntax

```
1968 <sp:Layout xmlns:sp="..." ... >  
1969   <wsp:Policy xmlns:wsp="..." >  
1970     <sp:Strict ... /> |  
1971     <sp:Lax ... /> |  
1972     <sp:LaxTsFirst ... /> |  
1973     <sp:LaxTsLast ... /> |  
1974     ...  
1975   </wsp:Policy>  
1976   ...  
1977 </sp:Layout>
```

1978

1979 The following describes the attributes and elements listed in the schema outlined above:

1980 /sp:Layout

- 1981 This identifies a Layout assertion.
- 1982 /sp:Layout/wsp:Policy
- 1983 This REQUIRED element contains one or more policy assertions that indicate the specific security
1984 header layout to use.
- 1985 /sp:Layout/wsp:Policy/sp:Strict
- 1986 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1987 property is set to 'Strict'.
- 1988 /sp:Layout/wsp:Policy/sp:Lax
- 1989 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1990 property is set to 'Lax'.
- 1991 /sp:Layout/wsp:Policy/sp:LaxTsFirst
- 1992 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1993 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1994 'true' by the presence of an sp:IncludeTimestamp assertion.
- 1995 /sp:Layout/wsp:Policy/sp:LaxTsLast
- 1996 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1997 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
1998 'true' by the presence of an sp:IncludeTimestamp assertion.

1999 7.3 TransportBinding Assertion

- 2000 The TransportBinding assertion is used in scenarios in which message protection and security correlation
2001 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
2002 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
2003 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
2004 MUST apply to [Endpoint Policy Subject].

2005 Syntax

```

2006 <sp:TransportBinding xmlns:sp="..." ... >
2007   <wsp:Policy xmlns:wsp="...">
2008     <sp:TransportToken ... >
2009       <wsp:Policy> ... </wsp:Policy>
2010       ...
2011     </sp:TransportToken>
2012     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2013     <sp:Layout ... > ... </sp:Layout> ?
2014     <sp:IncludeTimestamp ... /> ?
2015     ...
2016   </wsp:Policy>
2017   ...
2018 </sp:TransportBinding>

```

- 2019
- 2020 The following describes the attributes and elements listed in the schema outlined above:

- 2021 /sp:TransportBinding
- 2022 This identifies a TransportBinding assertion.
- 2023 /sp:TransportBinding/wsp:Policy
- 2024 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
2025 assertion.
- 2026 /sp:TransportBinding/wsp:Policy/sp:TransportToken

2027 This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.
2028 The specified token populates the [Transport Token] property and indicates how the transport is
2029 secured.

2030 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

2031 This indicates a nested policy that identifies the type of Transport Token to use.

2032 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

2033 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2034 Suite] property. See Section 6.1 for more details.

2035 /sp:TransportBinding/wsp:Policy/sp:Layout

2036 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2037 Header Layout] property. See Section 6.7 for more details.

2038 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

2039 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2040 to 'true'.

2041 7.4 SymmetricBinding Assertion

2042 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
2043 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
2044 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
2045 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
2046 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
2047 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
2048 properties and is used as the basis for both encryption and signature in both directions. This assertion
2049 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

2050 Syntax

```
2051 <sp:SymmetricBinding xmlns:sp="..." ... >  
2052   <wsp:Policy xmlns:wsp="...">  
2053     (  
2054       <sp:EncryptionToken ... >  
2055         <wsp:Policy> ... </wsp:Policy>  
2056       </sp:EncryptionToken>  
2057       <sp:SignatureToken ... >  
2058         <wsp:Policy> ... </wsp:Policy>  
2059       </sp:SignatureToken>  
2060     ) | (  
2061       <sp:ProtectionToken ... >  
2062         <wsp:Policy> ... </wsp:Policy>  
2063       </sp:ProtectionToken>  
2064     )  
2065     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
2066     <sp:Layout ... > ... </sp:Layout> ?  
2067     <sp:IncludeTimestamp ... /> ?  
2068     <sp:EncryptBeforeSigning ... /> ?  
2069     <sp:EncryptSignature ... /> ?  
2070     <sp:ProtectTokens ... /> ?  
2071     <sp:OnlySignEntireHeadersAndBody ... /> ?  
2072     ...  
2073   </wsp:Policy>  
2074   ...  
2075 </sp:SymmetricBinding>
```

2076

2077 The following describes the attributes and elements listed in the schema outlined above:

2078 /sp:SymmetricBinding
2079 This identifies a SymmetricBinding assertion.

2080 /sp:SymmetricBinding/wsp:Policy
2081 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
2082 assertion.

2083 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
2084 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption
2085 Token. The specified token populates the [Encryption Token] property and is used for encryption.
2086 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

2087 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
2088 The policy contained here MUST identify exactly one token to use for encryption.

2089 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
2090 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.
2091 The specified token populates the [Signature Token] property and is used for the message
2092 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
2093 specified.

2094 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
2095 The policy contained here MUST identify exactly one token to use for signatures.

2096 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
2097 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.
2098 The specified token populates the [Encryption Token] and [Signature Token properties] and is
2099 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
2100 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
2101 specified.

2102 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
2103 The policy contained here MUST identify exactly one token to use for protection.

2104 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
2105 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2106 Suite] property. See Section 6.1 for more details.

2107 /sp:SymmetricBinding/wsp:Policy/sp:Layout
2108 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2109 Header Layout] property. See Section 6.7 for more details.

2110 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
2111 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2112 to 'true'.

2113 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
2114 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
2115 set to 'EncryptBeforeSigning'.

2116 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
2117 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
2118 property is set to 'true'.

2119 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
2120 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
2121 set to 'true'.

2122 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
2123 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
2124 Signatures] property is set to 'true'.

2125 7.5 AsymmetricBinding Assertion

2126 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means
2127 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly
2128 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and
2129 signature. However it is also common practice to use distinct keys for encryption and signature, because
2130 of their different lifecycles.

2131
2132 This binding enables either of these practices by means of four binding specific token properties: [Initiator
2133 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption
2134 Token].

2135
2136 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator
2137 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient
2138 Encryption Token] will both refer to the same token.

2139
2140 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator
2141 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient
2142 Encryption Token] will refer to different tokens.

2143
2144 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the
2145 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response
2146 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the
2147 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for
2148 the message encryption from initiator to the recipient. Note that in each case, the token is associated with
2149 the party (initiator or recipient) who knows the secret.

2150 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy
2151 Subject].

2152 Syntax

```
2153 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2154   <wsp:Policy xmlns:wsp="...">  
2155     (  
2156       <sp:InitiatorToken>  
2157         <wsp:Policy> ... </wsp:Policy>  
2158       </sp:InitiatorToken>  
2159     ) | (  
2160       <sp:InitiatorSignatureToken>  
2161         <wsp:Policy> ... </wsp:Policy>  
2162       </sp:InitiatorSignatureToken>  
2163       <sp:InitiatorEncryptionToken>  
2164         <wsp:Policy> ... </wsp:Policy>  
2165       </sp:InitiatorEncryptionToken>  
2166     )  
2167     (  
2168       <sp:RecipientToken>  
2169         <wsp:Policy> ... </wsp:Policy>  
2170       </sp:RecipientToken>  
2171     ) | (  
2172
```

```

2172     <sp:RecipientSignatureToken>
2173         <wsp:Policy> ... </wsp:Policy>
2174     </sp:RecipientSignatureToken>
2175     <sp:RecipientEncryptionToken>
2176         <wsp:Policy> ... </wsp:Policy>
2177     </sp:RecipientEncryptionToken>
2178 )
2179 <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2180 <sp:Layout ... > ... </sp:Layout> ?
2181 <sp:IncludeTimestamp ... /> ?
2182 <sp:EncryptBeforeSigning ... /> ?
2183 <sp:EncryptSignature ... /> ?
2184 <sp:ProtectTokens ... /> ?
2185 <sp:OnlySignEntireHeadersAndBody ... /> ?
2186 ...
2187 </wsp:Policy>
2188 ...
2189 </sp:AsymmetricBinding>

```

2190
2191 The following describes the attributes and elements listed in the schema outlined above:

2192 /sp:AsymmetricBinding

2193 This identifies a AsymmetricBinding assertion.

2194 /sp:AsymmetricBinding/wsp:Policy

2195 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
2196 assertion.

2197 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2198 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.
2199 The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
2200 properties and is used for the message signature from initiator to recipient, and encryption from
2201 recipient to initiator.

2202 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

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

2204 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2205 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2206 Signature Token. The specified token populates the [Initiator Signature Token] property and is
2207 used for the message signature from initiator to recipient.

2208 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2209 The policy contained here MUST identify one or more token assertions.

2210 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2211 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2212 Encryption Token. The specified token populates the [Initiator Encryption Token] property and is
2213 used for the message encryption from recipient to initiator.

2214 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2215 The policy contained here MUST identify one or more token assertions.

2216 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2217 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.
2218 The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
2219 property and is used for encryption from initiator to recipient, and for the message signature from
2220 recipient to initiator.

- 2221 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy
2222 The policy contained here MUST identify one or more token assertions.
- 2223 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken
2224 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
2225 Signature Token. The specified token populates the [Recipient Signature Token] property and is
2226 used for the message signature from recipient to initiator.
- 2227 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
2228 The policy contained here MUST identify one or more token assertions.
- 2229 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
2230 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
2231 Encryption Token. The specified token populates the [Recipient Encryption Token] property and
2232 is used for the message encryption from initiator to recipient.
- 2233 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
2234 The policy contained here MUST identify one or more token assertions.
- 2235 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
2236 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2237 Suite] property. See Section 6.1 for more details.
- 2238 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
2239 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2240 Header Layout] property. See Section 6.7 for more details.
- 2241 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
2242 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2243 to 'true'.
- 2244 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
2245 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
2246 set to 'EncryptBeforeSigning'.
- 2247 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
2248 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
2249 property is set to 'true'.
- 2250 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
2251 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
2252 set to 'true'.
- 2253 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
2254 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
2255 Signatures] property is set to 'true'.

2256

8 Supporting Tokens

2257 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to
2258 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore
2259 be referred to as the “message signature”. In case of Transport Binding the message is signed outside of
2260 the message XML by the underlying transport protocol and the signature itself is not part of the message.
2261 Additional tokens MAY be specified to augment the claims provided by the token associated with the
2262 “message signature” provided by the Security Binding. This section defines seven properties related to
2263 supporting token requirements which MAY be referenced by a Security Binding: [Supporting Tokens],
2264 [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens],
2265 [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing
2266 Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties:
2267 SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,
2268 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,
2269 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These
2270 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy
2271 Subject] or [Operation Policy Subject].

2272

2273 Supporting tokens MAY be specified at a different scope than the binding assertion which provides
2274 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while
2275 the supporting tokens might be defined at the scope of a message. When assertions that populate this
2276 property are defined in overlapping scopes, the sender SHOULD merge the requirements by including all
2277 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

2278

2279 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the
2280 tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements
2281 (tokens, signatures, reference lists etc.) in the security header would be used to determine which order
2282 signature and encryptions occurred in.

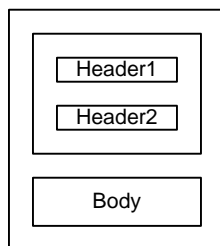
2283

2284 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional
2285 constraints defined by the supporting token assertion. For example, if the supporting token assertion
2286 specifies message parts that need to be encrypted, the specified tokens need to be capable of
2287 encryption.

2288

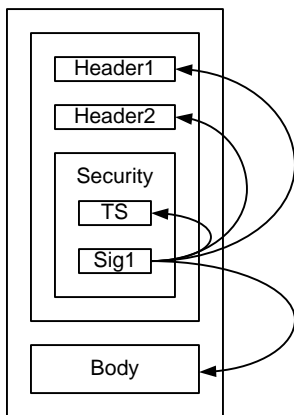
2289 To illustrate the different ways that supporting tokens MAY be bound to the message, let’s consider a
2290 message with three components: Header1, Header2, and Body.

2291

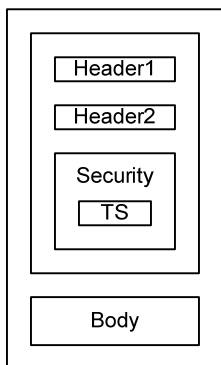


2292

2293 Even before any supporting tokens are added, each binding requires that the message is signed using a
 2294 token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important
 2295 parts of the message including the message timestamp (TS) facilitate replay detection. The signature is
 2296 then included as part of the Security header as illustrated below:
 2297



2298
 2299 Note: if REQUIRED, the initiator may also include in the Security header the token used as the basis for
 2300 the message signature (Sig1), not shown in the diagram.
 2301 If transport security is used, only the message timestamp (TS) is included in the Security header as
 2302 illustrated below. The “message signature” is provided by the underlying transport protocol and is not part
 2303 of the message XML.



