



WS-SecurityPolicy 1.3

OASIS Committee Draft 01

9 July 2008

Specification URIs:

This Version:

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

Previous Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.doc>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.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/>
```

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

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

1.2 Namespaces

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

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

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

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]

1.3 Schema Files

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

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

1.4 Terminology

Policy - A collection of policy alternatives.

Policy Alternative - A collection of policy assertions.

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

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

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

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

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

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

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

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

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

Supporting Token - A token used to provide additional claims.

1.4.1 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

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

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

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

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

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

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

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

1.5 Normative References

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

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		http://www.ietf.org/rfc/rfc3986.txt
150		
151		
152		
153	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
154		http://www.ietf.org/rfc/rfc2068.txt
155		
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		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
165		
166		
167	[WS-Policy]	W3C Recommendation, "Web Services Policy 1.5 - Framework", 04 September 2007.
168		http://www.w3.org/TR/2007/REC-ws-policy-20070904/
169		
170		W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
171		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
172		
173		
174	[WS-PolicyAttachment]	W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04 September 2007.
175		http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
176		
177		W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
178		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
179		
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
239		(REL) Token Profile 1.1", February 2006
240		http://www.oasis-open.org/committees/download.php/16687/oasis-
241		wss-rel-token-profile-1.1.pdf
242		
243	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with
244		Attachments (SwA) Profile 1.1", February 2006
245		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-
246		spec-os-SwAProfile.pdf
247		
248	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10
249		December 2002.
250		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
251		
252	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12
253		February 2002.
254		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
255		
256	[XPath]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16
257		November 1999.
258		http://www.w3.org/TR/1999/REC-xpath-19991116
259		
260	[XPath 2.0 Filter]	W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November
261		2002.
262		http://www.w3.org/TR/2002/REC-xmldsig-filter2-20021108/
263		
264	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second
265		Edition", 28 October 2004.
266		http://www.w3.org/TR/2004/REC-xmldsig-filter2-20021108/
267		
268	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second
269		Edition", 28 October 2004.
270		http://www.w3.org/TR/2004/REC-xmldsig-filter2-20021108/
271		

272 1.6 Non-Normative References

273 None.

274

2 Security Policy Model

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

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

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

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

2.1 Security Assertion Model

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

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

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

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

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

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

2.2 Nested Policy Assertions

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

2.3 Security Binding Abstraction

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

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

A binding defines the following security characteristics:

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

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

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

- 364 • TransportBinding (Section 7.3)
- 365 • SymmetricBinding (Section 7.4)
- 366 • AsymmetricBinding (Section 7.5)

3 Policy Considerations

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

3.1 Nested Policy

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

3.2 Policy Subjects

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

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

[Message Policy Subject]

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

wsdl:message

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

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

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

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

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

[Operation Policy Subject]

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

wsdl:portType/wsdl:operation

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

wsdl:binding/wsdl:operation

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

[Endpoint Policy Subject]

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

wsdl:portType

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

408 wsdl:binding

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

411 wsdl:port

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

4 Protection Assertions

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

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

4.1 Integrity Assertions

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

4.1.1 SignedParts Assertion

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

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

Syntax

```
<sp:SignedParts xmlns:sp="..." ... >
  <sp:Body />?
  <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
  <sp:Attachments>
    <sp13:ContentSignatureTransform /> ?
    <sp13:AttachmentCompleteSignatureTransform /> ?
  </sp:Attachments> ?
  ...
</sp:SignedParts>
```

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

/sp:SignedParts

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

/sp:SignedParts/sp:Body

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

/sp:SignedParts/sp:Header

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

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

/sp:SignedParts/sp:Header/@Name

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

/sp:SignedParts/sp:Header/@Namespace

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

/sp:SignedParts/sp:Attachments

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

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

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

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

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

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

4.1.2 SignedElements Assertion

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

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

Syntax

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

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

521 4.2 Confidentiality Assertions

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

525 4.2.1 EncryptedParts Assertion

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

531
532 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present
533 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all
534 specified message parts. Note that this assertion does not require that a given part appear in a message,
535 just that if such a part appears, it requires confidentiality protection.

536 Syntax

```
537 <sp:EncryptedParts xmlns:sp="..." ... >  
538   <sp:Body/>?  
539   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />?  
540   <sp:Attachments />?  
541   ...  
542 </sp:EncryptedParts>
```

543
544 The following describes the attributes and elements listed in the schema outlined above:
545 /sp:EncryptedParts

546 This assertion specifies the parts of the message that need confidentiality protection. The single
 547 child element of this assertion specifies the set of message parts using an extensible dialect.
 548 If no child elements are specified, the body of the message MUST be confidentiality protected.
 549 /sp:EncryptedParts/sp:Body
 550 Presence of this OPTIONAL empty element indicates that the entire body of the message needs
 551 to be confidentiality protected. In the case where mechanisms from WSS: SOAP Message
 552 Security are used to satisfy this assertion, then the soap:Body element is encrypted using the
 553 #Content encryption type.
 554 /sp:EncryptedParts/sp:Header
 555 Presence of this OPTIONAL element indicates that a specific SOAP header (or set of such
 556 headers) needs to be protected. There may be multiple sp:Header elements within a single Parts
 557 element. Each header or set of headers MUST be encrypted. Such encryption will encrypt such
 558 elements using WSS 1.1 Encrypted Headers. As such, if WSS 1.1 Encrypted Headers are not
 559 supported by a service, then this element cannot be used to specify headers that require
 560 encryption using message level security. If multiple SOAP headers with the same local name but
 561 different namespace names are to be encrypted then multiple sp:Header elements are needed,
 562 either as part of a single sp:EncryptedParts assertion or as part of separate sp:EncryptedParts
 563 assertions.
 564 /sp:EncryptedParts/sp:Header/@Name
 565 This OPTIONAL attribute indicates the local name of the SOAP header to be confidentiality
 566 protected. If this attribute is not specified, all SOAP headers whose namespace matches the
 567 Namespace attribute are to be protected.
 568 /sp:EncryptedParts/sp:Header/@Namespace
 569 This REQUIRED attribute indicates the namespace of the SOAP header(s) to be confidentiality
 570 protected.
 571 /sp:EncryptedParts/sp:Attachments
 572 Presence of this OPTIONAL empty element indicates that all SwA (SOAP Messages with
 573 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message
 574 Security is used to accomplish this, all message parts other than the part containing the primary
 575 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security
 576 [WSS:SwAProfile1.1].

577 4.2.2 EncryptedElements Assertion

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

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

587 Syntax

```
588 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
589   <sp:XPath>xs:string</sp:XPath>+
590   ...
591 </sp:EncryptedElements>
```

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

`/sp:EncryptedElements`

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

`/sp:EncryptedElements/@XPathVersion`

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

`/sp:EncryptedElements/sp:XPath`

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

4.2.3 ContentEncryptedElements Assertion

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

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

Syntax

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

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

`/sp:ContentEncryptedElements`

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

`/sp:ContentEncryptedElements/@XPathVersion`

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

`/sp:ContentEncryptedElements/sp:XPath`

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

4.3 Required Elements Assertion

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

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

4.3.1 RequiredElements Assertion

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

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

Syntax

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

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

/sp:RequiredElements

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

/sp:RequiredElements/@XPathVersion

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

/sp:RequiredElements/sp:XPath

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

4.3.2 RequiredParts Assertion

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

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

Syntax

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

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

/sp:RequiredParts/sp:Header

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

/sp:RequiredParts/sp:Header/@Name

679 This REQUIRED attribute indicates the local name of the SOAPHeader that needs to be present
680 in the message.
681 /sp:RequiredParts/sp:Header/@Namespace
682 This REQUIRED attribute indicates the namespace of the SOAP header that needs to be present
683 in the message.

5 Token Assertions

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

5.1 Token Inclusion

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

5.1.1 Token Inclusion Values

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

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

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

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

5.1.2 Token Inclusion and Token References

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

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

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

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

5.2 Token Issuer and Required Claims

5.2.1 Token Issuer

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

5.2.2 Token Issuer Name

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

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

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

5.2.3 Required Claims

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

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

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

5.2.4 Processing Rules and Token Matching

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

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

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

5.3 Token Properties

5.3.1 [Derived Keys] Property

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

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

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

5.3.2 [Explicit Derived Keys] Property

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

5.3.3 [Implied Derived Keys] Property

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

5.4 Token Assertion Types

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

5.4.1 UsernameToken Assertion

This element represents a requirement to include a username token.

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

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

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

Syntax

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

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

/sp:UsernameToken

This identifies a UsernameToken assertion.

/sp:UsernameToken/@sp:IncludeToken

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

/sp:UsernameToken/sp:Issuer

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

/sp:UsernameToken/sp:IssuerName

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

/sp:UsernameToken/wst:Claims

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

842 /sp:UsernameToken/wsp:Policy

843 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken
844 assertion.

845 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

848 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

853 /sp13:UsernameToken/wsp:Policy/sp13:Created

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

857 /sp13:UsernameToken/wsp:Policy/sp13:Nonce

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

861 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

864 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

868 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

872 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

875 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

878 5.4.2 ICreatessuedToken Assertion

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

883 Syntax

```

884 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
885 (
886   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
887   <sp:IssuerName>xs:anyURI</sp:IssuerName>
888 ) ?
889 <wst:Claims Dialect="..."> ... </wst:Claims> ?
890 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
891   ...
892 </sp:RequestSecurityTokenTemplate>
893 <wsp:Policy xmlns:wsp="...">
894   (
895     <sp:RequireDerivedKeys ... /> |
896     <sp:RequireImpliedDerivedKeys ... /> |
897     <sp:RequireExplicitDerivedKeys ... />
898   ) ?
899   <sp:RequireExternalReference ... /> ?
900   <sp:RequireInternalReference ... /> ?
901   ...
902 </wsp:Policy>
903   ...
904 </sp:IssuedToken>

```

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

/sp:IssuedToken

This identifies an IssuedToken assertion.

/sp:IssuedToken/@sp:IncludeToken

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

/sp:IssuedToken/sp:Issuer

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

/sp:IssuedToken/sp:IssuerName

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

/sp:IssuedToken/wst:Claims

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

/sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

/sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

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

/sp:IssuedToken/wsp:Policy

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

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

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

935 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

939 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

943 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

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

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

947 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

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

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

951 Note: The IssuedToken MAY or MAY NOT be associated with key material and such key material may be
952 symmetric or asymmetric. The Binding assertion will imply the type of key associated with this token.
953 Services MAY also include information in the sp:RequestSecurityTokenTemplate element to
954 explicitly define the expected key type. See [Appendix B](#) for details of the
955 sp:RequestSecurityTokenTemplate element.

956 5.4.3 X509Token Assertion

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

958 Syntax

```
959 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
960   (  
961     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
962     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
963   ) ?  
964   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```

965 <wsp:Policy xmlns:wsp="...">
966   (
967     <sp:RequireDerivedKeys ... /> |
968     <sp:RequireExplicitDerivedKeys ... /> |
969     <sp:RequireImpliedDerivedKeys ... />
970   ) ?
971   <sp:RequireKeyIdentifierReference ... /> ?
972   <sp:RequireIssuerSerialReference ... /> ?
973   <sp:RequireEmbeddedTokenReference ... /> ?
974   <sp:RequireThumbprintReference ... /> ?
975   (
976     <sp:WssX509V3Token10 ... /> |
977     <sp:WssX509Pkcs7Token10 ... /> |
978     <sp:WssX509PkiPathV1Token10 ... /> |
979     <sp:WssX509V1Token11 ... /> |
980     <sp:WssX509V3Token11 ... /> |
981     <sp:WssX509Pkcs7Token11 ... /> |
982     <sp:WssX509PkiPathV1Token11 ... />
983   ) ?
984   ...
985 </wsp:Policy>
986 ...
987 </sp:X509Token>

```

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

/sp:X509Token

This identifies an X509Token assertion.

/sp:X509Token/@sp:IncludeToken

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

/sp:X509Token/sp:Issuer

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

/sp:X509Token/sp:IssuerName

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

/sp:X509Token/wst:Claims

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

/sp:X509Token/wsp:Policy

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

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

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

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

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

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

1014 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1015 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1016 'false'.

1017 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

1018 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1019 REQUIRED when referencing this token.

1020 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

1021 This OPTIONAL element is a policy assertion that indicates that an issuer serial reference is
1022 REQUIRED when referencing this token.

1023 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

1024 This OPTIONAL element is a policy assertion that indicates that an embedded token reference is
1025 REQUIRED when referencing this token.

1026 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1027 This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is
1028 REQUIRED when referencing this token.

1029 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1030 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
1031 be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1032 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1033 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
1034 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1035 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1036 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
1037 token should be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1038 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1039 This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should
1040 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1041 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1042 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
1043 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1044 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1045 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
1046 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1047 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1048 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
1049 token should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1050 5.4.4 KerberosToken Assertion

1051 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

1052 Syntax

```
1053 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1054 (
1055   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1056   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1057 ) ?
```

```

1058 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1059 <wsp:Policy xmlns:wsp="...">
1060   (
1061     <sp:RequireDerivedKeys ... /> |
1062     <sp:RequireImpliedDerivedKeys ... /> |
1063     <sp:RequireExplicitDerivedKeys ... />
1064   ) ?
1065   <sp:RequireKeyIdentifierReference ... /> ?
1066   (
1067     <sp:WssKerberosV5ApReqToken11 ... /> |
1068     <sp:WssGssKerberosV5ApReqToken11 ... />
1069   ) ?
1070   ...
1071 </wsp:Policy>
1072 ...
1073 </sp:KerberosToken>

```

1075

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

1076 /sp:KerberosToken

This identifies a KerberosV5ApReqToken assertion.

1077 /sp:KerberosToken/@sp:IncludeToken

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

1080 /sp:KerberosToken/sp:Issuer

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

1081 /sp:KerberosToken/sp:IssuerName

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

1084 /sp:KerberosToken/wst:Claims

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

1087 /sp:KerberosToken/wsp:Policy

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

1090 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

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

1093 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1096 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1100 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1105 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1106 REQUIRED when referencing this token.

1107 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

1108 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ
 1109 token should be used as defined in [WSS:KerberosTokenProfile1.1].

1110 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1111 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-
 1112 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

1113 5.4.5 SpnegoContextToken Assertion

1114 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
 1115 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1116 Syntax