2304

2305 8.1 SupportingTokens Assertion

2306 Supporting tokens are included in the security header and MAY OPTIONALLY include additional
 2307 message parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and
 2308 do not require any protection (signature or encryption) to be applied to the message before they are
 2309 added. More specifically there is no requirement on “message signature” being present before the
 2310 supporting tokens are added. However it is RECOMMENDED to employ underlying protection
 2311 mechanism to ensure that the supporting tokens are cryptographically bound to the message during the
 2312 transmission.

2313 Syntax

```
2314 <sp:SupportingTokens xmlns:sp="..." ... >
2315   <wsp:Policy xmlns:wsp="...">
2316     [Token Assertion]+
2317     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2318     (
2319       <sp:SignedParts ... > ... </sp:SignedParts> |
```


2320
2321
2322
2323
2324
2325
2326
2327

```
<sp:SignedElements ... > ... </sp:SignedElements> |  
<sp:EncryptedParts ... > ... </sp:EncryptedParts> |  
<sp:EncryptedElements ... > ... </sp:EncryptedElements> |  
  ) *  
  ...  
</wsp:Policy>  
  ...  
</sp:SupportingTokens>
```

2328

2329 The following describes the attributes and elements listed in the schema outlined above:

2330

/sp:SupportingTokens

2331
2332

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

2333

/sp:SupportingTokens/wsp:Policy

2334

This describes additional requirements for satisfying the SupportingTokens assertion.

2335

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

2336

The policy MUST identify one or more token assertions.

2337

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

2338
2339
2340

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.

2341

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

2342
2343
2344

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.

2345

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

2346
2347
2348

This OPTIONAL element is a policy assertion that 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.

2349

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

2350
2351
2352

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.

2353

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

2354
2355
2356

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.

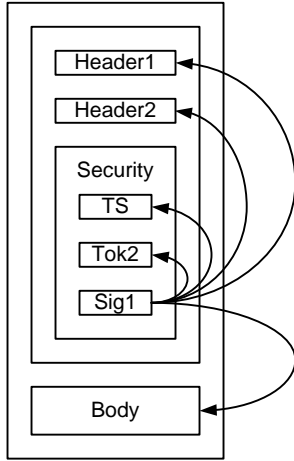
2357

8.2 SignedSupportingTokens Assertion

2358

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

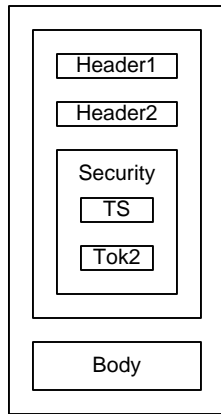
2360
2361



2362

2363 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2364



2365

2366 **Syntax**

```

2367 <sp:SignedSupportingTokens xmlns:sp="..." ... >
2368   <wsp:Policy xmlns:wsp="..." >
2369     [Token Assertion]+
2370     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2371     (
2372       <sp:SignedParts ... > ... </sp:SignedParts> |
2373       <sp:SignedElements ... > ... </sp:SignedElements> |
2374       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2375       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2376     ) *
2377     ...
2378   </wsp:Policy>
2379   ...
2380 </sp:SignedSupportingTokens>

```

2381

2382 The following describes the attributes and elements listed in the schema outlined above:

2383 /sp:SignedSupportingTokens

2384 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed
2385 Supporting Tokens] property.

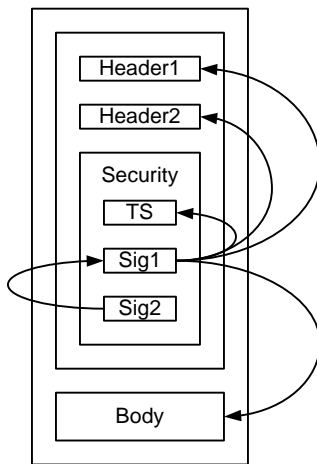
2386 /sp:SignedSupportingTokens/wsp:Policy

2387 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

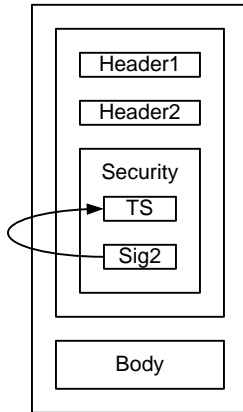
- 2388 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 2389 The policy MUST identify one or more token assertions.
- 2390 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 2391 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
 2392 describes the algorithms to use for cryptographic operations performed with the tokens identified
 2393 by this policy assertion.
- 2394 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 2395 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
 2396 and describes additional message parts that MUST be included in the signature generated with
 2397 the token identified by this policy assertion.
- 2398 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 2399 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
 2400 and describes additional message elements that MUST be included in the signature generated
 2401 with the token identified by this policy assertion.
- 2402 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 2403 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
 2404 and describes additional message parts that MUST be encrypted using the token identified by
 2405 this policy assertion.
- 2406 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 2407 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
 2408 and describes additional message elements that MUST be encrypted using the token identified
 2409 by this policy assertion.

2410 8.3 EndorsingSupportingTokens Assertion

- 2411 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 2412 produced from the message signature and MAY OPTIONALLY include additional message parts to sign
 2413 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message
 2414 signature (Sig1):
 2415



- 2416
 2417 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 2418 below:
 2419



2420

2421 **Syntax**

```

2422 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2423   <wsp:Policy xmlns:wsp="...">
2424     [Token Assertion]+
2425     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2426     (
2427       <sp:SignedParts ... > ... </sp:SignedParts> |
2428       <sp:SignedElements ... > ... </sp:SignedElements> |
2429       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2430       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2431     ) *
2432     ...
2433   </wsp:Policy>
2434   ...
2435 </sp:EndorsingSupportingTokens>

```

2436

2437 The following describes the attributes and elements listed in the schema outlined above:

2438 /sp:EndorsingSupportingTokens

2439 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
2440 [Endorsing Supporting Tokens] property.

2441 /sp:EndorsingSupportingTokens/wsp:Policy

2442 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2443 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2444 The policy MUST identify one or more token assertions.

2445 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2446 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2447 describes the algorithms to use for cryptographic operations performed with the tokens identified
2448 by this policy assertion.

2449 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2450 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2451 and describes additional message parts that MUST be included in the signature generated with
2452 the token identified by this policy assertion.

2453 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

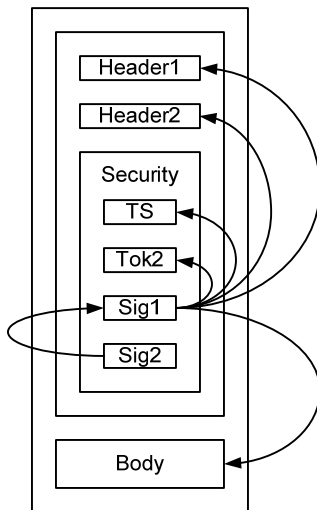
2454 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2455 and describes additional message elements that MUST be included in the signature generated
2456 with the token identified by this policy assertion.

2457 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
2458 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2459 and describes additional message parts that MUST be encrypted using the token identified by
2460 this policy assertion.

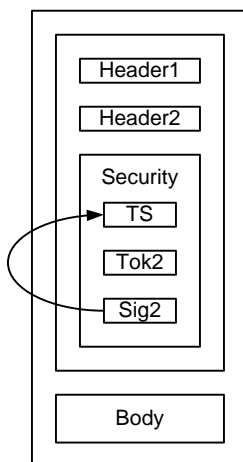
2461 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
2462 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2463 and describes additional message elements that MUST be encrypted using the token identified
2464 by this policy assertion.

2465 8.4 SignedEndorsingSupportingTokens Assertion

2466 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
2467 and are themselves signed by that message signature, that is both tokens (the token used for the
2468 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY
2469 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
2470 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
2471 message signature (Sig1):
2472



2473
2474 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
2475 SHOULD cover the message timestamp as illustrated below:
2476



2478 **Syntax**

```
2479 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2480 <wsp:Policy xmlns:wsp="...">
2481 [Token Assertion]+
2482 <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2483 (
2484 <sp:SignedParts ... > ... </sp:SignedParts> |
2485 <sp:SignedElements ... > ... </sp:SignedElements> |
2486 <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2487 <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2488 ) *
2489 ...
2490 </wsp:Policy>
2491 ...
2492 </sp:SignedEndorsingSupportingTokens>
```

2493

2494 The following describes the attributes and elements listed in the schema outlined above:

2495 /sp:SignedEndorsingSupportingTokens

2496 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the
2497 [Signed Endorsing Supporting Tokens] property.

2498 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2499 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2500 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2501 The policy MUST identify one or more token assertions.

2502 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2503 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2504 describes the algorithms to use for cryptographic operations performed with the tokens identified
2505 by this policy assertion.

2506 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2507 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2508 and describes additional message parts that MUST be included in the signature generated with
2509 the token identified by this policy assertion.

2510 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2511 This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional
2512 message elements that MUST be included in the signature generated with the token identified by
2513 this policy assertion.

2514 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2515 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2516 and describes additional message parts that MUST be encrypted using the token identified by
2517 this policy assertion.

2518 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2519 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2520 and describes additional message elements that MUST be encrypted using the token identified
2521 by this policy assertion.

2522 **8.5 SignedEncryptedSupportingTokens Assertion**

2523 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also
2524 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2525 encrypting the supporting tokens.

2526 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of
2527 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2528 sp:SignedSupportingTokens assertion.

2529 **8.6 EncryptedSupportingTokens Assertion**

2530 Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in
2531 the security header and MUST be encrypted when they appear in the security header.
2532 Element encryption SHOULD be used for encrypting these tokens. The encrypted supporting
2533 tokens can be added to any SOAP message and do not require the “message signature”
2534 being present before the encrypted supporting tokens are added.

2535 The syntax for the sp:EncryptedSupportingTokens differs from the syntax of
2536 sp:SupportingTokens only in the name of the assertion itself. All nested policy is as per the
2537 sp:SupportingTokens assertion.

2538 The encrypted supporting tokens SHOULD be used only when the sender cannot provide the
2539 “message signature” and it is RECOMMENDED that the receiver employs some security
2540 mechanisms external to the message to prevent the spoofing attacks. In all other cases it is
2541 RECOMMENDED to use signed encrypted supporting tokens instead to ensure that the
2542 encrypted tokens are cryptographically bound to the message (See section 8.5).

2543 **8.7 EndorsingEncryptedSupportingTokens Assertion**

2544 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also
2545 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for
2546 encrypting the supporting tokens.

2547 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of
2548 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the
2549 sp:EndorsingSupportingTokens assertion.

2550 **8.8 SignedEndorsingEncryptedSupportingTokens Assertion**

2551 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section
2552 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD
2553 be used for encrypting the supporting tokens.

2554 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of
2555 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per
2556 the sp:SignedEndorsingSupportingTokens assertion.

2557 **8.9 Interaction between [Token Protection] property and supporting 2558 token assertions**

2559 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that
2560 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2561 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or
2562 the [Signature Token] in the Symmetric binding case, covers that token.
- 2563 • Endorsing signatures cover the main signature and the endorsing token.

- 2564 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the
2565 message signature and once by the endorsing signature.

2566 In addition, signed supporting tokens are covered by the message signature, although this is independent
2567 of [Token Protection].

2568 8.10 Example

2569 Example policy containing supporting token assertions:

```
2570 <!-- Example Endpoint Policy -->
2571 <wsp:Policy xmlns:wsp="...">
2572   <sp:SymmetricBinding xmlns:sp="...">
2573     <wsp:Policy>
2574       <sp:ProtectionToken>
2575         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2576           <sp:Issuer>...</sp:Issuer>
2577           <sp:RequestSecurityTokenTemplate>
2578             ...
2579           </sp:RequestSecurityTokenTemplate>
2580         </sp:IssuedToken>
2581       </sp:ProtectionToken>
2582       <sp:AlgorithmSuite>
2583         <wsp:Policy>
2584           <sp:Basic256 />
2585         </wsp:Policy>
2586       </sp:AlgorithmSuite>
2587       ...
2588     </wsp:Policy>
2589   </sp:SymmetricBinding>
2590   ...
2591   <sp:SignedSupportingTokens>
2592     <wsp:Policy>
2593       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2594     </wsp:Policy>
2595   </sp:SignedSupportingTokens>
2596   <sp:SignedEndorsingSupportingTokens>
2597     <wsp:Policy>
2598       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2599         <wsp:Policy>
2600           <sp:WssX509v3Token10 />
2601         </wsp:Policy>
2602       </sp:X509Token>
2603     </wsp:Policy>
2604   </sp:SignedEndorsingSupportingTokens>
2605   ...
2606 </wsp:Policy>
```

2607 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be
2608 included in the security header and covered by the message signature. The
2609 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the
2610 security header and covered by the message signature. In addition, a signature over the message
2611 signature based on the key material associated with the X509 certificate must be included in the security
2612 header.

2613

9 WSS: SOAP Message Security Options

2614 There are several OPTIONAL aspects to the WSS: SOAP Message Security specification that are
2615 independent of the trust and token taxonomies. This section describes another class of properties and
2616 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The
2617 assertions defined here MUST apply to [Endpoint Policy Subject].

2618 The properties and assertions dealing with token references defined in this section indicate whether the
2619 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and
2620 recipient MAY send a fault if such references are encountered.

2621

2622 Note: This approach is chosen because:

2623 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used
2624 in a single reference.

2625 B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending
2626 on which of a series of messages is being secured.

2627

2628 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a
2629 `wsse:InvalidSecurity` fault.

2630

2631 **WSS: SOAP Message Security 1.0 Properties**

2632 **[Direct References]**

2633 This property indicates whether the initiator and recipient MUST be able to process direct token
2634 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations
2635 MUST be able to process such references.

2636

2637 **[Key Identifier References]**

2638 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific
2639 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to
2640 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST
2641 NOT generate such references and that the initiator and recipient MAY send a fault if such references are
2642 encountered. This property has a default value of 'false'.

2643

2644 **[Issuer Serial References]**

2645 This boolean property indicates whether the initiator and recipient MUST be able to process references
2646 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST
2647 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT
2648 generate such references and that the initiator and recipient MAY send a fault if such references are
2649 encountered. This property has a default value of 'false'.

2650

2651 **[External URI References]**

2652 This boolean property indicates whether the initiator and recipient MUST be able to process references to
2653 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST
2654 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2655 generate such references and that the initiator and recipient MAY send a fault if such references are
2656 encountered. This property has a default value of 'false'.

2657 **[Embedded Token References]**

2658 This boolean property indicates whether the initiator and recipient MUST be able to process references
2659 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to
2660 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2661 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2662 This property has a default value of 'false'.

2663

2664 **WSS: SOAP Message Security 1.1 Properties**

2665 **[Thumbprint References]**

2666 This boolean property indicates whether the initiator and recipient MUST be able to process references
2667 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to
2668 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2669 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2670 This property has a default value of 'false'.

2671

2672 **[EncryptedKey References]**

2673 This boolean property indicates whether the initiator and recipient MUST be able to process references
2674 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to
2675 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2676 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2677 This property has a default value of 'false'.

2678

2679 **[Signature Confirmation]**

2680 This boolean property specifies whether `wss11:SignatureConfirmation` elements SHOULD be
2681 used as defined in WSS: Soap Message Security 1.1. If the value is 'true',
2682 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If
2683 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property
2684 applies to all signatures that are included in the security header. This property has a default value of
2685 'false'. This value of this property does not affect the message parts protected by the message signature
2686 (see the `sp:SignedParts` and `sp:SignedElements` assertions)

2687 **9.1 Wss10 Assertion**

2688 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are
2689 supported.

2690 **Syntax**

```
2691 <sp:Wss10 xmlns:sp="..." ... >  
2692   <wsp:Policy xmlns:wsp="...">  
2693     <sp:MustSupportRefKeyIdentifier ... /> ?  
2694     <sp:MustSupportRefIssuerSerial ... /> ?  
2695     <sp:MustSupportRefExternalURI ... /> ?  
2696     <sp:MustSupportRefEmbeddedToken ... /> ?  
2697     ...  
2698   </wsp:Policy>  
2699   ...  
2700 </sp:Wss10>
```

2701

2702 The following describes the attributes and elements listed in the schema outlined above:

2703 /sp:Wss10

2704 This identifies a WSS10 assertion.

2705 /sp:Wss10/wsp:Policy

2706 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2707 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier

2708 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]

2709 property is set to 'true'.

2710 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial

2711 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]

2712 property is set to 'true'.

2713 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI

2714 This OPTIONAL element is a policy assertion indicates that the [External URI References]

2715 property is set to 'true'.

2716 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken

2717 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]

2718 property is set to 'true'.

2719 9.2 Wss11 Assertion

2720 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are

2721 supported.

2722 Syntax

```

2723 <sp:Wss11 xmlns:sp="..." ... >
2724   <wsp:Policy xmlns:wsp="...">
2725     <sp:MustSupportRefKeyIdentifier ... /> ?
2726     <sp:MustSupportRefIssuerSerial ... /> ?
2727     <sp:MustSupportRefExternalURI ... /> ?
2728     <sp:MustSupportRefEmbeddedToken ... /> ?
2729     <sp:MustSupportRefThumbprint ... /> ?
2730     <sp:MustSupportRefEncryptedKey ... /> ?
2731     <sp:RequireSignatureConfirmation ... /> ?
2732     ...
2733   </wsp:Policy>
2734 </sp:Wss11>

```

2735

2736 The following describes the attributes and elements listed in the schema outlined above:

2737 /sp:Wss11

2738 This identifies an WSS11 assertion.

2739 /sp:Wss11/wsp:Policy

2740 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2741 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier

2742 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]

2743 property is set to 'true'.

2744 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial

2745 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]

2746 property is set to 'true'.

2747 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2748 This OPTIONAL element is a policy assertion indicates that the [External URI References]
2749 property is set to 'true'.

2750 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2751 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2752 property is set to 'true'.

2753 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2754 This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property
2755 is set to 'true'.

2756 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2757 This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]
2758 property is set to 'true'.

2759 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2760 This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property
2761 is set to 'true'.

2762

10 WS-Trust Options

2763 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically
2764 with client and server challenges and entropy behaviors. These assertions relate to interactions with a
2765 Security Token Service and MAY augment the behaviors defined by the Binding Property Assertions
2766 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2767

2768 **WS-Trust Properties**

2769 **[Client Challenge]**

2770 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a
2771 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of
2772 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of
2773 messages exchanged by the client and service in satisfying the RST. This property has a default value of
2774 'false'.

2775

2776 **[Server Challenge]**

2777 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a
2778 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of
2779 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server MAY
2780 increase the number of messages exchanged by the client and service in order to accommodate the
2781 `wst:SignChallengeResponse` element sent by the client to the server in response to the
2782 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the
2783 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2784

2785 **[Client Entropy]**

2786 This boolean property indicates whether client entropy is REQUIRED to be used as key material for a
2787 requested proof token. A value of 'true' indicates that client entropy is REQUIRED. A value of 'false'
2788 indicates that client entropy is NOT REQUIRED. This property has a default value of 'false'.

2789

2790 **[Server Entropy]**

2791 This boolean property indicates whether server entropy is REQUIRED to be used as key material for a
2792 requested proof token. A value of 'true' indicates that server entropy is REQUIRED. A value of 'false'
2793 indicates that server entropy is NOT REQUIRED. This property has a default value of 'false'.

2794 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy
2795 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm
2796 Suite] property.

2797

2798 **[Issued Tokens]**

2799 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in
2800 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'
2801 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of
2802 'false'.

2803 **[Collection]**

2804 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A
2805 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and
2806 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that
2807 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default
2808 value of 'false'.

2809

2810 **[Scope Policy 1.5]**

2811 This boolean property indicates whether the wsp:AppliesTo element in the [WS-Policy] 1.5 namespace is
2812 supported as described in [WS-Trust]. A value of 'true' indicates that the wsp:AppliesTo element in the
2813 [WS-Policy] 1.5 namespace is supported. A value of 'false' indicates that the wsp:AppliesTo element in
2814 the [WS-Policy] 1.5 namespace is not supported, the [WS-Policy] 1.2 namespace is used instead in this
2815 case. This property has a default value of 'false'.

2816

2817 **[Interactive Challenge]**

2818 This boolean property indicates whether interactive challenges are supported. A value of 'true' indicates
2819 that a wst14:InteractiveChallenge element is supported inside of an RSTR sent by the server to the client.
2820 A value of 'false' indicates that wst14:InteractiveChallenge is not supported. A challenge issued by the
2821 server may increase the number of messages exchanged by the client and service in order to
2822 accommodate the wst14:InteractiveChallengeResponse element sent by the client to the server in
2823 response to the wst14:InteractiveChallenge element. There is an optimization in which a client MAY send
2824 the wst14:InteractiveChallengeResponse element in an initial RST to the server. A final RSTR containing
2825 the issued token will follow subsequent to the server receiving the wst14:InteractiveChallengeResponse
2826 element. This property has a default value of 'false'.

2827

2828 **10.1 Trust13 Assertion**

2829 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.

2830 **Syntax**

```
2831 <sp:Trust13 xmlns:sp="..." ... >  
2832   <wsp:Policy xmlns:wsp="...">  
2833     <sp:MustSupportClientChallenge ... />?  
2834     <sp:MustSupportServerChallenge ... />?  
2835     <sp:RequireClientEntropy ... />?  
2836     <sp:RequireServerEntropy ... />?  
2837     <sp:MustSupportIssuedTokens ... />?  
2838     <sp:RequireRequestSecurityTokenCollection />?  
2839     <sp:RequireAppliesTo />?  
2840     <sp13:ScopePolicy15 />?  
2841     <sp13:MustSupportInteractiveChallenge />?  
2842     ...  
2843   </wsp:Policy>  
2844   ...  
2845 </sp:Trust13 ... >
```

2846

2847 The following describes the attributes and elements listed in the schema outlined above:

2848 /sp:Trust13

2849 This identifies a Trust13 assertion.

2850 /sp:Trust13/wsp:Policy

2851 This indicates a policy that controls WS-Trust 1.3 options.

2852 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge
2853 This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set
2854 to 'true'.

2855 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge
2856 This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set
2857 to 'true'.

2858 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy
2859 This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to
2860 'true'.

2861 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy
2862 This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to
2863 'true'.

2864 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens
2865 This OPTIONAL element is a policy assertion indicates that the [Issued Tokens] property is set to
2866 'true'.

2867 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection
2868 This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to
2869 'true'.

2870 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo
2871 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor
2872 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2873 /sp:Trust13/wsp:Policy/sp13:ScopePolicy15
2874 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5]
2875 property is set to 'true'.

2876 /sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge
2877 This optional element is a policy assertion indicates that the [Interactive Challenge]
2878 property is set to 'true'.

2879 11 Guidance on creating new assertions and assertion 2880 extensibility

2881 This non-normative appendix provides guidance for designers of new assertions intended for use with this
2882 specification.

2883 11.1 General Design Points

- 2884 • Prefer Distinct Qnames
- 2885 • Parameterize using nested policy where possible.
- 2886 • Parameterize using attributes and/or child elements where necessary.

2887 11.2 Detailed Design Guidance

2888 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per
2889 WS-Policy is performed by matching element Qnames. Matching does not take into account attributes
2890 that are present on the assertion element. Nor does it take into account child elements except for
2891 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions
2892 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2893
2894 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken
2895 into account. In general, multiple assertions with distinct Qnames are preferably to a single assertion that
2896 uses attributes and/or content to distinguish different cases. For example, given two possible assertion
2897 designs;

```
2898  
2899 Design 1  
2900  
2901 <A1/>  
2902 <A2/>  
2903 <A3/>  
2904  
2905 Design 2.  
2906  
2907 <A Parameter='1' />  
2908 <A Parameter='2' />  
2909 <A Parameter='3' />  
2910
```

2911 then design 1. Would generally be preferred because it allows the policy matching logic to provide more
2912 accurate matches between policies.

2913
2914 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct
2915 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These
2916 distinct token assertions make policy matching much more useful as less false positives are generated
2917 when performing policy matching.

2918
2919 There are cases where using attributes or child elements as parameters in assertion design is
2920 reasonable. Examples include cases when implementations are expected to understand all the values for
2921 a given parameter and when encoding the parameter information into the assertion QName would result
2922 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2923 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2924 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2925 was encoded into the assertion Qnames, each existing token assertion would require five variants, one
2926 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2927

2928 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2929 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2930 assertion is parameterized by the nested token version assertions. Policy matching can use these
2931 parameters to find matches between policies where the broad token type is support by both parties but
2932 they might not support the same specific versions.

2933

2934 Note, when designing assertions for new token types such assertions SHOULD allow the
2935 sp:IncludeToken attribute and SHOULD allow nested policy.

2936

2937

12 Security Considerations

2938 It is strongly recommended that policies and assertions be signed to prevent tampering.

2939 It is recommended that policies should not be accepted unless they are signed and have an associated
2940 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely
2941 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that
2942 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2943

2944 It should be noted that the mechanisms described in this document could be secured as part of a SOAP
2945 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using
2946 object-specific security mechanisms.