```

1117 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1118   (
1119     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1120     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1121   ) ?
1122   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1123   <wsp:Policy xmlns:wsp="...">
1124     (
1125       <sp:RequireDerivedKeys ... /> |
1126       <sp:RequireImpliedDerivedKeys ... /> |
1127       <sp:RequireExplicitDerivedKeys ... />
1128     ) ?
1129     <sp:MustNotSendCancel ... /> ?
1130     <sp:MustNotSendAmend ... /> ?
1131     <sp:MustNotSendRenew ... /> ?
1132     ...
1133   </wsp:Policy>
1134   ...
1135 </sp:SpnegoContextToken>
  
```

1136

1137 The following describes the attributes and elements listed in the schema outlined above:

1138 /sp:SpnegoContextToken

1139 This identifies a SpnegoContextToken assertion.

1140 /sp:SpnegoContextToken/@sp:IncludeToken

1141 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1142 /sp:SpnegoContextToken/sp:Issuer

1143 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
 1144 for the Spnego Context Token.

1145 /sp:SpnegoContextToken/sp:IssuerName

1146 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
 1147 sp:SpnegoContextToken issuer.

1148 /sp:SpnegoContextToken/wst:Claims

1149 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1150 order to satisfy the token assertion requirements.

1151 /sp:SpnegoContextToken/wsp:Policy

1152 This REQUIRED element identifies additional requirements for use of the
 1153 sp:SpnegoContextToken assertion.

1154 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1155 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1156 and [Implied Derived Keys] properties for this token to 'true'.

1157 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1158 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1159 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1160 'false'.

1161 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1162 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1163 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1164 'false'.

1165 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1166 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1167 token does not support SCT/Cancel RST messages. If this assertion is missing it means that
 1168 SCT/Cancel RST messages are supported by the STS.

1169 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1170 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1171 token does not support SCT/Amend RST messages. If this assertion is missing it means that
 1172 SCT/Amend RST messages are supported by the STS.

1173 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1174 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1175 token does not support SCT/Renew RST messages. If this assertion is missing it means that
 1176 SCT/Renew RST messages are supported by the STS.

1177 5.4.6 SecurityContextToken Assertion

1178 This element represents a requirement for a SecurityContextToken token.

1179 Syntax

```

1180 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1181 (
1182   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1183   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1184 ) ?
1185 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1186 <wsp:Policy xmlns:wsp="...">
1187   (
1188     <sp:RequireDerivedKeys ... /> |
1189     <sp:RequireImpliedDerivedKeys ... /> |
1190     <sp:RequireExplicitDerivedKeys ... />
1191   ) ?
1192   <sp:RequireExternalUriReference ... /> ?
1193   <sp:SC13SecurityContextToken... /> ?
1194   ...
1195 </wsp:Policy>
1196 ...
1197 </sp:SecurityContextToken>

```

1198

1199 The following describes the attributes and elements listed in the schema outlined above:

1200 /sp:SecurityContextToken

1201 This identifies a SecurityContextToken assertion.

1202 /sp:SecurityContextToken/@sp:IncludeToken

1203 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1204 /sp:SecurityContextToken/sp:Issuer

1205 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer

1206 of the sp:SecurityContextToken.

1207 /sp:SecurityContextToken/sp:IssuerName

1208 This OPTIONAL element, of type xs:anyURI, contains the logical name of the

1209 sp:SecurityContextToken issuer.

1210 /sp:SecurityContextToken/wst:Claims

1211 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in

1212 order to satisfy the token assertion requirements.

1213 /sp:SecurityContextToken/wsp:Policy

1214 This REQUIRED element identifies additional requirements for use of the

1215 sp:SecurityContextToken assertion.

1216 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1217 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]

1218 and [Implied Derived Keys] properties for this token to 'true'.

1219 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1220 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived

1221 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to

1222 'false'.

1223 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1224 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived

1225 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to

1226 'false'.

1227 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1228 This OPTIONAL element is a policy assertion that indicates that an external URI reference is

1229 REQUIRED when referencing this token.

1230 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1231 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should

1232 be used as defined in [\[WS-SecureConversation\]](#).

1233

1234 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that

1235 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If

1236 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the

1237 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

1238 5.4.7 SecureConversationToken Assertion

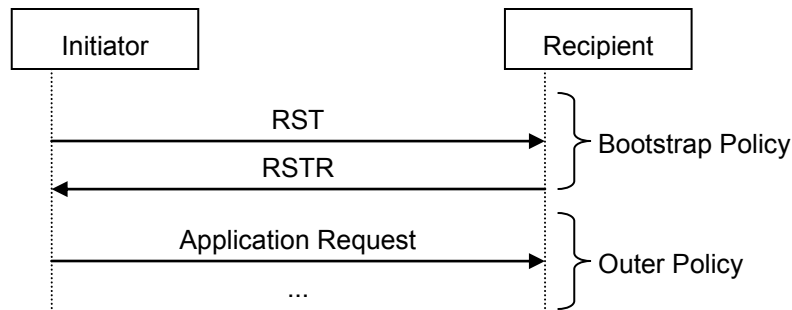
1239 This element represents a requirement for a Security Context Token retrieved from the indicated issuer

1240 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the

1241 service endpoint address.

1242

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



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

Amending Context

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

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

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

Renewing Context

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

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

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

Canceling Context

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

Handling Policy Alternatives

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

- The policy alternative used as a basis for the context renewal MUST be the same as the policy 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:

Policy

Policy-alternative-1

SecureConversationToken-assertion-1

Policy-alternative-2

SecureConversationToken-assertion-2

The policy alternative used as basis for context amend and cancel MUST be the same as the policy alternative that was used to obtain the context. This means that Policy-alternative-1 above cannot be 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:

Policy

SecureConversationToken-assertion-1

Policy-alternative-1

Policy-alternative-2

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

Syntax

```
<sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
(
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
  <sp:IssuerName>xs:anyURI</sp:IssuerName>
) ?
<wst:Claims Dialect="..."> ... </wst:Claims> ?
<wsp:Policy xmlns:wsp="...">
  (
    <sp:RequireDerivedKeys ... /> |
    <sp:RequireImpliedDerivedKeys ... /> |
    <sp:RequireExplicitDerivedKeys ... />
  ) ?
  <sp:RequireExternalUriReference ... /> ?
  <sp:SC13SecurityContextToken ... /> ?
  <sp:MustNotSendCancel ... /> ?
  <sp:MustNotSendAmend ... /> ?
  <sp:MustNotSendRenew ... /> ?
  <sp:BootstrapPolicy ... >
    <wsp:Policy> ... </wsp:Policy>
  </sp:BootstrapPolicy> ?
</wsp:Policy>
...
</sp:SecureConversationToken>
```

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

/sp:SecureConversationToken

This identifies a SecureConversationToken assertion.

/sp:SecureConversationToken/@sp:IncludeToken

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

1330 /sp:SecureConversationToken/sp:Issuer
 1331 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
 1332 for the Security Context Token.

1333 /sp:SecureConversationToken/sp:IssuerName
 1334 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
 1335 sp:SecureConversationToken issuer.

1336 /sp:SpnegoContextToken/wst:Claims
 1337 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1338 order to satisfy the token assertion requirements.

1339 /sp:SecureConversationToken/wsp:Policy
 1340 This REQUIRED element identifies additional requirements for use of the
 1341 sp:SecureConversationToken assertion.

1342 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys
 1343 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1344 and [Implied Derived Keys] properties for this token to 'true'.

1345 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1346 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1347 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1348 'false'.

1349 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1350 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1351 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1352 'false'.

1353 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference
 1354 This OPTIONAL element is a policy assertion that indicates that an external URI reference is
 1355 REQUIRED when referencing this token.

1356 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken
 1357 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
 1358 be used as obtained using the protocol defined in [\[WS-SecureConversation\]](#).

1359 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel
 1360 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
 1361 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it
 1362 means that SCT/Cancel RST messages are supported by the STS.

1363 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend
 1364 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
 1365 conversation token does not support SCT/Amend RST messages. If this assertion is missing it
 1366 means that SCT/Amend RST messages are supported by the STS.

1367 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew
 1368 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
 1369 conversation token does not support SCT/Renew RST messages. If this assertion is missing it
 1370 means that SCT/Renew RST messages are supported by the STS.

1371 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy
 1372 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements
 1373 for obtaining the Security Context Token.

1374 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1375 This element contains the security binding requirements for obtaining the Security Context Token.

1376 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with

1377 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that

1378 are to be protected.

1379 Example

```

1380 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1381   <sp:SymmetricBinding>
1382     <wsp:Policy>
1383       <sp:ProtectionToken>
1384         <wsp:Policy>
1385           <sp:SecureConversationToken>
1386             <sp:Issuer>
1387               <wsa:Address>http://example.org/sts</wsa:Address>
1388             </sp:Issuer>
1389             <wsp:Policy>
1390               <sp:SC13SecurityContextToken />
1391               <sp:BootstrapPolicy>
1392                 <wsp:Policy>
1393                   <sp:AsymmetricBinding>
1394                     <wsp:Policy>
1395                       <sp:InitiatorToken>
1396                         ...
1397                       </sp:InitiatorToken>
1398                       <sp:RecipientToken>
1399                         ...
1400                       </sp:RecipientToken>
1401                     </wsp:Policy>
1402                   </sp:AsymmetricBinding>
1403                   <sp:SignedParts>
1404                     ...
1405                   </sp:SignedParts>
1406                   ...
1407                 </wsp:Policy>
1408               </sp:BootstrapPolicy>
1409             </wsp:Policy>
1410           </sp:SecureConversationToken>
1411         </wsp:Policy>
1412       </sp:ProtectionToken>
1413       ...
1414     </wsp:Policy>
1415   </sp:SymmetricBinding>
1416   <sp:SignedParts>
1417     ...
1418   </sp:SignedParts>
1419   ...
1420 </wsp:Policy>

```

1421 5.4.8 SamlToken Assertion

1422 This element represents a requirement for a SAML token.

1423 Syntax

```

1424 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1425   (
1426     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1427     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1428   ) ?
1429   <wst:Claims Dialect="..."> ... </wst:Claims> ?

```

```

1430 <wsp:Policy xmlns:wsp="...">
1431 (
1432   <sp:RequireDerivedKeys ... /> |
1433   <sp:RequireImpliedDerivedKeys ... /> |
1434   <sp:RequireExplicitDerivedKeys ... />
1435 ) ?
1436 <sp:RequireKeyIdentifierReference ... /> ?
1437 (
1438   <sp:WssSamlV11Token10 ... /> |
1439   <sp:WssSamlV11Token11 ... /> |
1440   <sp:WssSamlV20Token11 ... />
1441 ) ?
1442 ...
1443 </wsp:Policy>
1444 ...
1445 </sp:SamlToken>

```

1446

1447 The following describes the attributes and elements listed in the schema outlined above:

1448 /sp:SamlToken

1449 This identifies a SamlToken assertion.

1450 /sp:SamlToken/@sp:IncludeToken

1451 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1452 /sp:SamlToken/sp:Issuer

1453 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1454 of the sp:SamlToken.

1455 /sp:SamlToken/sp:IssuerName

1456 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken
1457 issuer.

1458 /sp:SamlToken/wst:Claims

1459 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1460 order to satisfy the token assertion requirements.

1461 /sp:SamlToken/wsp:Policy

1462 This REQUIRED element identifies additional requirements for use of the sp:SamlToken
1463 assertion.

1464 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

1465 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1466 and [Implied Derived Keys] properties for this token to 'true'.

1467 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1468 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1469 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1470 'false'.

1471 /sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1472 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1473 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1474 'false'.

1475 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1476 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1477 REQUIRED when referencing this token.

1478 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10
1479 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1480 should be used as defined in [WSS:SAMLTOKENProfile1.0].

1481 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11
1482 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1483 should be used as defined in [WSS:SAMLTOKENProfile1.1].

1484 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11
1485 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token
1486 should be used as defined in [WSS:SAMLTOKENProfile1.1].

1487
1488 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1489 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1490 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion
1491 SHOULD be used instead.

1492 5.4.9 RelToken Assertion

1493 This element represents a requirement for a REL token.

1494 Syntax

```
1495 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1496   (  
1497     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1498     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1499   ) ?  
1500   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1501   <wsp:Policy xmlns:wsp="...">  
1502     (  
1503       <sp:RequireDerivedKeys ... /> |  
1504       <sp:RequireImpliedDerivedKeys ... /> |  
1505       <sp:RequireExplicitDerivedKeys ... />  
1506     ) ?  
1507     <sp:RequireKeyIdentifierReference ... /> ?  
1508     (  
1509       <sp:WssRelV10Token10 ... /> |  
1510       <sp:WssRelV20Token10 ... /> |  
1511       <sp:WssRelV10Token11 ... /> |  
1512       <sp:WssRelV20Token11 ... />  
1513     ) ?  
1514     ...  
1515   </wsp:Policy>  
1516   ...  
1517 </sp:RelToken>
```

1518
1519 The following describes the attributes and elements listed in the schema outlined above:

1520 /sp:RelToken

1521 This identifies a RelToken assertion.

1522 /sp:RelToken/@sp:IncludeToken

1523 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1524 /sp:RelToken/sp:Issuer

1525 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1526 of the sp:RelToken.

1527 /sp:RelToken/sp:IssuerName
 1528 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken
 1529 issuer.

1530 /sp:RelToken/wst:Claims
 1531 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1532 order to satisfy the token assertion requirements.

1533 /sp:RelToken/wsp:Policy
 1534 This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

1535 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys
 1536 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1537 and [Implied Derived Keys] property for this token to 'true'.

1538 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1539 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1540 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1541 'false'.

1542 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1543 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1544 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1545 'false'.

1546 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1547 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1548 REQUIRED when referencing this token.

1549 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10
 1550 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1551 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1552 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10
 1553 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1554 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1555 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11
 1556 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1557 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1558 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11
 1559 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1560 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1561
 1562 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
 1563 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
 1564 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion
 1565 SHOULD be used instead.

1566 5.4.10 HttpsToken Assertion

1567 This element represents a requirement for a transport binding to support the use of HTTPS.

1568 Syntax