2947

2948 It is recommended that policies not specify two (or more) SignedSupportingTokens or
2949 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are
2950 subject to modification which may be undetectable.

2951

2952 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest
2953 of the policy in order to combat certain XML substitution attacks.

2954

2955

13 Conformance

2956 An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level
2957 requirements defined within this specification. A SOAP Node MUST NOT use the XML namespace
2958 identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is compliant with this
2959 specification.

2960

2961 This specification references a number of other specifications (see the table above). In order to comply
2962 with this specification, an implementation MUST implement the portions of referenced specifications
2963 necessary to comply with the required provisions of this specification. Additionally, the implementation of
2964 the portions of the referenced specifications that are specifically cited in this specification MUST comply
2965 with the rules for those portions as established in the referenced specification.

2966 Additionally normative text within this specification takes precedence over normative outlines (as
2967 described in section 1.4.1), which in turn take precedence over the XML Schema [XML Schema Part 1,
2968 Part 2] and WSDL [WSDL 1.1] descriptions. That is, the normative text in this specification further
2969 constrains the schemas and/or WSDL that are part of this specification; and this specification contains
2970 further constraints on the elements defined in referenced schemas.

2971 This specification defines a number of extensions; compliant services are NOT REQUIRED to implement
2972 OPTIONAL features defined in this specification. However, if a service implements an aspect of the
2973 specification, it MUST comply with the requirements specified (e.g. related "MUST" statements). If an
2974 OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for any
2975 other unrecognized/unsupported message. If an OPTIONAL message is supported, then the
2976 implementation MUST satisfy all of the MUST and REQUIRED sections of the message.

2977

2978 **A. Assertions and WS-PolicyAttachment**

2979 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per
2980 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical
2981 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any
2982 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy
2983 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

2984 **A.1 Endpoint Policy Subject Assertions**

2985 **A.1.1 Security Binding Assertions**

2986 [TransportBinding Assertion](#) (Section 7.3)
2987 [SymmetricBinding Assertion](#) (Section 7.4)
2988 [AsymmetricBinding Assertion](#) (Section 7.5)

2989 **A.1.2 Token Assertions**

2990 [SupportingTokens Assertion](#) (Section 8.1)
2991 [SignedSupportingTokens Assertion](#) (Section 8.2)
2992 [EndorsingSupportingTokens Assertion](#) (Section 8.3)
2993 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)
2994 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)
2995 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)
2996 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

2997 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

2998 [Wss10 Assertion](#) (Section 9.1)

2999 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

3000 [Wss11 Assertion](#) (Section 9.2)

3001 **A.1.5 Trust 1.0 Assertions**

3002 [Trust13 Assertion](#) (Section 10.1)

3003 **A.2 Operation Policy Subject Assertions**

3004 **A.2.1 Security Binding Assertions**

3005 [SymmetricBinding Assertion](#) (Section 7.4)
3006 [AsymmetricBinding Assertion](#) (Section 7.5)

3007 **A.2.2 Supporting Token Assertions**

3008 [SupportingTokens Assertion](#) (Section 8.1)
3009 [SignedSupportingTokens Assertion](#) (Section 8.2)

3010	EndorsingSupportingTokens Assertion	(Section 8.3)
3011	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3012	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3013	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3014	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3015 **A.3 Message Policy Subject Assertions**

3016 **A.3.1 Supporting Token Assertions**

3017	SupportingTokens Assertion	(Section 8.1)
3018	SignedSupportingTokens Assertion	(Section 8.2)
3019	EndorsingSupportingTokens Assertion	(Section 8.3)
3020	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3021	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3022	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3023	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3024 **A.3.2 Protection Assertions**

3025	SignedParts Assertion	(Section 4.1.1)
3026	SignedElements Assertion	(Section 4.1.2)
3027	EncryptedParts Assertion	(Section 4.2.1)
3028	EncryptedElements Assertion	(Section 4.2.2)
3029	ContentEncryptedElements Assertion	(Section 4.2.3)
3030	RequiredElements Assertion	(Section 4.3.1)
3031	RequiredParts Assertion	(Section 4.3.2)

3032 **A.4 Assertions With Undefined Policy Subject**

3033 The assertions listed in this section do not have a defined policy subject because they appear nested
 3034 inside some other assertion which does have a defined policy subject. This list is derived from nested
 3035 assertions in the specification that have independent sections. It is not a complete list of nested
 3036 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
 3037 additional nested assertions.

3038 **A.4.1 General Assertions**

3039	AlgorithmSuite Assertion	(Section 7.1)
3040	Layout Assertion	(Section 7.2)

3041 **A.4.2 Token Usage Assertions**

3042 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
 3043 assertions.

3044 **A.4.3 Token Assertions**

3045	UsernameToken Assertion	(Section 5.3.1)
------	-----------------------------------------	-----------------

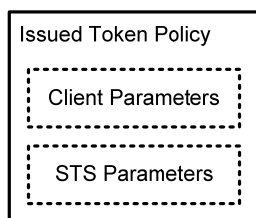
3046	IssuedToken Assertion	(Section 5.3.2)
3047	X509Token Assertion	(Section 5.3.3)
3048	KerberosToken Assertion	(Section 5.3.4)
3049	SpnegoContextToken Assertion	(Section 5.3.5)
3050	SecurityContextToken Assertion	(Section 5.3.6)
3051	SecureConversationToken Assertion	(Section 5.3.7)
3052	SamlToken Assertion	(Section 5.3.8)
3053	RelToken Assertion	(Section 5.3.9)
3054	HttpsToken Assertion	(Section 5.3.10)

3055 B. Issued Token Policy

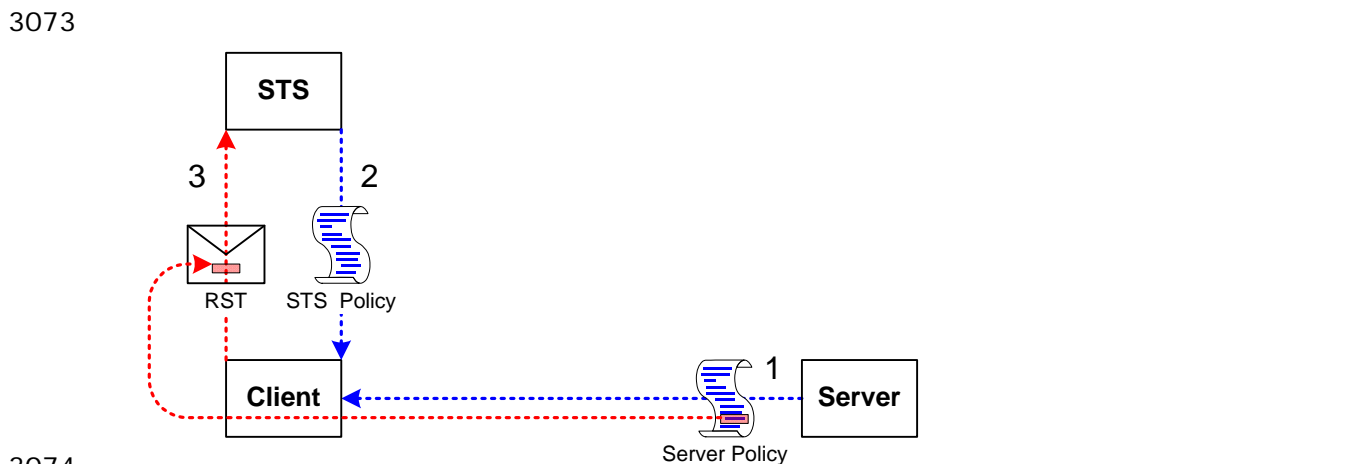
3056 The section provides further detail about behavior associated with the IssuedToken assertion in section
3057 5.3.2.

3058
3059 The issued token security model involves a three-party setup. There's a target Server, a Client, and a
3060 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from
3061 STS to Client. Policy MAY be embedded inside an Issued Token assertion, or acquired out-of-band.
3062 There MAY be an explicit trust relationship between the Server and the STS. There MUST be a trust
3063 relationship between the Client and the STS.

3064
3065 The Issued Token policy assertion includes two parts: 1) client-specific parameters that MUST be
3066 understood and processed by the client and 2) STS specific parameters which are to be processed by the
3067 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



3068
3069 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server
3070 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are
3071 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the
3072 RST request sent by the Client to the STS as illustrated in the figure below.



3074
3075 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to
3076 formulate the RST request and will include any security-specific requirements of the STS.

3077
3078 The Client MAY augment or replace the contents of the RST made to the STS based on the Client-
3079 specific parameters received from the Issued Token policy assertion contained in the Server policy, from
3080 policy it received for the STS, or any other local parameters.

3082 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The
3083 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along
3084 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
3085 request sent by the Client to the STS following the protocol defined in WS-Trust.

3086
3087 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [[WS-](#)
3088 [Trust](#)]. All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship
3089 which specifies some or all of the conditions and constraints for issued tokens.

3090 C. Strict Security Header Layout Examples

3091 The following sections describe the security header layout for specific bindings when applying the 'Strict'
3092 layout rules defined in Section 6.7.

3093 C.1 Transport Binding

3094 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

3095 C.1.1 Policy

3096 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport
3097 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username
3098 token attached to the message, and finally an X509 token attached to the message and endorsing the
3099 message signature. No message protection requirements are described since the transport covers all
3100 message parts.

```
3101 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3102   <sp:TransportBinding>
3103     <wsp:Policy>
3104       <sp:TransportToken>
3105         <wsp:Policy>
3106           <sp:HttpsToken />
3107         </wsp:Policy>
3108       </sp:TransportToken>
3109       <sp:AlgorithmSuite>
3110         <wsp:Policy>
3111           <sp:Basic256 />
3112         </wsp:Policy>
3113       </sp:AlgorithmSuite>
3114       <sp:Layout>
3115         <wsp:Policy>
3116           <sp:Strict />
3117         </wsp:Policy>
3118       </sp:Layout>
3119       <sp:IncludeTimestamp />
3120     </wsp:Policy>
3121   </sp:TransportBinding>
3122   <sp:SignedSupportingTokens>
3123     <wsp:Policy>
3124       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3125     </wsp:Policy>
3126   </sp:SignedSupportingTokens>
3127   <sp:SignedEndorsingSupportingTokens>
3128     <wsp:Policy>
3129       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3130         <wsp:Policy>
3131           <sp:WssX509v3Token10 />
3132         </wsp:Policy>
3133       </sp:X509Token>
3134     </wsp:Policy>
3135   </sp:SignedEndorsingSupportingTokens>
3136   <sp:Wss11>
3137     <sp:RequireSignatureConfirmation />
3138   </sp:Wss11>
3139 </wsp:Policy>
```

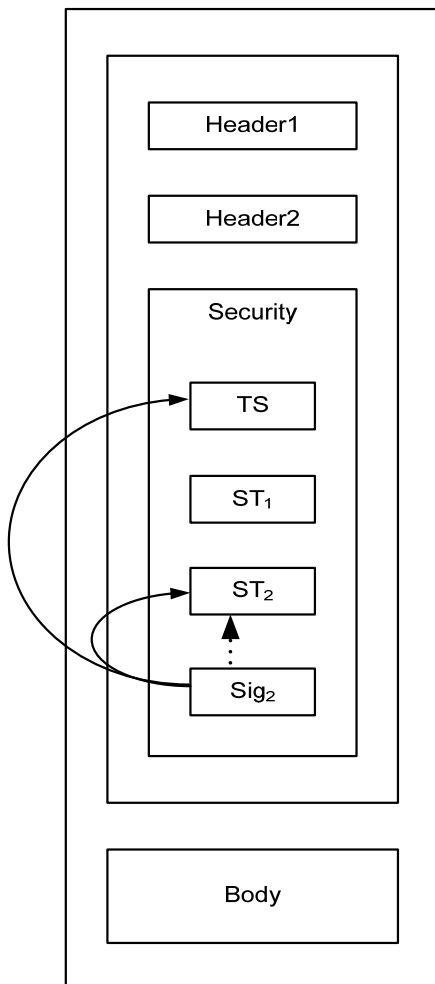
3140 This policy is used as the basis for the examples shown in the subsequent section describing the security
3141 header layout for this binding.

3142 **C.1.2 Initiator to Recipient Messages**

3143 Messages sent from initiator to recipient have the following layout for the security header:

- 3144 1. A `wsu:Timestamp` element.
- 3145 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 3146 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the
3147 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1
3148 above and **SHOULD** cover any other unique identifier for the message in order to prevent
3149 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If
3150 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a
3151 Derived Key Token, based on the supporting token, appears between the supporting token and
3152 the signature.
- 3153 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each
3154 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least
3155 some other unique identifier for the message in order to prevent replays. If [Token Protection] is
3156 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the
3157 supporting token is associated with a symmetric key, then a Derived Key Token, based on the
3158 supporting token, appears before the signature.