```

1569 <sp:HttpsToken xmlns:sp="..." ... >
1570 (
1571   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1572   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1573 ) ?
1574 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1575 <wsp:Policy xmlns:wsp="...">
1576 (
1577   <sp:HttpBasicAuthentication /> |
1578   <sp:HttpDigestAuthentication /> |
1579   <sp:RequireClientCertificate /> |
1580   ...
1581 ) ?
1582   ...
1583 </wsp:Policy>
1584   ...
1585 </sp:HttpsToken>

```

1586 The following describes the attributes and elements listed in the schema outlined above:

1587 /sp:HttpsToken

1588 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1589 supported.

1590 /sp:HttpsToken/sp:Issuer

1591 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1592 of the sp:HttpsToken.

1593 /sp:HttpsToken/sp:IssuerName

1594 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken
1595 issuer.

1596 /sp:HttpsToken/wst:Claims

1597 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1598 order to satisfy the token assertion requirements.

1599 /sp:HttpsToken/wsp:Policy

1600 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken
1601 assertion.

1602 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1603 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic
1604 Authentication [[RFC2068](#)] to authenticate to the service.

1605 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1606 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP
1607 Digest Authentication [[RFC2068](#)] to authenticate to the service.

1608 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1609 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a
1610 certificate when negotiating the HTTPS session.

1611 5.4.11 KeyValueToken Assertion

1612 This element represents a requirement for a KeyValue token. The next section defines the KeyValue
1613 security token abstraction for purposes of this token assertion.
1614

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

Syntax

```
<sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:RsaKeyValue ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:KeyValueToken>
```

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

/sp:KeyValueToken

This identifies a RsaToken assertion.

/sp:KeyValueToken/@sp:IncludeToken

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

/sp:KeyValueToken/wsp:Policy

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

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

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

5.4.11.1 Key Value Token

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

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

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

```
<ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
  <ds:KeyValue>
    <ds:RSAKeyValue>
      <ds:Modulus>ds:CryptoBinary</ds:Modulus>
      <ds:Exponent>ds:CryptoBinary</ds:Exponent>
    </ds:RSAKeyValue>
  </ds:KeyValue>
</ds:KeyInfo>
```

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

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

```

1667     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1668   </Transforms>
1669   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1670   <DigestValue>...</DigestValue>
1671 </Reference>
1672 </SignedInfo>
1673 <SignatureValue>...</SignatureValue>
1674 <KeyInfo>
1675   <KeyValue>
1676     <RSAKeyValue>
1677       <Modulus>...</Modulus>
1678       <Exponent>...</Exponent>
1679     </RSAKeyValue>
1680   </KeyValue>
1681 </KeyInfo>
1682 </Signature>

```

1683
 1684 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no
 1685 identifier can be associated with the token, the KeyValue token cannot be referenced by using
 1686 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the KeyValue
 1687 token can be used whenever a security token can be used as illustrated on the following example:

```

1688 <t:RequestSecurityToken xmlns:t="...">
1689   <t:RequestType>...</t:RequestType>
1690   ...
1691   <t:UseKey>
1692     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1693       <KeyValue>
1694         <RSAKeyValue>
1695           <Modulus>...</Modulus>
1696           <Exponent>...</Exponent>
1697         </RSAKeyValue>
1698       </KeyValue>
1699     </KeyInfo>
1700   </t:UseKey>
1701 </t:RequestSecurityToken>

```

6 Security Binding Properties

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

6.1 [Algorithm Suite] Property

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

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

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

- [Sym Sig] Symmetric Key Signature
- [Asym Sig] Signature with an asymmetric key
- [Dig] Digest
- [Enc] Encryption
- [Sym KW] Symmetric Key Wrap
- [Asym KW] Asymmetric Key Wrap
- [Comp Key] Computed key
- [Enc KD] Encryption key derivation
- [Sig KD] Signature key derivation
- [Min SKL] Minimum symmetric key length
- [Max SKL] Maximum symmetric key length
- [Min AKL] Minimum asymmetric key length
- [Max AKL] Maximum asymmetric key length

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

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

Sha512 <http://www.w3.org/2001/04/xmlenc#sha512>
 Aes128 <http://www.w3.org/2001/04/xmlenc#aes128-cbc>
 Aes192 <http://www.w3.org/2001/04/xmlenc#aes192-cbc>
 Aes256 <http://www.w3.org/2001/04/xmlenc#aes256-cbc>
 TripleDes <http://www.w3.org/2001/04/xmlenc#tripledes-cbc>
 KwAes128 <http://www.w3.org/2001/04/xmlenc#kw-aes128>
 KwAes192 <http://www.w3.org/2001/04/xmlenc#kw-aes192>
 KwAes256 <http://www.w3.org/2001/04/xmlenc#kw-aes256>
 KwTripleDes <http://www.w3.org/2001/04/xmlenc#kw-tripledes>
 KwRsaOaep <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>
 KwRsa15 http://www.w3.org/2001/04/xmlenc#rsa-1_5
 PSha1 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L128 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L192 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L256 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 XPath <http://www.w3.org/TR/1999/REC-xpath-19991116>
 XPath20 <http://www.w3.org/2002/06/xmldsig-filter2>
 C14n <http://www.w3.org/2001/10/xml-c14n#>
 ExC14n <http://www.w3.org/2001/10/xml-exc-c14n#>
 SNT <http://www.w3.org/TR/soap12-n11n>
 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>

1738

1739 The tables below show all the base algorithm suites defined by this specification. This table defines
 1740 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

1741 This table defines additional properties whose values can be specified along with the default value for that
 1742 property.

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

1743 This table defines values for the remaining components for each algorithm suite.

Algorithm Suite	[Dig]	[Enc]	[Sym KW]	[Asym KW]	[Enc KD]	[Sig KD]	[Min SKL]
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

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

6.2 [Timestamp] Property

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

6.3 [Protection Order] Property

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

EncryptBeforeSigning	Signature MUST 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.

The default value for this property is 'SignBeforeEncrypting'.

6.4 [Signature Protection] Property

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

6.5 [Token Protection] Property

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

6.6 [Entire Header and Body Signatures] Property

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

6.7 [Security Header Layout] Property

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

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

6.7.1 Strict Layout Rules for WSS 1.0

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

- 1795 b. A Username token (usually in encrypted form) MUST occur before the signature that
1796 signs it.
- 1797 c. A primary signature MUST occur before the supporting token signature that signs the
1798 primary signature's signature value element.
- 1799 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1800 has the same order requirements as the source plain text element, unless requirement 4
1801 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1802 supporting token signature per 2.c above and an encrypted token has the same ordering
1803 requirements as the unencrypted token.
- 1804 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1805 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1806 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1807 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1808 Layout Rules for WSS 1.1
- 1809 1. Tokens that are included in the message MUST be declared before use. For example:
- 1810 a. A local signing token MUST occur before the signature that uses it.
- 1811 b. A local token serving as the source token for a derived key token MUST occur before that
1812 derived key token.
- 1813 c. A local encryption token MUST occur before the reference list that points to
1814 xenc:EncryptedData elements that use it.
- 1815 d. If the same token is used for both signing and encryption, then it SHOULD appear before
1816 the ds:Signature and xenc:ReferenceList elements in the security header that are
1817 generated using the token.
- 1818 2. Signed elements inside the security header MUST occur before the signature that signs them.
1819 For example:
- 1820 a. A timestamp MUST occur before the signature that signs it.
- 1821 b. A Username token (usually in encrypted form) MUST occur before the signature that
1822 signs it.
- 1823 c. A primary signature MUST occur before the supporting token signature that signs the
1824 primary signature's signature value element.
- 1825 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1826 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1827 has the same order requirements as the source plain text element, unless requirement 4
1828 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1829 supporting token signature per 2.c above and an encrypted token has the same ordering
1830 requirements as the unencrypted token.
- 1831 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1832 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1833 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1834 However, the xenc:ReferenceList is NOT REQUIRED to appear before independently encrypted
1835 tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1836 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1837 1.1] MUST obey rule 1 above.

7 Security Binding Assertions

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

7.1 AlgorithmSuite Assertion

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

Syntax

```
<sp:AlgorithmSuite xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    (<sp:Basic256 ... /> |
    <sp:Basic192 ... /> |
    <sp:Basic128 ... /> |
    <sp:TripleDes ... /> |
    <sp:Basic256Rsa15 ... /> |
    <sp:Basic192Rsa15 ... /> |
    <sp:Basic128Rsa15 ... /> |
    <sp:TripleDesRsa15 ... /> |
    <sp:Basic256Sha256 ... /> |
    <sp:Basic192Sha256 ... /> |
    <sp:Basic128Sha256 ... /> |
    <sp:TripleDesSha256 ... /> |
    <sp:Basic256Sha256Rsa15 ... /> |
    <sp:Basic192Sha256Rsa15 ... /> |
    <sp:Basic128Sha256Rsa15 ... /> |
    <sp:TripleDesSha256Rsa15 ... /> |
    ...)
    <sp:InclusiveC14N ... /> ?
    <sp:SOAPNormalization10 ... /> ?
    <sp:STRTransform10 ... /> ?
    (<sp:XPath10 ... /> |
    <sp:XPathFilter20 ... /> |
    <sp:AbsXPath ... /> |
    ...) ?
    ...
  </wsp:Policy>
  ...
</sp:AlgorithmSuite>
```

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

/sp:AlgorithmSuite

This identifies an AlgorithmSuite assertion.

/sp:AlgorithmSuite/wsp:Policy

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

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

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

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

1887 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1888 set to 'Basic192'.

1889 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1890 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1891 set to 'Basic128'.

1892 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1893 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1894 set to 'TripleDes'.

1895 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1896 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1897 set to 'Basic256Rsa15'.

1898 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1899 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1900 set to 'Basic192Rsa15'.

1901 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1902 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1903 set to 'Basic128Rsa15'.

1904 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1905 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1906 set to 'TripleDesRsa15'.

1907 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1908 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1909 set to 'Basic256Sha256'.

1910 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1911 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1912 set to 'Basic192Sha256'.

1913 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1914 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1915 set to 'Basic128Sha256'.

1916 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1917 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1918 set to 'TripleDesSha256'.

1919 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1920 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1921 set to 'Basic256Sha256Rsa15'.

1922 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1923 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1924 set to 'Basic192Sha256Rsa15'.

1925 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1926 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1927 set to 'Basic128Sha256Rsa15'.

1928 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1929 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
 1930 set to 'TripleDesSha256Rsa15'.

1931 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1932 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an
 1933 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]
 1934 property is 'ExcC14N'.

1935 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1936 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set
 1937 to 'SNT'.

1938 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1939 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is
 1940 set to 'STRT10'.

1941 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1942 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
 1943 'XPath'.

1944 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1945 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
 1946 'XPath20'.

1947 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1948 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
 1949 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1950

1951 7.2 Layout Assertion

1952 This assertion indicates a requirement for a particular security header layout as defined under the
 1953 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
 1954 containing assertion.

1955 Syntax

```

1956 <sp:Layout xmlns:sp="..." ... >
1957   <wsp:Policy xmlns:wsp="...">
1958     <sp:Strict ... /> |
1959     <sp:Lax ... /> |
1960     <sp:LaxTsFirst ... /> |
1961     <sp:LaxTsLast ... /> |
1962     ...
1963   </wsp:Policy>
1964   ...
1965 </sp:Layout>

```

1966

1967 The following describes the attributes and elements listed in the schema outlined above:

1968 /sp:Layout

1969 This identifies a Layout assertion.

1970 /sp:Layout/wsp:Policy

1971 This REQUIRED element contains one or more policy assertions that indicate the specific security
 1972 header layout to use.

1973 /sp:Layout/wsp:Policy/sp:Strict

1974 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
 1975 property is set to 'Strict'.
 1976 /sp:Layout/wsp:Policy/sp:Lax
 1977 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
 1978 property is set to 'Lax'.
 1979 /sp:Layout/wsp:Policy/sp:LaxTsFirst
 1980 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
 1981 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
 1982 'true' by the presence of an sp:IncludeTimestamp assertion.
 1983 /sp:Layout/wsp:Policy/sp:LaxTsLast
 1984 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
 1985 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
 1986 'true' by the presence of an sp:IncludeTimestamp assertion.

1987 7.3 TransportBinding Assertion

1988 The TransportBinding assertion is used in scenarios in which message protection and security correlation
 1989 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
 1990 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
 1991 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
 1992 MUST apply to [Endpoint Policy Subject].

1993 Syntax

```

1994 <sp:TransportBinding xmlns:sp="..." ... >
1995   <wsp:Policy xmlns:wsp="...">
1996     <sp:TransportToken ... >
1997       <wsp:Policy> ... </wsp:Policy>
1998       ...
1999     </sp:TransportToken>
2000     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2001     <sp:Layout ... > ... </sp:Layout> ?
2002     <sp:IncludeTimestamp ... /> ?
2003     ...
2004   </wsp:Policy>
2005   ...
2006 </sp:TransportBinding>
  
```

2007
 2008 The following describes the attributes and elements listed in the schema outlined above:
 2009 /sp:TransportBinding
 2010 This identifies a TransportBinding assertion.
 2011 /sp:TransportBinding/wsp:Policy
 2012 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
 2013 assertion.
 2014 /sp:TransportBinding/wsp:Policy/sp:TransportToken
 2015 This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.
 2016 The specified token populates the [Transport Token] property and indicates how the transport is
 2017 secured.
 2018 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy
 2019 This indicates a nested policy that identifies the type of Transport Token to use.

2020 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite
 2021 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
 2022 Suite] property. See Section 6.1 for more details.

2023 /sp:TransportBinding/wsp:Policy/sp:Layout
 2024 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
 2025 Header Layout] property. See Section 6.7 for more details.

2026 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp
 2027 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
 2028 to 'true'.

2029 7.4 SymmetricBinding Assertion

2030 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
 2031 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
 2032 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
 2033 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
 2034 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
 2035 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
 2036 properties and is used as the basis for both encryption and signature in both directions. This assertion
 2037 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

2038 Syntax