3159 The following diagram illustrates the security header layout for the initiator to recipient message:



3160

3161 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The
3162 arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token labeled ST₂,
3163 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST₂.
3164 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering
3165 of the items in the security header follows the most optimal layout for a receiver to process its contents.

3166 *Example:*

3167 Initiator to recipient message

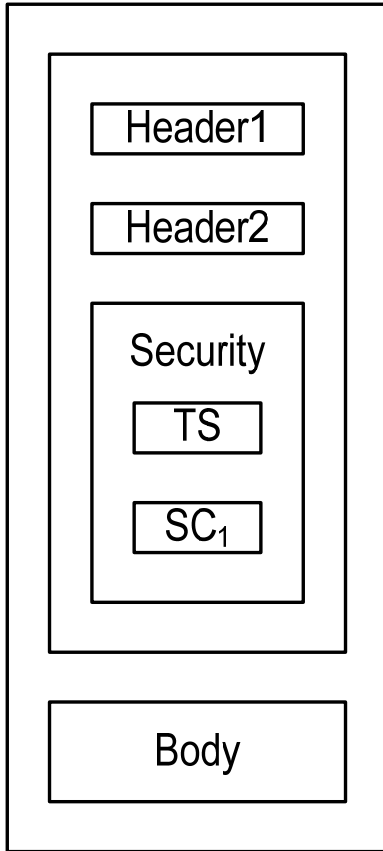
```
3168 <S:Envelope xmlns:S="..." xmlns:wss="..." xmlns:wsu="..." xmlns:ds="...">
3169   <S:Header>
3170     ...
3171     <wsse:Security>
3172       <wsu:Timestamp wsu:Id="timestamp">
3173         <wsu:Created>[datetime]</wsu:Created>
3174         <wsu:Expires>[datetime]</wsu:Expires>
3175       </wsu:Timestamp>
3176       <wsse:UsernameToken wsu:Id='SomeSignedToken' >
3177         ...
3178       </wsse:UsernameToken>
3179       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
3180         ...
3181       </wsse:BinarySecurityToken>
3182       <ds:Signature>
3183         <ds:SignedInfo>
3184           <ds:References>
3185             <ds:Reference URI="#timestamp" />
3186             <ds:Reference URI="#SomeSignedEndorsingToken" />
3187           </ds:References>
3188         </ds:SignedInfo>
3189         <ds:SignatureValue>...</ds:SignatureValue>
3190         <ds:KeyInfo>
3191           <wsse:SecurityTokenReference>
3192             <wsse:Reference URI="#SomeSignedEndorsingToken" />
3193           </wsse:SecurityTokenReference>
3194         </ds:KeyInfo>
3195       </ds:Signature>
3196     </wsse:Security>
3197     ...
3198   </S:Header>
3200   <S:Body>
3201     ...
3202   </S:Body>
3203 </S:Envelope>
```

3204 C.1.3 Recipient to Initiator Messages

3205 Messages sent from recipient to initiator have the following layout for the security header:

- 3206 1. A `wsu:Timestamp` element.
- 3207 2. If the [Signature Confirmation] property has a value of 'true', then a
3208 `wss11:SignatureConfirmation` element for each signature in the corresponding message
3209 sent from initiator to recipient. If there are no signatures in the corresponding message from the
3210 initiator to the recipient, then a `wss11:SignatureConfirmation` element with no `Value`
3211 attribute.

3212 The following diagram illustrates the security header layout for the recipient to initiator message:



3213

3214 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 3215 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 3216 message illustrated previously is included. In general, the ordering of the items in the security header
 3217 follows the most optimal layout for a receiver to process its contents.

3218 *Example:*

3219 Recipient to initiator message

```

3220 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3221   <S:Header>
3222     ...
3223     <wsse:Security>
3224       <wsu:Timestamp wsu:Id="timestamp">
3225         <wsu:Created>[datetime]</wsu:Created>
3226         <wsu:Expires>[datetime]</wsu:Expires>
3227       </wsu:Timestamp>
3228       <wsse11:SignatureConfirmation Value="..." />
3229     ...
3230   </wsse:Security>
3231   ...
3232 </S:Header>
3233 <S:Body>
3234   ...
3235 </S:Body>
3236 </S:Envelope>

```

3237 C.2 Symmetric Binding

3238 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

3239 C.2.1 Policy

3240 The following example shows a policy indicating a Symmetric Binding, a symmetric key based
3241 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message
3242 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in
3243 the message signature and the supporting signatures, a username token attached to the message, and
3244 finally an X509 token attached to the message and endorsing the message signature. Minimum message
3245 protection requirements are described as well.

```
3246 <!-- Example Endpoint Policy -->
3247 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3248   <sp:SymmetricBinding>
3249     <wsp:Policy>
3250       <sp:ProtectionToken>
3251         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3252           <sp:Issuer>...</sp:Issuer>
3253           <sp:RequestSecurityTokenTemplate>
3254             ...
3255           </sp:RequestSecurityTokenTemplate>
3256         </sp:IssuedToken>
3257       </sp:ProtectionToken>
3258       <sp:AlgorithmSuite>
3259         <wsp:Policy>
3260           <sp:Basic256 />
3261         </wsp:Policy>
3262       </sp:AlgorithmSuite>
3263       <sp:Layout>
3264         <wsp:Policy>
3265           <sp:Strict />
3266         </wsp:Policy>
3267       </sp:Layout>
3268       <sp:IncludeTimestamp />
3269       <sp:EncryptBeforeSigning />
3270       <sp:EncryptSignature />
3271       <sp:ProtectTokens />
3272     </wsp:Policy>
3273   </sp:SymmetricBinding>
3274   <sp:SignedSupportingTokens>
3275     <wsp:Policy>
3276       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3277     </wsp:Policy>
3278   </sp:SignedSupportingTokens>
3279   <sp:SignedEndorsingSupportingTokens>
3280     <wsp:Policy>
3281       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3282         <wsp:Policy>
3283           <sp:WssX509v3Token10 />
3284         </wsp:Policy>
3285       </sp:X509Token>
3286     </wsp:Policy>
3287   </sp:SignedEndorsingSupportingTokens>
3288   <sp:Wss11>
3289     <wsp:Policy>
3290       <sp:RequireSignatureConfirmation />
3291     </wsp:Policy>
3292   </sp:Wss11>
3293 </wsp:Policy>
3294
```

```

3295
3296 <!-- Example Message Policy -->
3297 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3298   <sp:SignedParts>
3299     <sp:Header Name="Header1" Namespace="..." />
3300     <sp:Header Name="Header2" Namespace="..." />
3301     <sp:Body/>
3302   </sp:SignedParts>
3303   <sp:EncryptedParts>
3304     <sp:Header Name="Header2" Namespace="..." />
3305     <sp:Body/>
3306   </sp:EncryptedParts>
3307 </wsp:Policy>

```

3308 This policy is used as the basis for the examples shown in the subsequent section describing the security
3309 header layout for this binding.

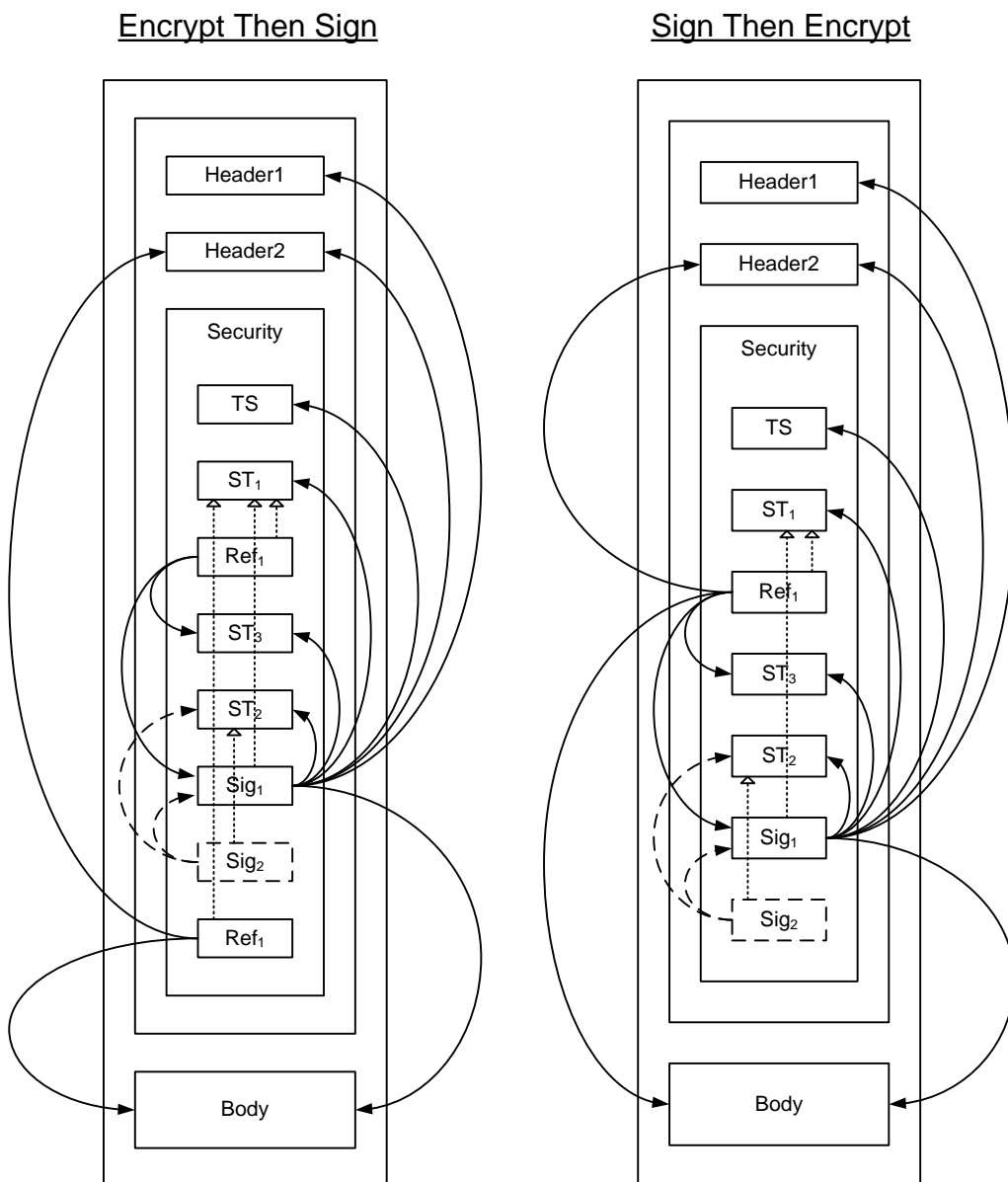
3310 C.2.2 Initiator to Recipient Messages

3311 Messages sent from initiator to recipient have the following layout for the security header:

- 3312 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3313 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
3314 `.../IncludeToken/Always`, then the [Encryption Token].
- 3315 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3316 Derived Key Token is used for encryption.
- 3317 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3318 reference list MUST include a reference to the message signature. If [Protection Order] is
3319 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3320 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3321 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3322 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3323 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
3324 `.../IncludeToken/Always`.
- 3325 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3326 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
3327 [Signature Token].
- 3328 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3329 Derived Key Token is used for signature.
- 3330 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3331 whether they are included in the message, and any message parts specified in SignedParts
3332 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3333 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3334 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3335 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3336 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3337 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3338 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3339 endorsing token, appears before the signature.
- 3340 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3341 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3342 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3343 above.

3344

3345 The following diagram illustrates the security header layout for the initiator to recipient message:



3346

3347 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
3348 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
3349 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
3350 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
3351 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
3352 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
3353 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
3354 process its contents.