```

2039 <sp:SymmetricBinding xmlns:sp="..." ... >
2040   <wsp:Policy xmlns:wsp="...">
2041     (
2042       <sp:EncryptionToken ... >
2043         <wsp:Policy> ... </wsp:Policy>
2044       </sp:EncryptionToken>
2045       <sp:SignatureToken ... >
2046         <wsp:Policy> ... </wsp:Policy>
2047       </sp:SignatureToken>
2048     ) | (
2049       <sp:ProtectionToken ... >
2050         <wsp:Policy> ... </wsp:Policy>
2051       </sp:ProtectionToken>
2052     )
2053   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2054   <sp:Layout ... > ... </sp:Layout> ?
2055   <sp:IncludeTimestamp ... /> ?
2056   <sp:EncryptBeforeSigning ... /> ?
2057   <sp:EncryptSignature ... /> ?
2058   <sp:ProtectTokens ... /> ?
2059   <sp:OnlySignEntireHeadersAndBody ... /> ?
2060   ...
2061 </wsp:Policy>
2062   ...
2063 </sp:SymmetricBinding>
  
```

2064
 2065 The following describes the attributes and elements listed in the schema outlined above:

2066 /sp:SymmetricBinding

2067 This identifies a SymmetricBinding assertion.

2068 /sp:SymmetricBinding/wsp:Policy

2069 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
 2070 assertion.

2071 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
 2072 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption
 2073 Token. The specified token populates the [Encryption Token] property and is used for encryption.
 2074 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

2075 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
 2076 The policy contained here MUST identify exactly one token to use for encryption.

2077 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
 2078 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.
 2079 The specified token populates the [Signature Token] property and is used for the message
 2080 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
 2081 specified.

2082 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
 2083 The policy contained here MUST identify exactly one token to use for signatures.

2084 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
 2085 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.
 2086 The specified token populates the [Encryption Token] and [Signature Token properties] and is
 2087 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
 2088 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
 2089 specified.

2090 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
 2091 The policy contained here MUST identify exactly one token to use for protection.

2092 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2093 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
 2094 Suite] property. See Section 6.1 for more details.

2095 /sp:SymmetricBinding/wsp:Policy/sp:Layout
 2096 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
 2097 Header Layout] property. See Section 6.7 for more details.

2098 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 2099 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
 2100 to 'true'.

2101 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 2102 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
 2103 set to 'EncryptBeforeSigning'.

2104 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
 2105 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
 2106 property is set to 'true'.

2107 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
 2108 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
 2109 set to 'true'.

2110 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 2111 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
 2112 Signatures] property is set to 'true'.

7.5 AsymmetricBinding Assertion

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

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

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

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

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

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

Syntax

```
<sp:AsymmetricBinding xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:InitiatorToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorToken>
    ) | (
      <sp:InitiatorSignatureToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorSignatureToken>
      <sp:InitiatorEncryptionToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorEncryptionToken>
    )
    (
      <sp:RecipientToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientToken>
    ) | (
      <sp:RecipientSignatureToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientSignatureToken>
      <sp:RecipientEncryptionToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientEncryptionToken>
    )
  </wsp:Policy>
</sp:AsymmetricBinding>
```

```

2165     </sp:RecipientEncryptionToken>
2166   )
2167   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2168   <sp:Layout ... > ... </sp:Layout> ?
2169   <sp:IncludeTimestamp ... /> ?
2170   <sp:EncryptBeforeSigning ... /> ?
2171   <sp:EncryptSignature ... /> ?
2172   <sp:ProtectTokens ... /> ?
2173   <sp:OnlySignEntireHeadersAndBody ... /> ?
2174   ...
2175 </wsp:Policy>
2176   ...
2177 </sp:AsymmetricBinding>

```

2178

2179 The following describes the attributes and elements listed in the schema outlined above:

2180 /sp:AsymmetricBinding

2181 This identifies a AsymmetricBinding assertion.

2182 /sp:AsymmetricBinding/wsp:Policy

2183 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
2184 assertion.

2185 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

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

2190 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

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

2192 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

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

2196 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

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

2198 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

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

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

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

2204 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2205 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.
2206 The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
2207 property and is used for encryption from initiator to recipient, and for the message signature from
2208 recipient to initiator.

2209 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2210 The policy contained here MUST identify one or more token assertions.

2211 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2212 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
 2213 Signature Token. The specified token populates the [Recipient Signature Token] property and is
 2214 used for the message signature from recipient to initiator.

2215 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
 2216 The policy contained here MUST identify one or more token assertions.

2217 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
 2218 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
 2219 Encryption Token. The specified token populates the [Recipient Encryption Token] property and
 2220 is used for the message encryption from initiator to recipient.

2221 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
 2222 The policy contained here MUST identify one or more token assertions.

2223 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2224 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
 2225 Suite] property. See Section 6.1 for more details.

2226 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
 2227 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
 2228 Header Layout] property. See Section 6.7 for more details.

2229 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 2230 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
 2231 to 'true'.

2232 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 2233 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
 2234 set to 'EncryptBeforeSigning'.

2235 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
 2236 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
 2237 property is set to 'true'.

2238 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
 2239 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
 2240 set to 'true'.

2241 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 2242 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
 2243 Signatures] property is set to 'true'.

8 Supporting Tokens

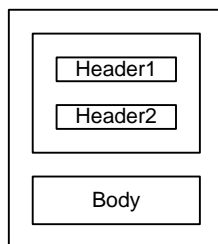
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. In case of Transport Binding the message is signed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

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

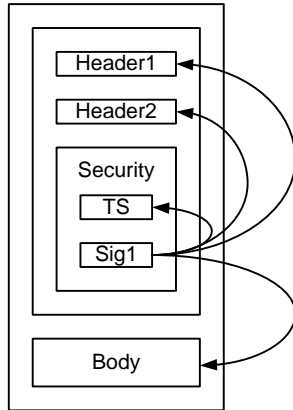
In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine which order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

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

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

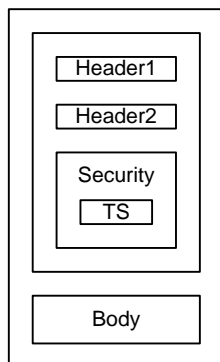


Even before any supporting tokens are added, each binding requires that the message is signed using a token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection. The signature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



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

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



8.1 SupportingTokens Assertion

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

Syntax

```
<sp:SupportingTokens xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    [Token Assertion]+
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
    (
      <sp:SignedParts ... > ... </sp:SignedParts> |
```

2308
2309
2310
2311
2312
2313
2314
2315

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

2316

2317 The following describes the attributes and elements listed in the schema outlined above:

2318 /sp:SupportingTokens

2319 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2320 Tokens] property.

2321 /sp:SupportingTokens/wsp:Policy

2322 This describes additional requirements for satisfying the SupportingTokens assertion.

2323 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2324 The policy MUST identify one or more token assertions.

2325 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2326 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2327 describes the algorithms to use for cryptographic operations performed with the tokens identified
2328 by this policy assertion.

2329 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2330 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2331 and describes additional message parts that MUST be included in the signature generated with
2332 the token identified by this policy assertion.

2333 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2334 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2335 and describes additional message elements that MUST be included in the signature generated
2336 with the token identified by this policy assertion.

2337 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2338 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2339 and describes additional message parts that MUST be encrypted using the token identified by
2340 this policy assertion.

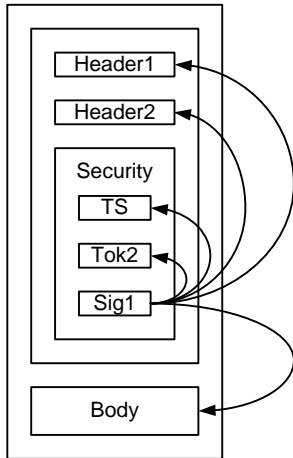
2341 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2342 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2343 and describes additional message elements that MUST be encrypted using the token identified
2344 by this policy assertion.

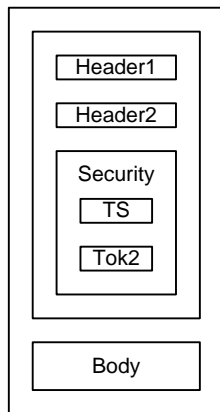
2345 8.2 SignedSupportingTokens Assertion

2346 Signed tokens are included in the “message signature” as defined above and MAY OPTIONALLY include
2347 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
2348 (Tok2) is signed by the message signature (Sig1):

2349



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



Syntax

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

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

/sp:SignedSupportingTokens

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

/sp:SignedSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the SignedSupportingTokens assertion.

2376 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 2377 The policy MUST identify one or more token assertions.

2378 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 2379 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
 2380 describes the algorithms to use for cryptographic operations performed with the tokens identified
 2381 by this policy assertion.

2382 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 2383 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
 2384 and describes additional message parts that MUST be included in the signature generated with
 2385 the token identified by this policy assertion.

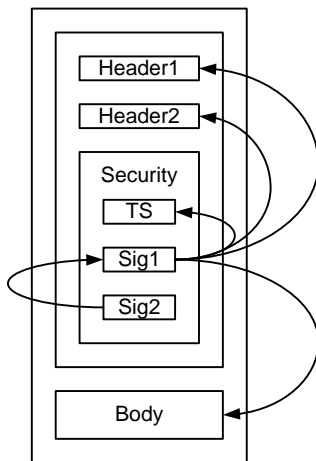
2386 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 2387 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
 2388 and describes additional message elements that MUST be included in the signature generated
 2389 with the token identified by this policy assertion.

2390 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 2391 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
 2392 and describes additional message parts that MUST be encrypted using the token identified by
 2393 this policy assertion.

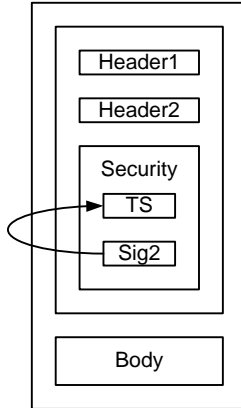
2394 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 2395 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
 2396 and describes additional message elements that MUST be encrypted using the token identified
 2397 by this policy assertion.

2398 8.3 EndorsingSupportingTokens Assertion

2399 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 2400 produced from the message signature and MAY OPTIONALLY include additional message parts to sign
 2401 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message
 2402 signature (Sig1):
 2403



2404
 2405 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 2406 below:
 2407



2408

2409 Syntax

```

2410 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2411   <wsp:Policy xmlns:wsp="...">
2412     [Token Assertion]+
2413     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2414     (
2415       <sp:SignedParts ... > ... </sp:SignedParts> |
2416       <sp:SignedElements ... > ... </sp:SignedElements> |
2417       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2418       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2419     ) *
2420     ...
2421   </wsp:Policy>
2422   ...
2423 </sp:EndorsingSupportingTokens>
  
```

2424

2425 The following describes the attributes and elements listed in the schema outlined above:

2426 /sp:EndorsingSupportingTokens

2427 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
2428 [Endorsing Supporting Tokens] property.

2429 /sp:EndorsingSupportingTokens/wsp:Policy

2430 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2431 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2432 The policy MUST identify one or more token assertions.

2433 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2434 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2435 describes the algorithms to use for cryptographic operations performed with the tokens identified
2436 by this policy assertion.

2437 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2438 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2439 and describes additional message parts that MUST be included in the signature generated with
2440 the token identified by this policy assertion.

2441 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

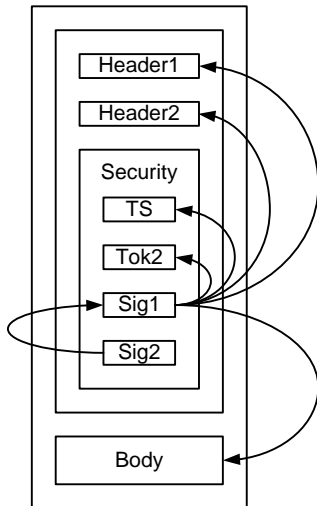
2442 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2443 and describes additional message elements that MUST be included in the signature generated
2444 with the token identified by this policy assertion.

2445 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
 2446 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
 2447 and describes additional message parts that MUST be encrypted using the token identified by
 2448 this policy assertion.

2449 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
 2450 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
 2451 and describes additional message elements that MUST be encrypted using the token identified
 2452 by this policy assertion.

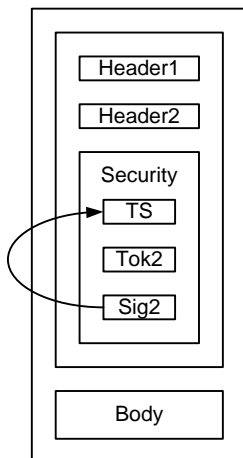
2453 8.4 SignedEndorsingSupportingTokens Assertion

2454 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
 2455 and are themselves signed by that message signature, that is both tokens (the token used for the
 2456 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY
 2457 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
 2458 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
 2459 message signature (Sig1):
 2460



2461

2462 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
 2463 SHOULD cover the message timestamp as illustrated below:
 2464



2465

2466 Syntax

```
2467 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2468   <wsp:Policy xmlns:wsp="...">
2469     [Token Assertion]+
2470     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2471     (
2472       <sp:SignedParts ... > ... </sp:SignedParts> |
2473       <sp:SignedElements ... > ... </sp:SignedElements> |
2474       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2475       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2476     ) *
2477     ...
2478   </wsp:Policy>
2479   ...
2480 </sp:SignedEndorsingSupportingTokens>
```

2481

2482 The following describes the attributes and elements listed in the schema outlined above:

2483 /sp:SignedEndorsingSupportingTokens

2484 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the
2485 [Signed Endorsing Supporting Tokens] property.

2486 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2487 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2488 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2489 The policy MUST identify one or more token assertions.

2490 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2491 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2492 describes the algorithms to use for cryptographic operations performed with the tokens identified
2493 by this policy assertion.

2494 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2495 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2496 and describes additional message parts that MUST be included in the signature generated with
2497 the token identified by this policy assertion.

2498 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2499 This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional
2500 message elements that MUST be included in the signature generated with the token identified by
2501 this policy assertion.

2502 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2503 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2504 and describes additional message parts that MUST be encrypted using the token identified by
2505 this policy assertion.

2506 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2507 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2508 and describes additional message elements that MUST be encrypted using the token identified
2509 by this policy assertion.

8.5 SignedEncryptedSupportingTokens Assertion

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

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

8.6 EncryptedSupportingTokens Assertion

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

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

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

8.7 EndorsingEncryptedSupportingTokens Assertion

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

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

8.8 SignedEndorsingEncryptedSupportingTokens Assertion

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

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

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

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

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

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

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

8.10 Example

Example policy containing supporting token assertions:

```
<!-- Example Endpoint Policy -->
<wsp:Policy xmlns:wsp="...">
  <sp:SymmetricBinding xmlns:sp="...">
    <wsp:Policy>
      <sp:ProtectionToken>
        <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
          <sp:Issuer>...</sp:Issuer>
          <sp:RequestSecurityTokenTemplate>
            ...
          </sp:RequestSecurityTokenTemplate>
        </sp:IssuedToken>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      ...
    </wsp:Policy>
  </sp:SymmetricBinding>
  ...
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  ...
</wsp:Policy>
```

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

9 WSS: SOAP Message Security Options

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

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

Note: This approach is chosen because:

- A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used in a single reference.
- B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending on which of a series of messages is being secured.

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

WSS: SOAP Message Security 1.0 Properties

[Direct References]

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

[Key Identifier References]

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

[Issuer Serial References]

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

[External URI References]

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

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

[Embedded Token References]

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

WSS: SOAP Message Security 1.1 Properties

[Thumbprint References]

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

[EncryptedKey References]

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

[Signature Confirmation]

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

9.1 Wss10 Assertion

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

Syntax

```
<sp:Wss10 xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:MustSupportRefKeyIdentifier ... /> ?
    <sp:MustSupportRefIssuerSerial ... /> ?
    <sp:MustSupportRefExternalURI ... /> ?
    <sp:MustSupportRefEmbeddedToken ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:Wss10>
```

2690 The following describes the attributes and elements listed in the schema outlined above:

2691 /sp:Wss10

2692 This identifies a WSS10 assertion.

2693 /sp:Wss10/wsp:Policy

2694 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2695 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier

2696 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]

2697 property is set to 'true'.

2698 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial

2699 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]

2700 property is set to 'true'.

2701 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI

2702 This OPTIONAL element is a policy assertion indicates that the [External URI References]

2703 property is set to 'true'.

2704 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken

2705 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]

2706 property is set to 'true'.

2707 9.2 Wss11 Assertion

2708 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are

2709 supported.

2710 Syntax

```

2711 <sp:Wss11 xmlns:sp="..." ... >
2712   <wsp:Policy xmlns:wsp="...">
2713     <sp:MustSupportRefKeyIdentifier ... /> ?
2714     <sp:MustSupportRefIssuerSerial ... /> ?
2715     <sp:MustSupportRefExternalURI ... /> ?
2716     <sp:MustSupportRefEmbeddedToken ... /> ?
2717     <sp:MustSupportRefThumbprint ... /> ?
2718     <sp:MustSupportRefEncryptedKey ... /> ?
2719     <sp:RequireSignatureConfirmation ... /> ?
2720     ...
2721   </wsp:Policy>
2722 </sp:Wss11>

```

2723

2724 The following describes the attributes and elements listed in the schema outlined above:

2725 /sp:Wss11

2726 This identifies an WSS11 assertion.

2727 /sp:Wss11/wsp:Policy

2728 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2729 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier

2730 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]

2731 property is set to 'true'.

2732 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial

2733 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]

2734 property is set to 'true'.

2735 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2736 This OPTIONAL element is a policy assertion indicates that the [External URI References]
2737 property is set to 'true'.
2738 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2739 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2740 property is set to 'true'.
2741 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2742 This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property
2743 is set to 'true'.
2744 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2745 This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]
2746 property is set to 'true'.
2747 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2748 This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property
2749 is set to 'true'.

10 WS-Trust Options

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

WS-Trust Properties

[Client Challenge]

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

[Server Challenge]

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

[Client Entropy]

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

[Server Entropy]

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

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

[Issued Tokens]

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

[Collection]

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

[Scope Policy 1.5]

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

[Interactive Challenge]

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

10.1 Trust13 Assertion

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

Syntax

```
<sp:Trust13 xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:MustSupportClientChallenge ... />?
    <sp:MustSupportServerChallenge ... />?
    <sp:RequireClientEntropy ... />?
    <sp:RequireServerEntropy ... />?
    <sp:MustSupportIssuedTokens ... />?
    <sp:RequireRequestSecurityTokenCollection />?
    <sp:RequireAppliesTo />?
    <sp13:ScopePolicy15 />?
    <sp13:MustSupportInteractiveChallenge />?
    ...
  </wsp:Policy>
  ...
</sp:Trust13 ... >
```

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

/sp:Trust13

This identifies a Trust13 assertion.

/sp:Trust13/wsp:Policy

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

2840 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge
2841 This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set
2842 to 'true'.

2843 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge
2844 This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set
2845 to 'true'.

2846 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy
2847 This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to
2848 'true'.

2849 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy
2850 This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to
2851 'true'.

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

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

2858 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo
2859 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor
2860 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2861 /sp:Trust13/wsp:Policy/sp13:ScopePolicy15
2862 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5]
2863 property is set to 'true'.

2864 /sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge
2865 This optional element is a policy assertion indicates that the [Interactive Challenge]
2866 property is set to 'true'.

11 Guidance on creating new assertions and assertion extensibility

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

11.1 General Design Points

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

11.2 Detailed Design Guidance

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

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

```
Design 1
<A1/>
<A2/>
<A3/>

Design 2.
<A Parameter='1' />
<A Parameter='2' />
<A Parameter='3' />
```

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

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

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

2911 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2912 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2913 was encoded into the assertion Qnames, each existing token assertion would require five variants, one
2914 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2915

2916 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2917 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2918 assertion is parameterized by the nested token version assertions. Policy matching can use these
2919 parameters to find matches between policies where the broad token type is support by both parties but
2920 they might not support the same specific versions.

2921

2922 Note, when designing assertions for new token types such assertions SHOULD allow the
2923 sp:IncludeToken attribute and SHOULD allow nested policy.

2924

12 Security Considerations

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

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

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

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

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

13 Conformance

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

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

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

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

A. Assertions and WS-PolicyAttachment

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

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

TransportBinding Assertion	(Section 7.3)
SymmetricBinding Assertion	(Section 7.4)
AsymmetricBinding Assertion	(Section 7.5)

A.1.2 Token Assertions

SupportingTokens Assertion	(Section 8.1)
SignedSupportingTokens Assertion	(Section 8.2)
EndorsingSupportingTokens Assertion	(Section 8.3)
SignedEndorsingSupportingTokens Assertion	(Section 8.4)
SignedEncryptedSupportingTokens Assertion	(Section 8.5)
EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

A.1.3 WSS: SOAP Message Security 1.0 Assertions

Wss10 Assertion	(Section 9.1)
-----------------	---------------

A.1.4 WSS: SOAP Message Security 1.1 Assertions

Wss11 Assertion	(Section 9.2)
-----------------	---------------

A.1.5 Trust 1.0 Assertions

Trust13 Assertion	(Section 10.1)
-------------------	----------------

A.2 Operation Policy Subject Assertions

A.2.1 Security Binding Assertions

SymmetricBinding Assertion	(Section 7.4)
AsymmetricBinding Assertion	(Section 7.5)

A.2.2 Supporting Token Assertions

SupportingTokens Assertion	(Section 8.1)
SignedSupportingTokens Assertion	(Section 8.2)

2998	EndorsingSupportingTokens Assertion	(Section 8.3)
2999	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3000	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3001	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3002	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3003 **A.3 Message Policy Subject Assertions**

3004 **A.3.1 Supporting Token Assertions**

3005	SupportingTokens Assertion	(Section 8.1)
3006	SignedSupportingTokens Assertion	(Section 8.2)
3007	EndorsingSupportingTokens Assertion	(Section 8.3)
3008	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3009	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3010	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3011	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3012 **A.3.2 Protection Assertions**

3013	SignedParts Assertion	(Section 4.1.1)
3014	SignedElements Assertion	(Section 4.1.2)
3015	EncryptedParts Assertion	(Section 4.2.1)
3016	EncryptedElements Assertion	(Section 4.2.2)
3017	ContentEncryptedElements Assertion	(Section 4.2.3)
3018	RequiredElements Assertion	(Section 4.3.1)
3019	RequiredParts Assertion	(Section 4.3.2)

3020 **A.4 Assertions With Undefined Policy Subject**

3021 The assertions listed in this section do not have a defined policy subject because they appear nested
3022 inside some other assertion which does have a defined policy subject. This list is derived from nested
3023 assertions in the specification that have independent sections. It is not a complete list of nested
3024 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
3025 additional nested assertions.

3026 **A.4.1 General Assertions**

3027	AlgorithmSuite Assertion	(Section 7.1)
3028	Layout Assertion	(Section 7.2)

3029 **A.4.2 Token Usage Assertions**

3030 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
3031 assertions.

3032 **A.4.3 Token Assertions**

3033	UsernameToken Assertion	(Section 5.3.1)
------	---	-----------------

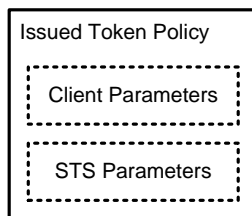
3034	IssuedToken Assertion	(Section 5.3.2)
3035	X509Token Assertion	(Section 5.3.3)
3036	KerberosToken Assertion	(Section 5.3.4)
3037	SpnegoContextToken Assertion	(Section 5.3.5)
3038	SecurityContextToken Assertion	(Section 5.3.6)
3039	SecureConversationToken Assertion	(Section 5.3.7)
3040	SamlToken Assertion	(Section 5.3.8)
3041	RelToken Assertion	(Section 5.3.9)
3042	HttpsToken Assertion	(Section 5.3.10)

B. Issued Token Policy

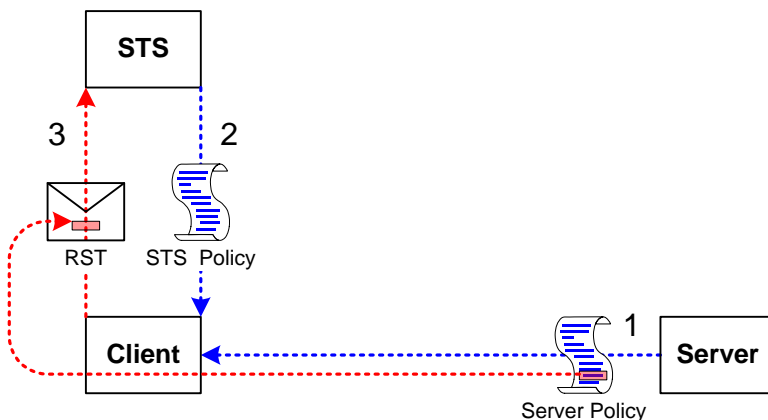
The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

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

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



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



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

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

3070 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The
3071 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along
3072 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
3073 request sent by the Client to the STS following the protocol defined in WS-Trust.

3074

3075 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)
3076 [Trust\]](#). All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship
3077 which specifies some or all of the conditions and constraints for issued tokens.

C. Strict Security Header Layout Examples

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

C.1 Transport Binding

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

C.1.1 Policy

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

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
    </wsp:Policy>
  </sp:TransportBinding>
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <sp:RequireSignatureConfirmation />
  </sp:Wss11>
</wsp:Policy>
```

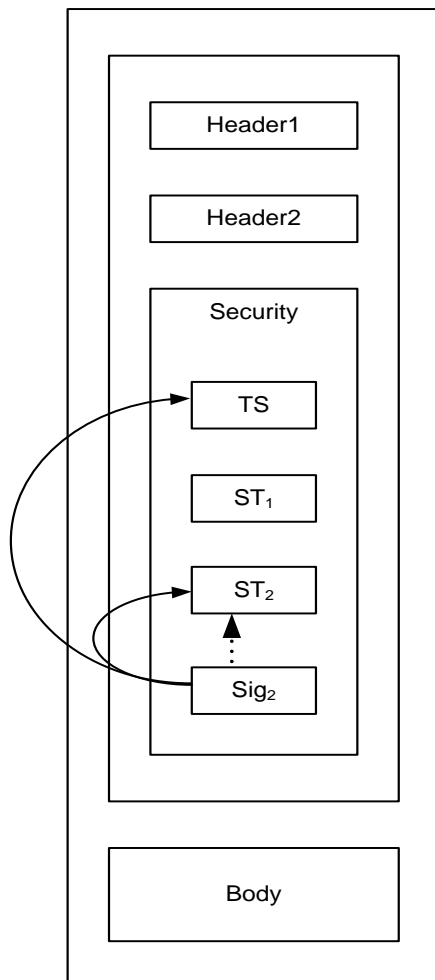
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.1.2 Initiator to Recipient Messages

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

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

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



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

Example:

Initiator to recipient message

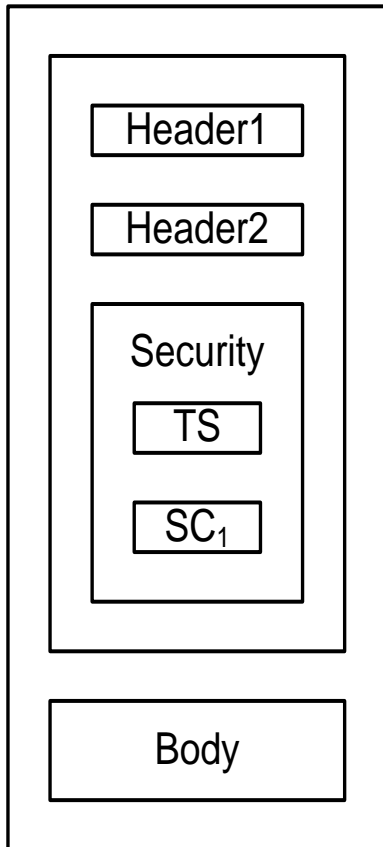
```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id='SomeSignedToken' >
        ...
      </wsse:UsernameToken>
      <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
        ...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:References>
            <ds:Reference URI="#timestamp" />
            <ds:Reference URI="#SomeSignedEndorsingToken" />
          </ds:References>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SomeSignedEndorsingToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      ...
    </wsse:Security>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

C.1.3 Recipient to Initiator Messages

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

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

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



3201

3202 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 3203 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 3204 message illustrated previously is included. In general, the ordering of the items in the security header
 3205 follows the most optimal layout for a receiver to process its contents.

3206 *Example:*

3207 Recipient to initiator message

```

3208 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3209   <S:Header>
3210     ...
3211     <wsse:Security>
3212       <wsu:Timestamp wsu:Id="timestamp">
3213         <wsu:Created>[datetime]</wsu:Created>
3214         <wsu:Expires>[datetime]</wsu:Expires>
3215       </wsu:Timestamp>
3216       <wsse11:SignatureConfirmation Value="..." />
3217       ...
3218     </wsse:Security>
3219     ...
3220   </S:Header>
3221   <S:Body>
3222     ...
3223   </S:Body>
3224 </S:Envelope>

```

3225 C.2 Symmetric Binding

3226 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

C.2.1 Policy

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

```
<!-- Example Endpoint Policy -->
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
          <sp:Issuer>...</sp:Issuer>
          <sp:RequestSecurityTokenTemplate>
            ...
          </sp:RequestSecurityTokenTemplate>
        </sp:IssuedToken>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EncryptBeforeSigning />
      <sp:EncryptSignature />
      <sp:ProtectTokens />
    </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <wsp:Policy>
      <sp:RequireSignatureConfirmation />
    </wsp:Policy>
  </sp:Wss11>
</wsp:Policy>
```

```

3283 <!-- Example Message Policy -->
3284 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3285   <sp:SignedParts>
3286     <sp:Header Name="Header1" Namespace="..." />
3287     <sp:Header Name="Header2" Namespace="..." />
3288     <sp:Body/>
3289   </sp:SignedParts>
3290   <sp:EncryptedParts>
3291     <sp:Header Name="Header2" Namespace="..." />
3292     <sp:Body/>
3293   </sp:EncryptedParts>
3294 </wsp:Policy>
3295

```

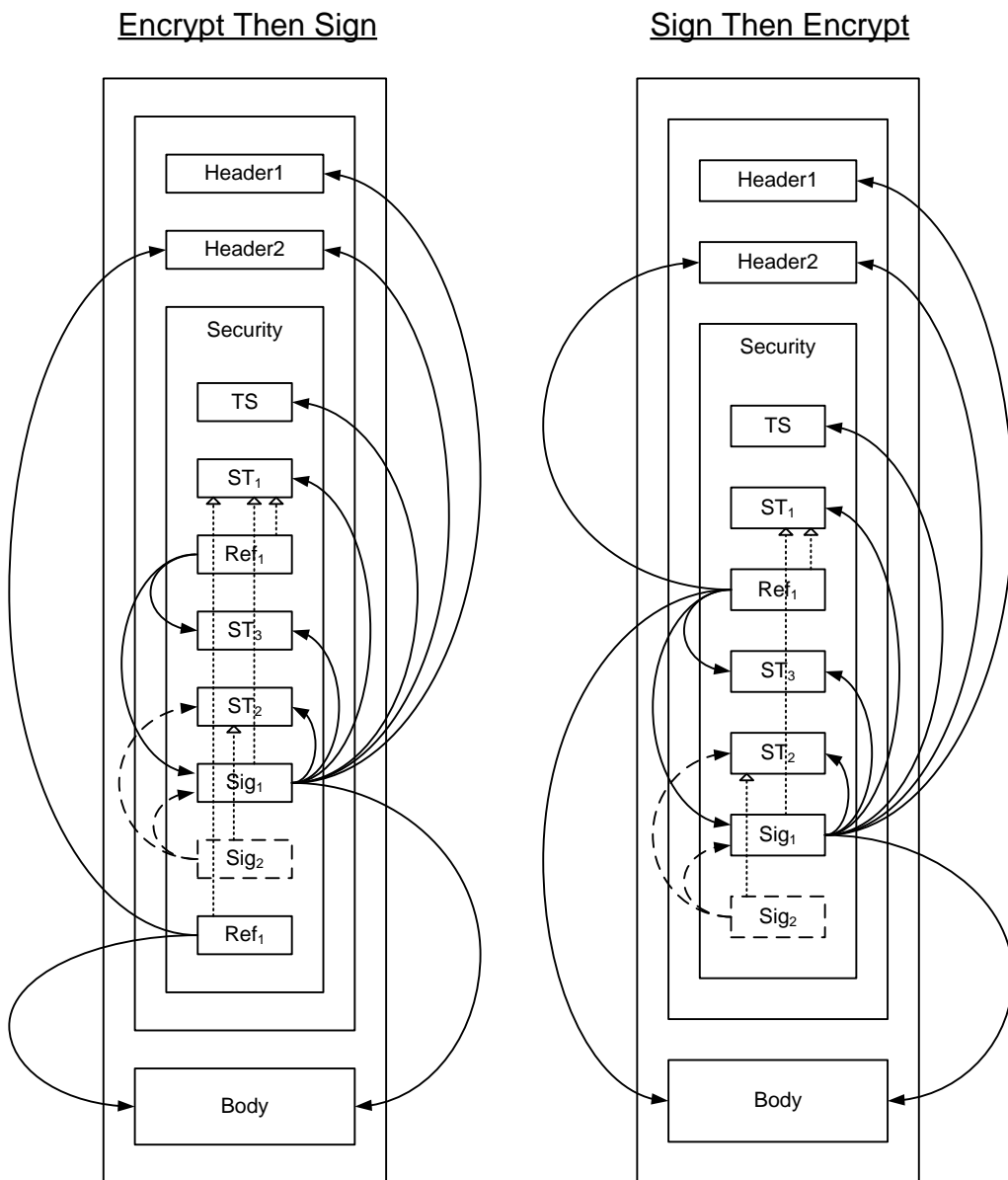
3296 This policy is used as the basis for the examples shown in the subsequent section describing the security
3297 header layout for this binding.

3298 C.2.2 Initiator to Recipient Messages

3299 Messages sent from initiator to recipient have the following layout for the security header:

- 3300 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3301 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
3302 `.../IncludeToken/Always`, then the [Encryption Token].
- 3303 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3304 Derived Key Token is used for encryption.
- 3305 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3306 reference list MUST include a reference to the message signature. If [Protection Order] is
3307 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3308 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3309 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3310 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3311 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
3312 `.../IncludeToken/Always`.
- 3313 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3314 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
3315 [Signature Token].
- 3316 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3317 Derived Key Token is used for signature.
- 3318 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3319 whether they are included in the message, and any message parts specified in SignedParts
3320 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3321 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3322 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3323 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3324 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3325 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3326 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3327 endorsing token, appears before the signature.
- 3328 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3329 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3330 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3331 above.

3332
 3333 The following diagram illustrates the security header layout for the initiator to recipient message:



3334
 3335 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 3336 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
 3337 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
 3338 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
 3339 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
 3340 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
 3341 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
 3342 process its contents.
 3343 *Example:*
 3344 Initiator to recipient message using EncryptBeforeSigning:

```
3345 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3346   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3347   xmlns:xenc="..." xmlns:ds="...">
3348   <S:Header>
3349     <x:Header1 wsu:Id="Header1" >
3350       ...
3351     </x:Header1>
3352
```

```

3353 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3354 <!-- Plaintext Header2
3355 <x:Header2 wsu:Id="Header2" >
3356 ...
3357 </x:Header2>
3358 -->
3359 ...
3360 </wsse1:EncryptedHeader>
3361 ...
3362 <wsse:Security>
3363 <wsu:Timestamp wsu:Id="Timestamp">
3364 <wsu:Created>...</wsu:Created>
3365 <wsu:Expires>...</wsu:Expires>
3366 </wsu:Timestamp>
3367 <saml:Assertion AssertionId="_SharedSecretToken" ...>
3368 ...
3369 </saml:Assertion>
3370 <xenc:ReferenceList>
3371 <xenc:DataReference URI="#enc_Signature" />
3372 <xenc:DataReference URI="#enc_SomeUsernameToken" />
3373 ...
3374 </xenc:ReferenceList>
3375 <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3376 <!-- Plaintext UsernameToken
3377 <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3378 ...
3379 </wsse:UsernameToken>
3380 -->
3381 ...
3382 <ds:KeyInfo>
3383 <wsse:SecurityTokenReference>
3384 <wsse:Reference URI="#_SharedSecretToken" />
3385 </wsse:SecurityTokenReference>
3386 </ds:KeyInfo>
3387 </xenc:EncryptedData>
3388 <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3389 ...
3390 </wsse:BinarySecurityToken>
3391 <xenc:EncryptedData ID="enc_Signature">
3392 <!-- Plaintext Signature
3393 <ds:Signature Id="Signature">
3394 <ds:SignedInfo>
3395 <ds:References>
3396 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3397 <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3398 <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3399 <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3400 <ds:Reference URI="#Header1" >...</ds:Reference>
3401 <ds:Reference URI="#Header2" >...</ds:Reference>
3402 <ds:Reference URI="#Body" >...</ds:Reference>
3403 </ds:References>
3404 </ds:SignedInfo>
3405 <ds:SignatureValue>...</ds:SignatureValue>
3406 <ds:KeyInfo>
3407 <wsse:SecurityTokenReference>
3408 <wsse:Reference URI="#_SharedSecretToken" />
3409 </wsse:SecurityTokenReference>
3410 </ds:KeyInfo>
3411 </ds:Signature>
3412 -->
3413 ...
3414 <ds:KeyInfo>
3415 <wsse:SecurityTokenReference>
3416 <wsse:Reference URI="#_SharedSecretToken" />

```

```

3417         </wsse:SecurityTokenReference>
3418     </ds:KeyInfo>
3419 </xenc:EncryptedData>
3420 <ds:Signature>
3421     <ds:SignedInfo>
3422         <ds:References>
3423             <ds:Reference URI="#Signature" >...</ds:Reference>
3424             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3425         </ds:References>
3426     </ds:SignedInfo>
3427     <ds:SignatureValue>...</ds:SignatureValue>
3428     <ds:KeyInfo>
3429         <wsse:SecurityTokenReference>
3430             <wsse:Reference URI="#SomeSupportingToken" />
3431         </wsse:SecurityTokenReference>
3432     </ds:KeyInfo>
3433 </ds:Signature>
3434 <xenc:ReferenceList>
3435     <xenc:DataReference URI="#enc_Body" />
3436     <xenc:DataReference URI="#enc_Header2" />
3437     ...
3438 </xenc:ReferenceList>
3439 </wsse:Security>
3440 </S:Header>
3441 <S:Body wsu:Id="Body">
3442     <xenc:EncryptedData Id="enc_Body">
3443         ...
3444         <ds:KeyInfo>
3445             <wsse:SecurityTokenReference>
3446                 <wsse:Reference URI="#_SharedSecretToken" />
3447             </wsse:SecurityTokenReference>
3448         </ds:KeyInfo>
3449     </xenc:EncryptedData>
3450 </S:Body>
3451 </S:Envelope>

```

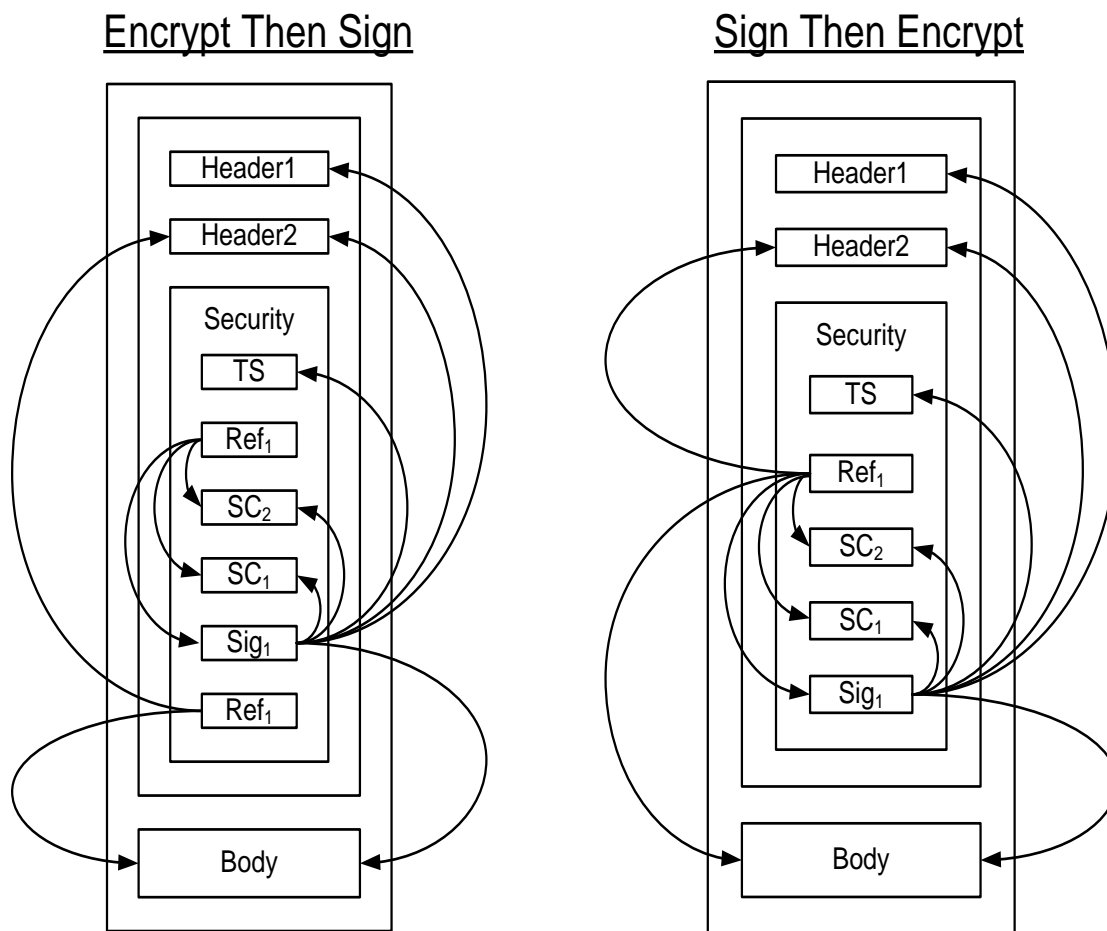
C.2.3 Recipient to Initiator Messages

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

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

7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This Derived Key Token is used for signature.
8. A signature over the wsu:Timestamp from 1 above, any `wsse11:SignatureConfirmation` elements from 5 above, and all the message parts specified in SignedParts assertions in the policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token from 6 above MUST be used, otherwise the key in the [Signature Token].
9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption Token].

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



The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁. The arrows on the left from boxes labeled Ref₁ indicate references to parts encrypted using a key based on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two `wsse11:SignatureConfirmation` elements labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is included. In general, the ordering of the items in the security header follows the most optimal layout for a receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

Example:

3493 Recipient to initiator message using EncryptBeforeSigning:

```
3494 <S:Envelope>
3495   <S:Header>
3496     <x:Header1 wsu:Id="Header1" >
3497       ...
3498     </x:Header1>
3499     <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3500       <!-- Plaintext Header2
3501       <x:Header2 wsu:Id="Header2" >
3502         ...
3503       </x:Header2>
3504       -->
3505     </wsse1:EncryptedHeader>
3506     ...
3507   <wsse:Security>
3508     <wsu:Timestamp wsu:Id="Timestamp">
3509       <wsu:Created>...</wsu:Created>
3510       <wsu:Expires>...</wsu:Expires>
3511     </wsu:Timestamp>
3512     <xenc:ReferenceList>
3513       <xenc:DataReference URI="#enc_Signature" />
3514       <xenc:DataReference URI="#enc_SigConf1" />
3515       <xenc:DataReference URI="#enc_SigConf2" />
3516       ...
3517     </xenc:ReferenceList>
3518     <xenc:EncryptedData ID="enc_SigConf1" >
3519       <!-- Plaintext SignatureConfirmation
3520       <wsse1:SignatureConfirmation wsu:Id="SigConf1" >
3521         ...
3522       </wsse1:SignatureConfirmation>
3523       -->
3524     </xenc:EncryptedData>
3525     <xenc:EncryptedData ID="enc_SigConf2" >
3526       <!-- Plaintext SignatureConfirmation
3527       <wsse1:SignatureConfirmation wsu:Id="SigConf2" >
3528         ...
3529       </wsse1:SignatureConfirmation>
3530       -->
3531     </xenc:EncryptedData>
```

```

3535 <xenc:EncryptedData Id="enc_Signature">
3536   <!-- Plaintext Signature
3537   <ds:Signature Id="Signature">
3538     <ds:SignedInfo>
3539       <ds:References>
3540         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3541         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3542         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3543         <ds:Reference URI="#Header1" >...</ds:Reference>
3544         <ds:Reference URI="#Header2" >...</ds:Reference>
3545         <ds:Reference URI="#Body" >...</ds:Reference>
3546       </ds:References>
3547     </ds:SignedInfo>
3548     <ds:SignatureValue>...</ds:SignatureValue>
3549     <ds:KeyInfo>
3550       <wsse:SecurityTokenReference>
3551         <wsse:Reference URI="#_SomeIssuedToken" />
3552       </wsse:SecurityTokenReference>
3553     </ds:KeyInfo>
3554   </ds:Signature>
3555   -->
3556 </xenc:EncryptedData>
3557 ...
3558 <ds:KeyInfo>
3559   <wsse:SecurityTokenReference>
3560     <wsse:Reference URI="#_SomeIssuedToken" />
3561   </wsse:SecurityTokenReference>
3562 </ds:KeyInfo>
3563 <xenc:EncryptedData>
3564 <xenc:ReferenceList>
3565   <xenc:DataReference URI="#enc_Body" />
3566   <xenc:DataReference URI="#enc_Header2" />
3567   ...
3568 </xenc:ReferenceList>
3569 </xenc:EncryptedData>
3570 </wsse:Security>
3571 </S:Header>
3572 <S:Body wsu:Id="Body">
3573   <xenc:EncryptedData Id="enc_Body">
3574     ...
3575     <ds:KeyInfo>
3576       <wsse:SecurityTokenReference>
3577         <wsse:Reference URI="#_SomeIssuedToken" />
3578       </wsse:SecurityTokenReference>
3579     </ds:KeyInfo>
3580   </xenc:EncryptedData>
3581 </S:Body>
3582 </S:Envelope>
3583

```

C.3 Asymmetric Binding

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

C.3.1 Policy

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

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

```
<!-- Example Endpoint Policy -->
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:AsymmetricBinding>
    <wsp:Policy>
      <sp:RecipientToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
        </wsp:Policy>
      </sp:RecipientToken>
      <sp:InitiatorToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
        </wsp:Policy>
      </sp:InitiatorToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EncryptBeforeSigning />
      <sp:EncryptSignature />
      <sp:ProtectTokens />
    </wsp:Policy>
  </sp:AsymmetricBinding>
  <sp:SignedEncryptedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedEncryptedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <wsp:Policy>
      <sp:RequireSignatureConfirmation />
    </wsp:Policy>
  </sp:Wss11>
</wsp:Policy>
```

```

3646 <!-- Example Message Policy -->
3647 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3648   <sp:SignedParts>
3649     <sp:Header Name="Header1" Namespace="..." />
3650     <sp:Header Name="Header2" Namespace="..." />
3651     <sp:Body/>
3652   </sp:SignedParts>
3653   <sp:EncryptedParts>
3654     <sp:Header Name="Header2" Namespace="..." />
3655     <sp:Body/>
3656   </sp:EncryptedParts>
3657 </wsp:All>

```

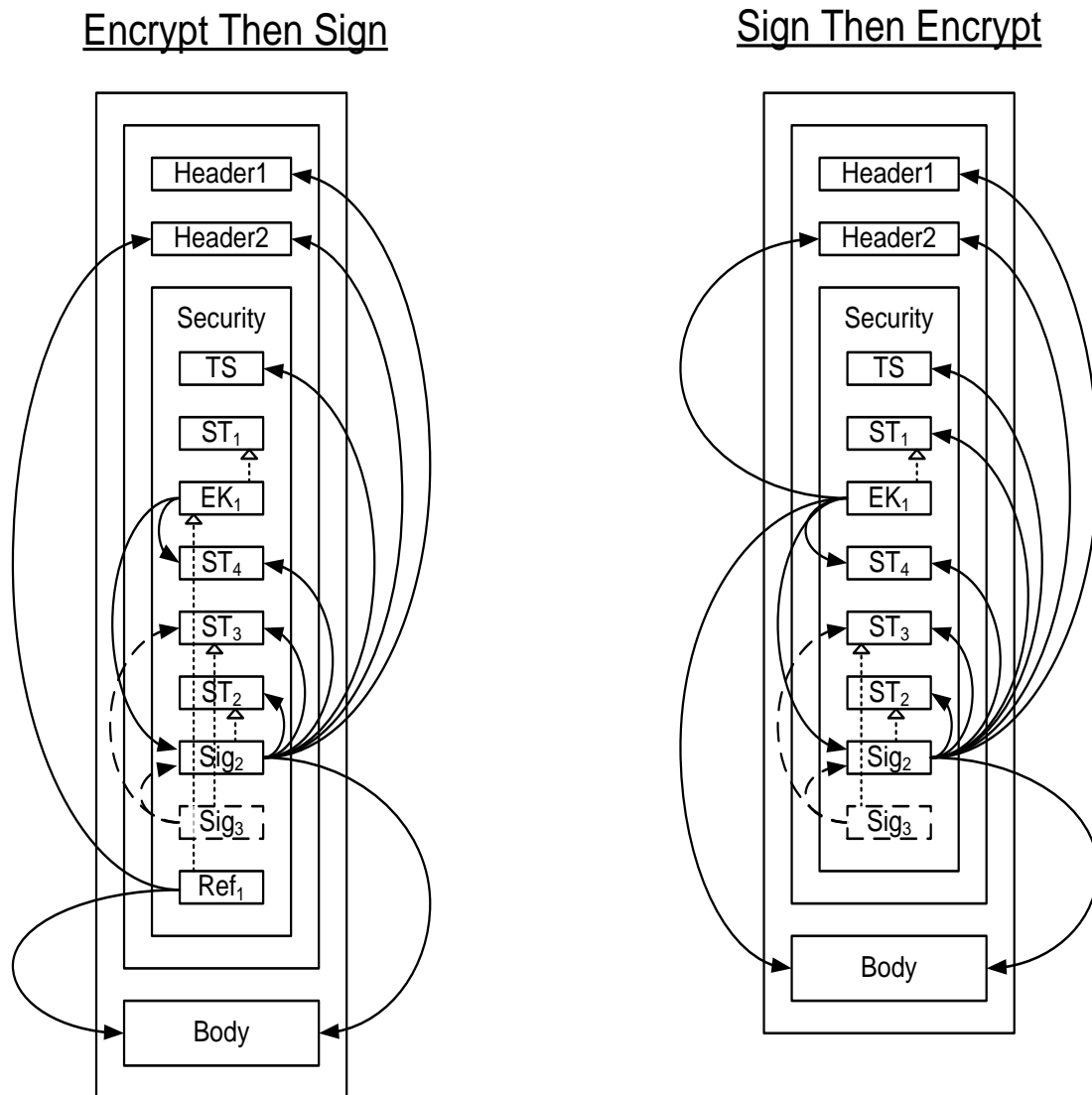
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.3.2 Initiator to Recipient Messages

Messages sent from initiator to recipient have the following layout:

1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a reference to all the message parts specified in `EncryptedParts` assertions in the policy. If [Signature Protection] is 'true' then the reference list MUST contain a reference to the message signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message signature.
4. Any tokens from the supporting tokens properties (as defined in section 8) whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from 1 above, any tokens from 4 above regardless of whether they are included in the message, and any message parts specified in `SignedParts` assertions in the policy. If [Token Protection] is 'true', the signature MUST also cover the [Initiator Token] regardless of whether it is included in the message.
7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a Derived Key Token, based on the supporting token, appears before the signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token regardless of whether it is included in the message.
8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The reference list includes a reference to all the message parts specified in `EncryptedParts` assertions in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey` element from 3 above.

3694 The following diagram illustrates the security header layout for the initiator to recipient messages:



3695
3696 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
3697 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate
3698 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as
3699 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references
3700 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left
3701 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the
3702 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as
3703 the basis for each cryptographic operation. In general, the ordering of the items in the security header
3704 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
3705 ordering are described in Appendix C.

3706
3707 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
3708 key contains an external reference to the token containing the encryption key. The diagram illustrates
3709 how one might attach a security token related to the encrypted key for completeness. One possible use-

3710 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3711 wishes to include the encryption token in the message signature.

3712 Initiator to recipient message *Example*

3713 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3714     xmlns:wsse1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3715 <S:Header>
3716   <x:Header1 wsu:Id="Header1" >
3717     ...
3718   </x:Header1>
3719   <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3720     <!-- Plaintext Header2
3721     <x:Header2 wsu:Id="Header2" >
3722       ...
3723     </x:Header2>
3724     -->
3725     ...
3726   </wsse1:EncryptedHeader>
3727   ...
3728   <wsse:Security>
3729     <wsu:Timestamp wsu:Id="Timestamp">
3730       <wsu:Created>...</wsu:Created>
3731       <wsu:Expires>...</wsu:Expires>
3732     </wsu:Timestamp>
3733     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3734       ...
3735     </wsse:BinarySecurityToken>
3736     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3737       ...
3738       <xenc:ReferenceList>
3739         <xenc:DataReference URI="#enc_Signature" />
3740         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3741         ...
3742       </xenc:ReferenceList>
3743     </xenc:EncryptedKey>
3744     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3745       <!-- Plaintext UsernameToken
3746       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3747         ...
3748       </wsse:UsernameToken>
3749       -->
3750       ...
3751     </xenc:EncryptedData>
3752     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3753       ...
3754     </wsse:BinarySecurityToken>
3755     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3756       ...
3757     </wsse:BinarySecurityToken>
3758     <xenc:EncryptedData ID="enc_Signature">
3759       <!-- Plaintext Signature
3760       <ds:Signature Id="Signature">
3761         <ds:SignedInfo>
3762           <ds:References>
3763             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3764             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3765             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3766             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3767             <ds:Reference URI="#Header1" >...</ds:Reference>
3768             <ds:Reference URI="#Header2" >...</ds:Reference>
3769             <ds:Reference URI="#Body" >...</ds:Reference>
3770           </ds:References>
3771         </ds:SignedInfo>
3772         <ds:SignatureValue>...</ds:SignatureValue>
3773         <ds:KeyInfo>
3774           <wsse:SecurityTokenReference>
3775             <wsse:Reference URI="#InitiatorToken" />
3776           </wsse:SecurityTokenReference>
3777         </ds:KeyInfo>

```

```

3778     </ds:Signature>
3779     -->
3780     ...
3781 </xenc:EncryptedData>
3782 <ds:Signature>
3783   <ds:SignedInfo>
3784     <ds:References>
3785       <ds:Reference URI="#Signature" >...</ds:Reference>
3786       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3787     </ds:References>
3788   </ds:SignedInfo>
3789   <ds:SignatureValue>...</ds:SignatureValue>
3790   <ds:KeyInfo>
3791     <wsse:SecurityTokenReference>
3792       <wsse:Reference URI="#SomeSupportingToken" />
3793     </wsse:SecurityTokenReference>
3794   </ds:KeyInfo>
3795 </ds:Signature>
3796 <xenc:ReferenceList>
3797   <xenc:DataReference URI="#enc_Body" />
3798   <xenc:DataReference URI="#enc_Header2" />
3799   ...
3800 </xenc:ReferenceList>
3801 </wsse:Security>
3802 </S:Header>
3803 <S:Body wsu:Id="Body">
3804   <xenc:EncryptedData Id="enc_Body">
3805     ...
3806     <ds:KeyInfo>
3807       <wsse:SecurityTokenReference>
3808         <wsse:Reference URI="#RecipientEncryptedKey" />
3809       </wsse:SecurityTokenReference>
3810     </ds:KeyInfo>
3811   </xenc:EncryptedData>
3812 </S:Body>
3813 </S:Envelope>