3355 *Example:*

3356 Initiator to recipient message using EncryptBeforeSigning:

```
3357 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3358   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3359   xmlns:xenc="..." xmlns:ds="...">
3360   <S:Header>
3361     <x:Header1 wsu:Id="Header1" >
3362       ...
3363     </x:Header1>
3364
```



```

3365 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3366   <!-- Plaintext Header2
3367   <x:Header2 wsu:Id="Header2" >
3368     ...
3369   </x:Header2>
3370   -->
3371   ...
3372 </wsse1:EncryptedHeader>
3373 ...
3374 <wsse:Security>
3375   <wsu:Timestamp wsu:Id="Timestamp">
3376     <wsu:Created>...</wsu:Created>
3377     <wsu:Expires>...</wsu:Expires>
3378   </wsu:Timestamp>
3379   <saml:Assertion AssertionId="_SharedSecretToken" ...>
3380     ...
3381   </saml:Assertion>
3382   <xenc:ReferenceList>
3383     <xenc:DataReference URI="#enc_Signature" />
3384     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3385     ...
3386   </xenc:ReferenceList>
3387   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3388     <!-- Plaintext UsernameToken
3389     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3390       ...
3391     </wsse:UsernameToken>
3392     -->
3393     ...
3394     <ds:KeyInfo>
3395       <wsse:SecurityTokenReference>
3396         <wsse:Reference URI="#_SharedSecretToken" />
3397       </wsse:SecurityTokenReference>
3398     </ds:KeyInfo>
3399   </xenc:EncryptedData>
3400   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3401     ...
3402   </wsse:BinarySecurityToken>
3403   <xenc:EncryptedData ID="enc_Signature">
3404     <!-- Plaintext Signature
3405     <ds:Signature Id="Signature">
3406       <ds:SignedInfo>
3407         <ds:References>
3408           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3409           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3410           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3411           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3412           <ds:Reference URI="#Header1" >...</ds:Reference>
3413           <ds:Reference URI="#Header2" >...</ds:Reference>
3414           <ds:Reference URI="#Body" >...</ds:Reference>
3415         </ds:References>
3416       </ds:SignedInfo>
3417     <ds:SignatureValue>...</ds:SignatureValue>
3418     <ds:KeyInfo>
3419       <wsse:SecurityTokenReference>
3420         <wsse:Reference URI="#_SharedSecretToken" />
3421       </wsse:SecurityTokenReference>
3422     </ds:KeyInfo>
3423   </ds:Signature>
3424   -->
3425   ...
3426   <ds:KeyInfo>
3427     <wsse:SecurityTokenReference>
3428     <wsse:Reference URI="#_SharedSecretToken" />

```

```

3429     </wsse:SecurityTokenReference>
3430     </ds:KeyInfo>
3431 </xenc:EncryptedData>
3432 <ds:Signature>
3433   <ds:SignedInfo>
3434     <ds:References>
3435       <ds:Reference URI="#Signature" >...</ds:Reference>
3436       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3437     </ds:References>
3438   </ds:SignedInfo>
3439 <ds:SignatureValue>...</ds:SignatureValue>
3440 <ds:KeyInfo>
3441   <wsse:SecurityTokenReference>
3442     <wsse:Reference URI="#SomeSupportingToken" />
3443   </wsse:SecurityTokenReference>
3444 </ds:KeyInfo>
3445 </ds:Signature>
3446 <xenc:ReferenceList>
3447   <xenc:DataReference URI="#enc_Body" />
3448   <xenc:DataReference URI="#enc_Header2" />
3449   ...
3450 </xenc:ReferenceList>
3451 </wsse:Security>
3452 </S:Header>
3453 <S:Body wsu:Id="Body">
3454   <xenc:EncryptedData Id="enc_Body">
3455     ...
3456   <ds:KeyInfo>
3457     <wsse:SecurityTokenReference>
3458       <wsse:Reference URI="#_SharedSecretToken" />
3459     </wsse:SecurityTokenReference>
3460   </ds:KeyInfo>
3461 </xenc:EncryptedData>
3462 </S:Body>
3463 </S:Envelope>

```

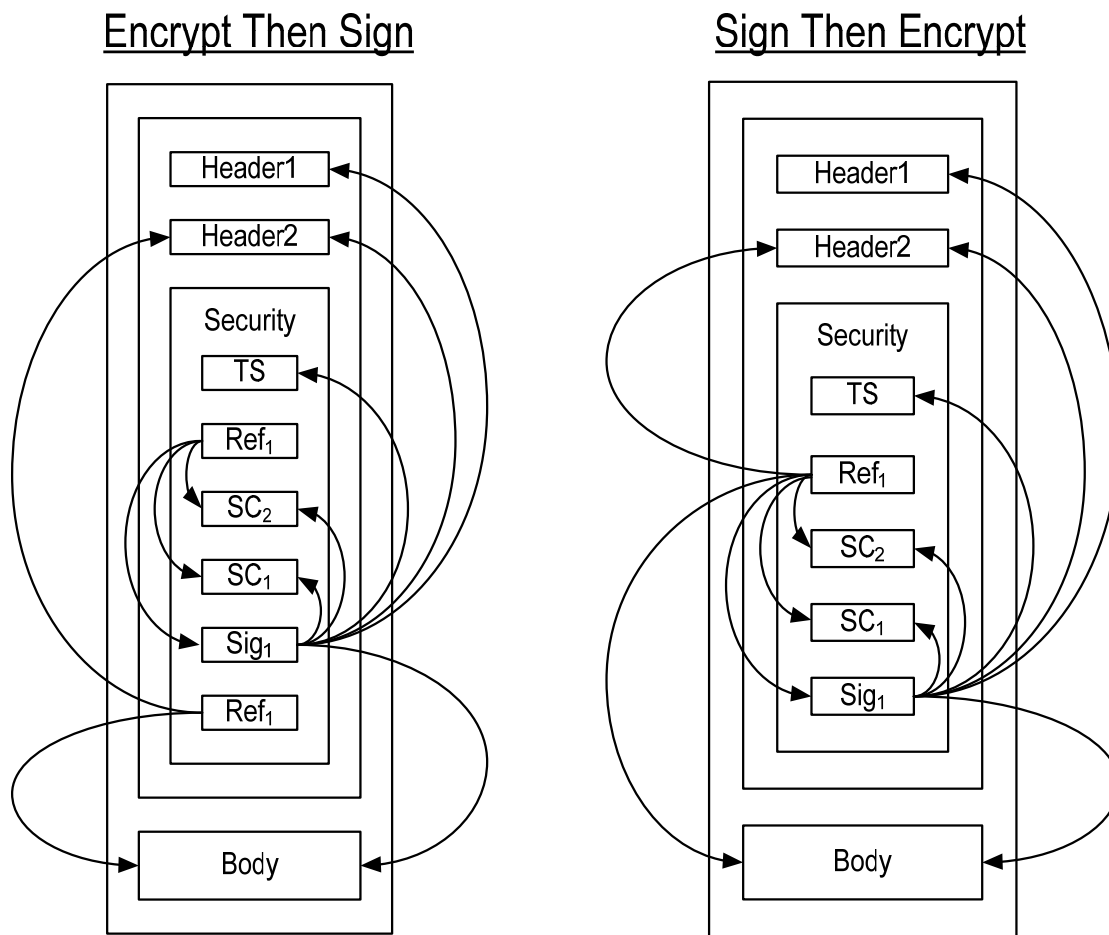
3464 C.2.3 Recipient to Initiator Messages

3465 Messages send from recipient to initiator have the following layout for the security header:

- 3466 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3467 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the
3468 [Encryption Token].
- 3469 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3470 Derived Key Token is used for encryption.
- 3471 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3472 reference list MUST include a reference to the message signature from 6 below, and the
3473 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is
3474 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3475 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3476 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2
3477 above.
- 3478 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each
3479 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3480 in the corresponding message from the initiator to the recipient, then a
3481 `wss11:SignatureConfirmation` element with no Value attribute.
- 3482 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3483 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].

- 3484 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This
 3485 Derived Key Token is used for signature.
- 3486 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation
 3487 elements from 5 above, and all the message parts specified in SignedParts assertions in the
 3488 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]
 3489 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token
 3490 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 3491 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message
 3492 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
 3493 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption
 3494 Token].

3495 The following diagram illustrates the security header layout for the recipient to initiator message:



3496

3497 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 3498 The arrows on the left from boxes labeled Ref₁ indicate references to parts encrypted using a key based
 3499 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two
 3500 wssell:SignatureConfirmation elements labeled SC₁ and SC₂ corresponding to the two signatures
 3501 in the initial message illustrated previously is included. In general, the ordering of the items in the security
 3502 header follows the most optimal layout for a receiver to process its contents. The rules used to determine
 3503 this ordering are described in Appendix C.

3504 *Example:*

3505 Recipient to initiator message using EncryptBeforeSigning:

```
3506 <S:Envelope>
3507   <S:Header>
3508     <x:Header1 wsu:Id="Header1" >
3509       ...
3510     </x:Header1>
3511     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3512       <!-- Plaintext Header2
3513       <x:Header2 wsu:Id="Header2" >
3514         ...
3515       </x:Header2>
3516       -->
3517       ...
3518     </wssell:EncryptedHeader>
3519     ...
3520   <wsse:Security>
3521     <wsu:Timestamp wsu:Id="Timestamp">
3522       <wsu:Created>...</wsu:Created>
3523       <wsu:Expires>...</wsu:Expires>
3524     </wsu:Timestamp>
3525     <xenc:ReferenceList>
3526       <xenc:DataReference URI="#enc_Signature" />
3527       <xenc:DataReference URI="#enc_SigConf1" />
3528       <xenc:DataReference URI="#enc_SigConf2" />
3529       ...
3530     </xenc:ReferenceList>
3531     <xenc:EncryptedData ID="enc_SigConf1" >
3532       <!-- Plaintext SignatureConfirmation
3533       <wssell:SignatureConfirmation wsu:Id="SigConf1" >
3534         ...
3535       </wssell:SignatureConfirmation>
3536       -->
3537       ...
3538     </xenc:EncryptedData>
3539     <xenc:EncryptedData ID="enc_SigConf2" >
3540       <!-- Plaintext SignatureConfirmation
3541       <wssell:SignatureConfirmation wsu:Id="SigConf2" >
3542         ...
3543       </wssell:SignatureConfirmation>
3544       -->
3545       ...
3546     </xenc:EncryptedData>
```

```

3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
<xenc:EncryptedData Id="enc_Signature">
  <!-- Plaintext Signature
  <ds:Signature Id="Signature">
    <ds:SignedInfo>
      <ds:References>
        <ds:Reference URI="#Timestamp" >...</ds:Reference>
        <ds:Reference URI="#SigConf1" >...</ds:Reference>
        <ds:Reference URI="#SigConf2" >...</ds:Reference>
        <ds:Reference URI="#Header1" >...</ds:Reference>
        <ds:Reference URI="#Header2" >...</ds:Reference>
        <ds:Reference URI="#Body" >...</ds:Reference>
      </ds:References>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
-->
</xenc:EncryptedData>
...
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#_SomeIssuedToken" />
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:EncryptedData>
<xenc:ReferenceList>
  <xenc:DataReference URI="#enc_Body" />
  <xenc:DataReference URI="#enc_Header2" />
  ...
</xenc:ReferenceList>
</xenc:EncryptedData>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="Body">
  <xenc:EncryptedData Id="enc_Body">
    ...
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </xenc:EncryptedData>
</S:Body>
</S:Envelope>

```

3596 C.3 Asymmetric Binding

3597 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

3598 C.3.1 Policy

3599 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator
3600 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the
3601 message parts before signing, a requirement to encrypt the message signature, a requirement to include
3602 tokens in the message signature and the supporting signatures, a requirement to include
3603 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3604 an X509 token attached to the message and endorsing the message signature. Minimum message
3605 protection requirements are described as well.

```
3606 <!-- Example Endpoint Policy -->  
3607 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
3608   <sp:AsymmetricBinding>  
3609     <wsp:Policy>  
3610       <sp:RecipientToken>  
3611         <wsp:Policy>  
3612           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3613         </wsp:Policy>  
3614       </sp:RecipientToken>  
3615       <sp:InitiatorToken>  
3616         <wsp:Policy>  
3617           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3618         </wsp:Policy>  
3619       </sp:InitiatorToken>  
3620       <sp:AlgorithmSuite>  
3621         <wsp:Policy>  
3622           <sp:Basic256 />  
3623         </wsp:Policy>  
3624       </sp:AlgorithmSuite>  
3625       <sp:Layout>  
3626         <wsp:Policy>  
3627           <sp:Strict />  
3628         </wsp:Policy>  
3629       </sp:Layout>  
3630       <sp:IncludeTimestamp />  
3631       <sp:EncryptBeforeSigning />  
3632       <sp:EncryptSignature />  
3633       <sp:ProtectTokens />  
3634     </wsp:Policy>  
3635   </sp:AsymmetricBinding>  
3636   <sp:SignedEncryptedSupportingTokens>  
3637     <wsp:Policy>  
3638       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />  
3639     </wsp:Policy>  
3640   </sp:SignedEncryptedSupportingTokens>  
3641   <sp:SignedEndorsingSupportingTokens>  
3642     <wsp:Policy>  
3643       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">  
3644         <wsp:Policy>  
3645           <sp:WssX509v3Token10 />  
3646         </wsp:Policy>  
3647       </sp:X509Token>  
3648     </wsp:Policy>  
3649   </sp:SignedEndorsingSupportingTokens>  
3650   <sp:Wss11>  
3651     <wsp:Policy>  
3652       <sp:RequireSignatureConfirmation />  
3653     </wsp:Policy>  
3654   </sp:Wss11>  
3655 </wsp:Policy>  
3656
```

3657

```
3658 <!-- Example Message Policy -->
3659 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3660   <sp:SignedParts>
3661     <sp:Header Name="Header1" Namespace="..." />
3662     <sp:Header Name="Header2" Namespace="..." />
3663     <sp:Body/>
3664   </sp:SignedParts>
3665   <sp:EncryptedParts>
3666     <sp:Header Name="Header2" Namespace="..." />
3667     <sp:Body/>
3668   </sp:EncryptedParts>
3669 </wsp:All>
```

3670

3671 This policy is used as the basis for the examples shown in the subsequent section describing the security
3672 header layout for this binding.

3673 **C.3.2 Initiator to Recipient Messages**

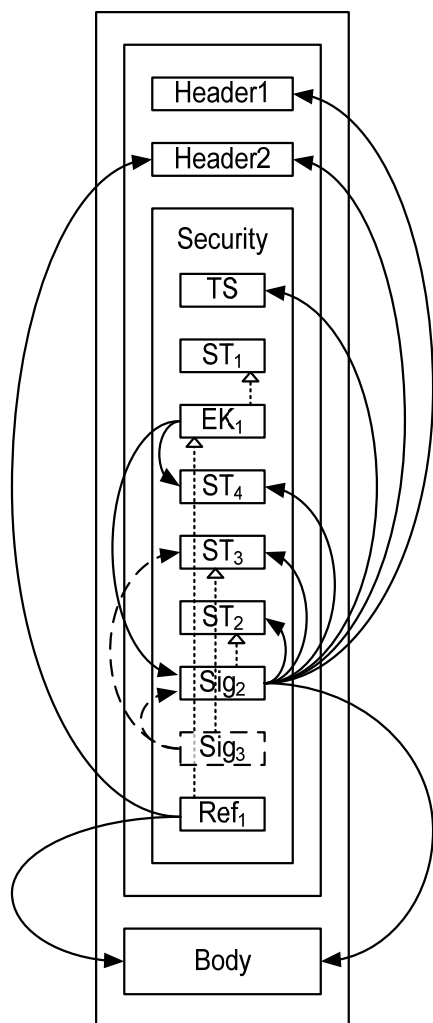
3674 Messages sent from initiator to recipient have the following layout:

- 3675 1. A `wsu:Timestamp` element if `[Timestamp]` is 'true'.
- 3676 2. If a `[Recipient Token]` is specified, and the associated `sp:IncludeToken` attribute is
3677 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Recipient Token]`.
- 3678 3. If a `[Recipient Token]` is specified and `[Protection Order]` is 'SignBeforeEncrypting' or
3679 `[SignatureProtection]` is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3680 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3681 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If
3682 `[Signature Protection]` is 'true' then the reference list MUST contain a reference to the message
3683 signature from 6 below. It is an error if `[Signature Protection]` is 'true' and there is not a message
3684 signature.
- 3685 4. Any tokens from the supporting tokens properties (as defined in section 8) whose
3686 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3687 5. If an `[Initiator Token]` is specified, and the associated `sp:IncludeToken` attribute is
3688 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Initiator Token]`.
- 3689 6. A signature based on the key in the `[Initiator Token]` if specified, over the `wsu:Timestamp` from
3690 1 above, any tokens from 4 above regardless of whether they are included in the message, and
3691 any message parts specified in `SignedParts` assertions in the policy. If `[Token Protection]` is 'true',
3692 the signature MUST also cover the `[Initiator Token]` regardless of whether it is included in the
3693 message.
- 3694 7. Signatures for tokens from the `[Endorsing Supporting Tokens]` and `[Signed Endorsing Supporting`
3695 `Tokens]` properties. If `[Derived Keys]` is 'true' and the supporting token is associated with a
3696 symmetric key, then a `Derived Key Token`, based on the supporting token, appears before the
3697 signature. If `[Token Protection]` is 'true', the signature MUST also cover the supporting token
3698 regardless of whether it is included in the message.
- 3699 8. If a `[Recipient Token]` is specified and `[Protection Order]` is 'EncryptBeforeSigning' then if
3700 `[Signature Protection]` is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
3701 for the recipient and a reference list, else if `[Signature Protection]` is 'true', a reference list. The
3702 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions
3703 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
3704 element from 3 above.

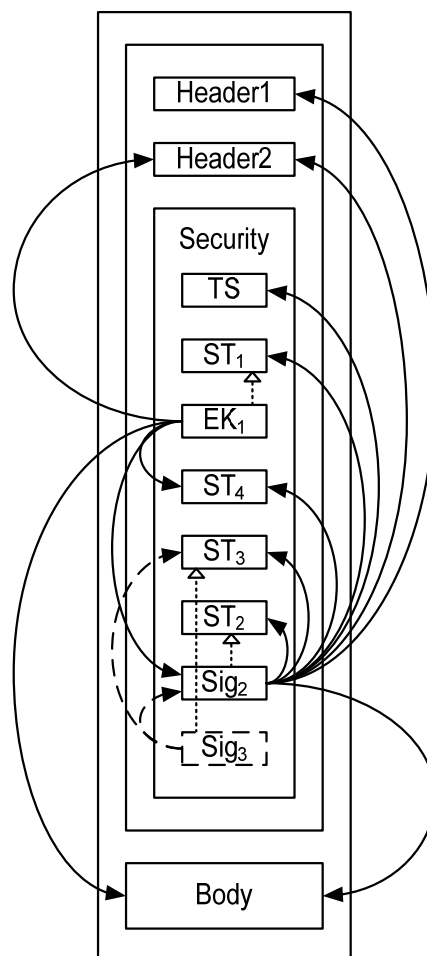
3705

3706 The following diagram illustrates the security header layout for the initiator to recipient messages:

Encrypt Then Sign



Sign Then Encrypt



3707

3708 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂

3709 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate

3710 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as

3711 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references

3712 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left

3713 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the

3714 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as

3715 the basis for each cryptographic operation. In general, the ordering of the items in the security header

3716 follows the most optimal layout for a receiver to process its contents. The rules used to determine this

3717 ordering are described in Appendix C.

3718

3719 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted

3720 key contains an external reference to the token containing the encryption key. The diagram illustrates

3721 how one might attach a security token related to the encrypted key for completeness. One possible use-

3722 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3723 wishes to include the encryption token in the message signature.

3724 Initiator to recipient message *Example*

3725 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3726     xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3727 <S:Header>
3728   <x:Header1 wsu:Id="Header1" >
3729     ...
3730   </x:Header1>
3731   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3732     <!-- Plaintext Header2
3733     <x:Header2 wsu:Id="Header2" >
3734       ...
3735     </x:Header2>
3736     -->
3737     ...
3738   </wssell1:EncryptedHeader>
3739   ...
3740   <wsse:Security>
3741     <wsu:Timestamp wsu:Id="Timestamp">
3742       <wsu:Created>...</wsu:Created>
3743       <wsu:Expires>...</wsu:Expires>
3744     </wsu:Timestamp>
3745     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3746       ...
3747     </wsse:BinarySecurityToken>
3748     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3749       ...
3750     <xenc:ReferenceList>
3751       <xenc:DataReference URI="#enc_Signature" />
3752       <xenc:DataReference URI="#enc_SomeUsernameToken" />
3753       ...
3754     </xenc:ReferenceList>
3755   </xenc:EncryptedKey>
3756   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3757     <!-- Plaintext UsernameToken
3758     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3759       ...
3760     </wsse:UsernameToken>
3761     -->
3762     ...
3763   </xenc:EncryptedData>
3764   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3765     ...
3766   </wsse:BinarySecurityToken>
3767   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3768     ...
3769   </wsse:BinarySecurityToken>
3770   <xenc:EncryptedData ID="enc_Signature">
3771     <!-- Plaintext Signature
3772     <ds:Signature Id="Signature">
3773       <ds:SignedInfo>
3774         <ds:References>
3775           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3776           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3777           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3778           <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3779           <ds:Reference URI="#Header1" >...</ds:Reference>
3780           <ds:Reference URI="#Header2" >...</ds:Reference>
3781           <ds:Reference URI="#Body" >...</ds:Reference>
3782         </ds:References>
3783       </ds:SignedInfo>
3784     <ds:SignatureValue>...</ds:SignatureValue>
3785     <ds:KeyInfo>
3786       <wsse:SecurityTokenReference>
3787         <wsse:Reference URI="#InitiatorToken" />
3788       </wsse:SecurityTokenReference>
3789     </ds:KeyInfo>

```

```

3790     </ds:Signature>
3791     -->
3792     ...
3793 </xenc:EncryptedData>
3794 <ds:Signature>
3795   <ds:SignedInfo>
3796     <ds:References>
3797       <ds:Reference URI="#Signature" >...</ds:Reference>
3798       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3799     </ds:References>
3800   </ds:SignedInfo>
3801   <ds:SignatureValue>...</ds:SignatureValue>
3802   <ds:KeyInfo>
3803     <wsse:SecurityTokenReference>
3804       <wsse:Reference URI="#SomeSupportingToken" />
3805     </wsse:SecurityTokenReference>
3806   </ds:KeyInfo>
3807 </ds:Signature>
3808 <xenc:ReferenceList>
3809   <xenc:DataReference URI="#enc_Body" />
3810   <xenc:DataReference URI="#enc_Header2" />
3811   ...
3812 </xenc:ReferenceList>
3813 </wsse:Security>
3814 </S:Header>
3815 <S:Body wsu:Id="Body">
3816   <xenc:EncryptedData Id="enc_Body">
3817     ...
3818     <ds:KeyInfo>
3819       <wsse:SecurityTokenReference>
3820         <wsse:Reference URI="#RecipientEncryptedKey" />
3821       </wsse:SecurityTokenReference>
3822     </ds:KeyInfo>
3823   </xenc:EncryptedData>
3824 </S:Body>
3825 </S:Envelope>

```

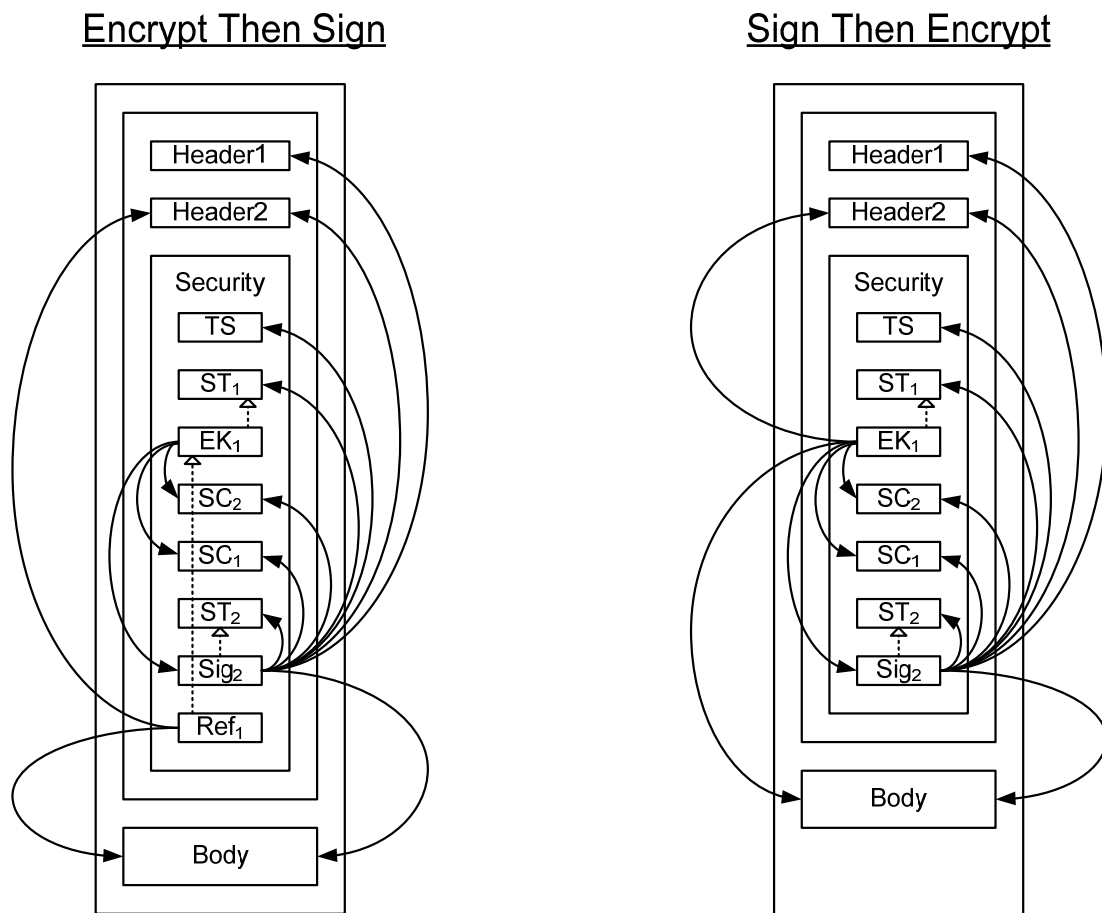
3826 C.3.3 Recipient to Initiator Messages

3827 Messages sent from recipient to initiator have the following layout:

- 3828 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3829 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3830 `.../IncludeToken/Always`, then the [Initiator Token].
- 3831 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3832 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3833 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3834 reference to all the message parts specified in EncryptedParts assertions in the policy. If
3835 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3836 message signature from 6 below, if any and references to the
3837 `wsse11:SignatureConfirmation` elements from 4 below, if any.
- 3838 4. If [Signature Confirmation] is 'true', then a `wsse11:SignatureConfirmation` element for each
3839 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3840 in the corresponding message from the initiator to the recipient, then a
3841 `wsse11:SignatureConfirmation` element with no Value attribute.
- 3842 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3843 `.../IncludeToken/Always`, then the [Recipient Token].

- 3844 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],
 3845 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements
 3846 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token
 3847 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3848 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
 3849 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted
 3850 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
 3851 reference list includes a reference to all the message parts specified in EncryptedParts assertions
 3852 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`
 3853 element from 3 above.

3854
 3855 The following diagram illustrates the security header layout for the recipient to initiator messages:



3856
 3857 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
 3858 using the [Recipient Token] labeled ST₂. The arrows on the left from boxes labeled EK₁ indicate
 3859 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on
 3860 the left from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained
 3861 in the encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token
 3862 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements
 3863 labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is
 3864 included. In general, the ordering of the items in the security header follows the most optimal layout for a
 3865 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.
 3866 Recipient to initiator message *Example*:

```

3867 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3868     xmlns:wssell="..." xmlns:wsse="..."
3869     xmlns:xenc="..." xmlns:ds="...">
3870 <S:Header>
3871     <x:Header1 wsu:Id="Header1" >
3872         ...
3873     </x:Header1>
3874     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3875         <!-- Plaintext Header2
3876         <x:Header2 wsu:Id="Header2" >
3877             ...
3878         </x:Header2>
3879         -->
3880         ...
3881     </wssell:EncryptedHeader>
3882     ...
3883     <wsse:Security>
3884         <wsu:Timestamp wsu:Id="Timestamp">
3885             <wsu:Created>...</wsu:Created>
3886             <wsu:Expires>...</wsu:Expires>
3887         </wsu:Timestamp>
3888         <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3889             ...
3890         </wsse:BinarySecurityToken>
3891         <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3892             ...
3893             <xenc:ReferenceList>
3894                 <xenc:DataReference URI="#enc_Signature" />
3895                 <xenc:DataReference URI="#enc_SigConf1" />
3896                 <xenc:DataReference URI="#enc_SigConf2" />
3897                 ...
3898             </xenc:ReferenceList>
3899         </xenc:EncryptedKey>
3900         <xenc:EncryptedData ID="enc_SigConf2" >
3901             <!-- Plaintext SignatureConfirmation
3902             <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3903                 ...
3904             </wssell:SignatureConfirmation>
3905             -->
3906             ...
3907         </xenc:EncryptedData>
3908         <xenc:EncryptedData ID="enc_SigConf1" >
3909             <!-- Plaintext SignatureConfirmation
3910             <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3911                 ...
3912             </wssell:SignatureConfirmation>
3913             -->
3914             ...
3915         </xenc:EncryptedData>
3916         <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3917             ...
3918         </wsse:BinarySecurityToken>
3919

```

```

3920 <xenc:EncryptedData ID="enc_Signature">
3921 <!-- Plaintext Signature
3922 <ds:Signature Id="Signature">
3923 <ds:SignedInfo>
3924 <ds:References>
3925 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3926 <ds:Reference URI="#SigConf1" >...</ds:Reference>
3927 <ds:Reference URI="#SigConf2" >...</ds:Reference>
3928 <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3929 <ds:Reference URI="#Header1" >...</ds:Reference>
3930 <ds:Reference URI="#Header2" >...</ds:Reference>
3931 <ds:Reference URI="#Body" >...</ds:Reference>
3932 </ds:References>
3933 </ds:SignedInfo>
3934 <ds:SignatureValue>...</ds:SignatureValue>
3935 <ds:KeyInfo>
3936 <wsse:SecurityTokenReference>
3937 <wsse:Reference URI="#RecipientToken" />
3938 </wsse:SecurityTokenReference>
3939 </ds:KeyInfo>
3940 </ds:Signature>
3941 -->
3942 ...
3943 </xenc:EncryptedData>
3944 <xenc:ReferenceList>
3945 <xenc:DataReference URI="#enc_Body" />
3946 <xenc:DataReference URI="#enc_Header2" />
3947 ...
3948 </xenc:ReferenceList>
3949 </wsse:Security>
3950 </S:Header>
3951 <S:Body wsu:Id="Body">
3952 <xenc:EncryptedData Id="enc_Body">
3953 ...
3954 <ds:KeyInfo>
3955 <wsse:SecurityTokenReference>
3956 <wsse:Reference URI="#InitiatorEncryptedKey" />
3957 </wsse:SecurityTokenReference>
3958 </ds:KeyInfo>
3959 </xenc:EncryptedData>
3960 </S:Body>
3961 </S:Envelope>

```

3962
3963

D. Signed and Encrypted Elements in the Security Header

3964
3965
3966
3967
3968

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

3969

D.1 Elements signed by the message signature

3970
3971
3972
3973
3974
3975
3976

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

3977

D.2 Elements signed by all endorsing signatures

3978
3979

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

3980

D.3 Elements signed by a specific endorsing signature

3981
3982

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

3983

D.4 Elements that are encrypted

3984
3985
3986
3987
3988
3989
3990

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

3991

E. Acknowledgements

3992 The following individuals have participated in the creation of this specification and are gratefully
3993 acknowledged:

3994 **Original Authors of the initial contribution:**

3995 Giovanni Della-Libera, Microsoft

3996 Martin Gudgin, Microsoft

3997 Phillip Hallam-Baker, VeriSign

3998 Maryann Hondo, IBM

3999 Hans Granqvist, Verisign

4000 Chris Kaler, Microsoft (editor)

4001 Hiroshi Maruyama, IBM

4002 Michael McIntosh, IBM

4003 Anthony Nadalin, IBM (editor)

4004 Nataraj Nagaratnam, IBM

4005 Rob Philpott, RSA Security

4006 Hemma Prafullchandra, VeriSign

4007 John Shewchuk, Microsoft

4008 Doug Walter, Microsoft

4009 Riaz Zolfonoon, RSA Security

4010

4011 **Original Acknowledgements of the initial contribution:**

4012 Vaithialingam B. Balayoghan, Microsoft

4013 Francisco Curbera, IBM

4014 Christopher Ferris, IBM

4015 Cédric Fournet, Microsoft

4016 Andy Gordon, Microsoft

4017 Tomasz Janczuk, Microsoft

4018 David Melgar, IBM

4019 Mike Perks, IBM

4020 Bruce Rich, IBM

4021 Jeffrey Schlimmer, Microsoft

4022 Chris Sharp, IBM

4023 Kent Tamura, IBM

4024 T.R. Vishwanath, Microsoft

4025 Elliot Waingold, Microsoft

4026

4027 **TC Members during the development of this specification:**

4028 Don Adams, Tibco Software Inc.

4029 Jan Alexander, Microsoft Corporation

4030 Steve Anderson, BMC Software

4031 Donal Arundel, IONA Technologies

4032 Howard Bae, Oracle Corporation

4033 Abbie Barbir, Nortel Networks Limited

4034 Charlton Barreto, Adobe Systems

4035 Michael Botha, Software AG, Inc.

4036 Toufic Boubez, Layer 7 Technologies Inc.

4037 Norman Brickman, Mitre Corporation

4038 Melissa Brumfield, Booz Allen Hamilton

4039 Geoff Bullen, Microsoft Corporation
4040 Lloyd Burch, Novell
4041 Scott Cantor, Internet2
4042 Greg Carpenter, Microsoft Corporation
4043 Steve Carter, Novell
4044 Symon Chang, Oracle Corporation Ching-Yun (C.Y.) Chao, IBM
4045 Martin Chapman, Oracle Corporation
4046 Kate Cherry, Lockheed Martin
4047 Henry (Hyenvui) Chung, IBM
4048 Luc Clement, Systinet Corp.
4049 Paul Cotton, Microsoft Corporation
4050 Glen Daniels, Sonic Software Corp.
4051 Peter Davis, Neustar, Inc.
4052 Duane DeCouteau, Veterans Health Administration
4053 Martijn de Boer, SAP AG
4054 Werner Dittmann, Siemens AG
4055 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
4056 Fred Dushin, IONA Technologies
4057 Petr Dvorak, Systinet Corp.
4058 Colleen Evans, Microsoft Corporation
4059 Ruchith Fernando, WSO2
4060 Mark Fussell, Microsoft Corporation
4061 Vijay Gajjala, Microsoft Corporation
4062 Marc Goodner, Microsoft Corporation
4063 Hans Granqvist, VeriSign
4064 Martin Gudgin, Microsoft Corporation
4065 Tony Gullotta, SOA Software Inc.
4066 Jiandong Guo, Sun Microsystems
4067 Phillip Hallam-Baker, VeriSign
4068 Patrick Harding, Ping Identity Corporation
4069 Heather Hinton, IBM
4070 Frederick Hirsch, Nokia Corporation
4071 Jeff Hodges, Neustar, Inc.
4072 Will Hopkins, Oracle Corporation
4073 Alex Hristov, Otecia Incorporated
4074 John Hughes, PA Consulting
4075 Diane Jordan, IBM
4076 Venugopal K, Sun Microsystems
4077 Chris Kaler, Microsoft Corporation
4078 Dana Kaufman, Forum Systems, Inc.
4079 Paul Knight, Nortel Networks Limited
4080 Ramanathan Krishnamurthy, IONA Technologies
4081 Christopher Kurt, Microsoft Corporation
4082 Kelvin Lawrence, IBM
4083 Hubert Le Van Gong, Sun Microsystems
4084 Jong Lee, Oracle Corporation
4085 Rich Levinson, Oracle Corporation
4086 Tommy Lindberg, Dajeil Ltd.
4087 Mark Little, JBoss Inc.
4088 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.
4089 Eve Maler, Sun Microsystems
4090 Ashok Malhotra, Oracle Corporation
4091 Anand Mani, CrimsonLogic Pte Ltd
4092 Jonathan Marsh, Microsoft Corporation
4093 Robin Martherus, Oracle Corporation
4094 Miko Matsumura, Infravio, Inc.
4095 Gary McAfee, IBM

4096 Michael McIntosh, IBM
4097 John Merrells, Sxip Networks SRL
4098 Jeff Mischkinsky, Oracle Corporation
4099 Prateek Mishra, Oracle Corporation
4100 Bob Morgan, Internet2
4101 Vamsi Motukuru, Oracle Corporation
4102 Raajmohan Na, EDS
4103 Anthony Nadalin, IBM
4104 Andrew Nash, Reactivity, Inc.
4105 Eric Newcomer, IONA Technologies
4106 Duane Nickull, Adobe Systems
4107 Toshihiro Nishimura, Fujitsu Limited
4108 Rob Philpott, RSA Security
4109 Denis Pilipchuk, Oracle Corporation.
4110 Darren Platt, Ping Identity Corporation
4111 Martin Raepple, SAP AG
4112 Nick Ragouzis, Enosis Group LLC
4113 Prakash Reddy, CA
4114 Alain Regnier, Ricoh Company, Ltd.
4115 Irving Reid, Hewlett-Packard
4116 Bruce Rich, IBM
4117 Tom Rutt, Fujitsu Limited
4118 Maneesh Sahu, Actional Corporation
4119 Frank Siebenlist, Argonne National Laboratory
4120 Joe Smith, Apani Networks
4121 Davanum Srinivas, WSO2
4122 David Staggs, Veterans Health Administration
4123 Yakov Sverdlov, CA
4124 Gene Thurston, AmberPoint
4125 Victor Valle, IBM
4126 Asir Vedamuthu, Microsoft Corporation
4127 Greg Whitehead, Hewlett-Packard
4128 Ron Williams, IBM
4129 Corinna Witt, Oracle Corporation
4130 Kyle Young, Microsoft Corporation
4131
4132