```

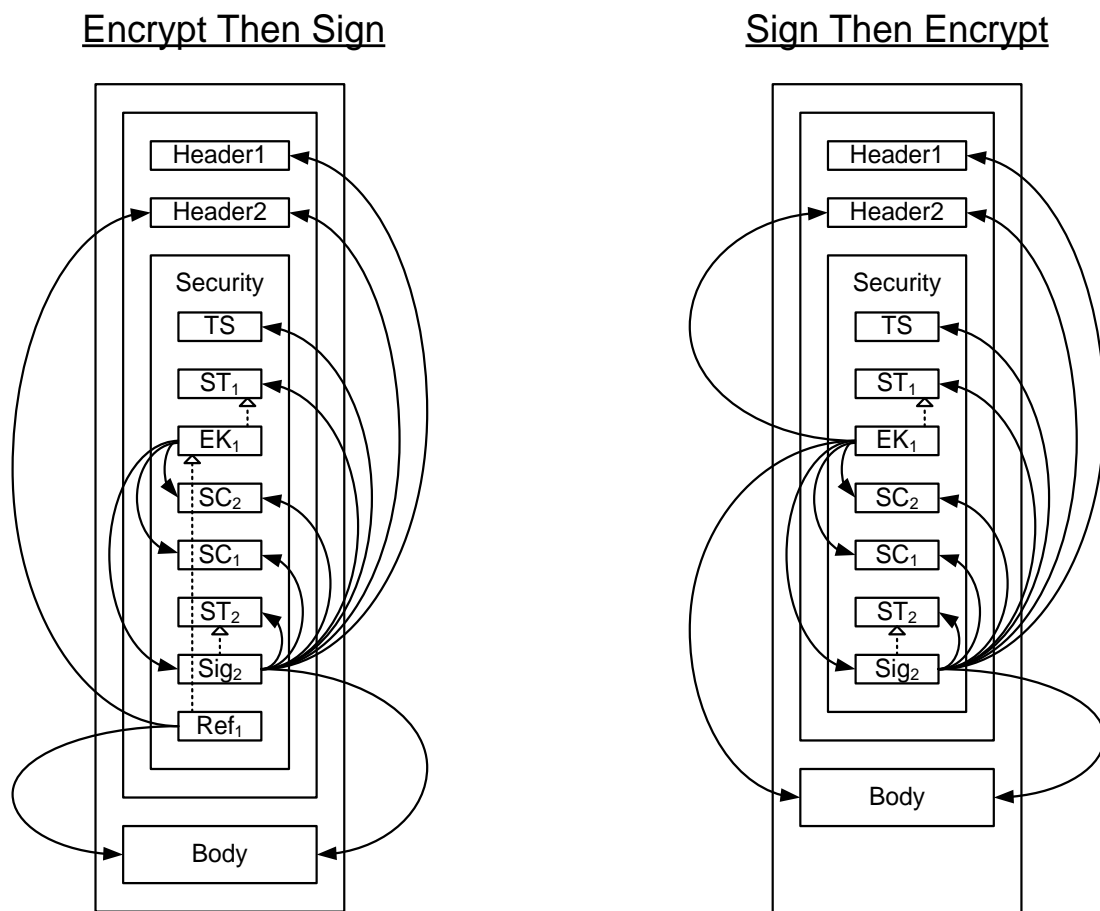
3814 C.3.3 Recipient to Initiator Messages

3815 Messages sent from recipient to initiator have the following layout:

- 3816 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3817 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3818 `.../IncludeToken/Always`, then the [Initiator Token].
- 3819 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3820 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3821 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3822 reference to all the message parts specified in EncryptedParts assertions in the policy. If
3823 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3824 message signature from 6 below, if any and references to the
3825 `wssell:SignatureConfirmation` elements from 4 below, if any.
- 3826 4. If [Signature Confirmation] is 'true', then a `wssell:SignatureConfirmation` element for each
3827 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3828 in the corresponding message from the initiator to the recipient, then a
3829 `wssell:SignatureConfirmation` element with no Value attribute.
- 3830 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3831 `.../IncludeToken/Always`, then the [Recipient Token].

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

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



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

Recipient to initiator message *Example*:

```

3855 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3856   xmlns:wssell="..." xmlns:wsse="..."
3857   xmlns:xenc="..." xmlns:ds="...">
3858 <S:Header>
3859   <x:Header1 wsu:Id="Header1" >
3860     ...
3861   </x:Header1>
3862   <wssell:EncryptedHeader wsu:Id="enc_Header2">
3863     <!-- Plaintext Header2
3864     <x:Header2 wsu:Id="Header2" >
3865       ...
3866     </x:Header2>
3867     -->
3868     ...
3869   </wssell:EncryptedHeader>
3870   ...
3871 <wsse:Security>
3872   <wsu:Timestamp wsu:Id="Timestamp">
3873     <wsu:Created>...</wsu:Created>
3874     <wsu:Expires>...</wsu:Expires>
3875   </wsu:Timestamp>
3876   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3877     ...
3878   </wsse:BinarySecurityToken>
3879   <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3880     ...
3881     <xenc:ReferenceList>
3882       <xenc:DataReference URI="#enc_Signature" />
3883       <xenc:DataReference URI="#enc_SigConf1" />
3884       <xenc:DataReference URI="#enc_SigConf2" />
3885       ...
3886     </xenc:ReferenceList>
3887   </xenc:EncryptedKey>
3888   <xenc:EncryptedData ID="enc_SigConf2" >
3889     <!-- Plaintext SignatureConfirmation
3890     <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3891       ...
3892     </wssell:SignatureConfirmation>
3893     -->
3894     ...
3895   </xenc:EncryptedData>
3896   <xenc:EncryptedData ID="enc_SigConf1" >
3897     <!-- Plaintext SignatureConfirmation
3898     <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3899       ...
3900     </wssell:SignatureConfirmation>
3901     -->
3902     ...
3903   </xenc:EncryptedData>
3904   <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3905     ...
3906   </wsse:BinarySecurityToken>
3907

```

```

3908 <xenc:EncryptedData ID="enc_Signature">
3909   <!-- Plaintext Signature
3910   <ds:Signature Id="Signature">
3911     <ds:SignedInfo>
3912       <ds:References>
3913         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3914         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3915         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3916         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3917         <ds:Reference URI="#Header1" >...</ds:Reference>
3918         <ds:Reference URI="#Header2" >...</ds:Reference>
3919         <ds:Reference URI="#Body" >...</ds:Reference>
3920       </ds:References>
3921     </ds:SignedInfo>
3922     <ds:SignatureValue>...</ds:SignatureValue>
3923     <ds:KeyInfo>
3924       <wsse:SecurityTokenReference>
3925         <wsse:Reference URI="#RecipientToken" />
3926       </wsse:SecurityTokenReference>
3927     </ds:KeyInfo>
3928   </ds:Signature>
3929   -->
3930   ...
3931 </xenc:EncryptedData>
3932 <xenc:ReferenceList>
3933   <xenc:DataReference URI="#enc_Body" />
3934   <xenc:DataReference URI="#enc_Header2" />
3935   ...
3936 </xenc:ReferenceList>
3937 </wsse:Security>
3938 </S:Header>
3939 <S:Body wsu:Id="Body">
3940   <xenc:EncryptedData Id="enc_Body">
3941     ...
3942     <ds:KeyInfo>
3943       <wsse:SecurityTokenReference>
3944         <wsse:Reference URI="#InitiatorEncryptedKey" />
3945       </wsse:SecurityTokenReference>
3946     </ds:KeyInfo>
3947   </xenc:EncryptedData>
3948 </S:Body>
3949 </S:Envelope>

```

D. Signed and Encrypted Elements in the Security Header

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

D.1 Elements signed by the message signature

1. The `wsu:Timestamp` element (Section 6.2).
2. All `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).

D.2 Elements signed by all endorsing signatures

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

D.3 Elements signed by a specific endorsing signature

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

D.4 Elements that are encrypted

1. The `ds:Signature` element that forms the message signature when [Signature Protection] has a value of 'true' (Section 6.4).
2. All `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).

E. Acknowledgements

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

Original Authors of the initial contribution:

Giovanni Della-Libera, Microsoft
Martin Gudgin, Microsoft
Phillip Hallam-Baker, VeriSign
Maryann Hondo, IBM
Hans Granqvist, Verisign
Chris Kaler, Microsoft (editor)
Hiroshi Maruyama, IBM
Michael McIntosh, IBM
Anthony Nadalin, IBM (editor)
Nataraj Nagaratnam, IBM
Rob Philpott, RSA Security
Hemma Prafullchandra, VeriSign
John Shewchuk, Microsoft
Doug Walter, Microsoft
Riaz Zolfonoon, RSA Security

Original Acknowledgements of the initial contribution:

Vaithialingam B. Balayoghan, Microsoft
Francisco Curbera, IBM
Christopher Ferris, IBM
Cédric Fournet, Microsoft
Andy Gordon, Microsoft
Tomasz Janczuk, Microsoft
David Melgar, IBM
Mike Perks, IBM
Bruce Rich, IBM
Jeffrey Schlimmer, Microsoft
Chris Sharp, IBM
Kent Tamura, IBM
T.R. Vishwanath, Microsoft
Elliot Waingold, Microsoft

TC Members during the development of this specification:

Don Adams, Tibco Software Inc.
Jan Alexander, Microsoft Corporation
Steve Anderson, BMC Software
Donal Arundel, IONA Technologies
Howard Bae, Oracle Corporation
Abbie Barbir, Nortel Networks Limited
Charlton Barreto, Adobe Systems
Mighael Botha, Software AG, Inc.
Toufic Boubez, Layer 7 Technologies Inc.
Norman Brickman, Mitre Corporation
Melissa Brumfield, Booz Allen Hamilton

4027 Geoff Bullen, Microsoft Corporation
4028 Lloyd Burch, Novell
4029 Scott Cantor, Internet2
4030 Greg Carpenter, Microsoft Corporation
4031 Steve Carter, Novell
4032 Symon Chang, BEA Systems, Inc.
4033 Ching-Yun (C.Y.) Chao, IBM
4034 Martin Chapman, Oracle Corporation
4035 Kate Cherry, Lockheed Martin
4036 Henry (Hyenvui) Chung, IBM
4037 Luc Clement, Systinet Corp.
4038 Paul Cotton, Microsoft Corporation
4039 Glen Daniels, Sonic Software Corp.
4040 Peter Davis, Neustar, Inc.
4041 Duane DeCouteau, Veterans Health Administration
4042 Martijn de Boer, SAP AG
4043 Werner Dittmann, Siemens AG
4044 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
4045 Fred Dushin, IONA Technologies
4046 Petr Dvorak, Systinet Corp.
4047 Colleen Evans, Microsoft Corporation
4048 Ruchith Fernando, WSO2
4049 Mark Fussell, Microsoft Corporation
4050 Vijay Gajjala, Microsoft Corporation
4051 Marc Goodner, Microsoft Corporation
4052 Hans Granqvist, VeriSign
4053 Martin Gudgin, Microsoft Corporation
4054 Tony Gullotta, SOA Software Inc.
4055 Jiandong Guo, Sun Microsystems
4056 Phillip Hallam-Baker, VeriSign
4057 Patrick Harding, Ping Identity Corporation
4058 Heather Hinton, IBM
4059 Frederick Hirsch, Nokia Corporation
4060 Jeff Hodges, Neustar, Inc.
4061 Will Hopkins, BEA Systems, Inc.
4062 Alex Hristov, Otecia Incorporated
4063 John Hughes, PA Consulting
4064 Diane Jordan, IBM
4065 Venugopal K, Sun Microsystems
4066 Chris Kaler, Microsoft Corporation
4067 Dana Kaufman, Forum Systems, Inc.
4068 Paul Knight, Nortel Networks Limited
4069 Ramanathan Krishnamurthy, IONA Technologies
4070 Christopher Kurt, Microsoft Corporation
4071 Kelvin Lawrence, IBM
4072 Hubert Le Van Gong, Sun Microsystems
4073 Jong Lee, BEA Systems, Inc.
4074 Rich Levinson, Oracle Corporation
4075 Tommy Lindberg, Dajeil Ltd.
4076 Mark Little, JBoss Inc.
4077 Hal Lockhart, BEA Systems, Inc.
4078 Mike Lyons, Layer 7 Technologies Inc.
4079 Eve Maler, Sun Microsystems
4080 Ashok Malhotra, Oracle Corporation
4081 Anand Mani, CrimsonLogic Pte Ltd
4082 Jonathan Marsh, Microsoft Corporation
4083 Robin Martherus, Oracle Corporation

4084 Miko Matsumura, Infravio, Inc.
4085 Gary McAfee, IBM
4086 Michael McIntosh, IBM
4087 John Merrells, Sxip Networks SRL
4088 Jeff Mischkinsky, Oracle Corporation
4089 Prateek Mishra, Oracle Corporation
4090 Bob Morgan, Internet2
4091 Vamsi Motukuru, Oracle Corporation
4092 Raajmohan Na, EDS
4093 Anthony Nadalin, IBM
4094 Andrew Nash, Reactivity, Inc.
4095 Eric Newcomer, IONA Technologies
4096 Duane Nickull, Adobe Systems
4097 Toshihiro Nishimura, Fujitsu Limited
4098 Rob Philpott, RSA Security
4099 Denis Pilipchuk, BEA Systems, Inc.
4100 Darren Platt, Ping Identity Corporation
4101 Martin Raepple, SAP AG
4102 Nick Ragouzis, Enosis Group LLC
4103 Prakash Reddy, CA
4104 Alain Regnier, Ricoh Company, Ltd.
4105 Irving Reid, Hewlett-Packard
4106 Bruce Rich, IBM
4107 Tom Rutt, Fujitsu Limited
4108 Maneesh Sahu, Actional Corporation
4109 Frank Siebenlist, Argonne National Laboratory
4110 Joe Smith, Apani Networks
4111 Davanum Srinivas, WSO2
4112 David Staggs, Veterans Health Administration
4113 Yakov Sverdlov, CA
4114 Gene Thurston, AmberPoint
4115 Victor Valle, IBM
4116 Asir Vedamuthu, Microsoft Corporation
4117 Greg Whitehead, Hewlett-Packard
4118 Ron Williams, IBM
4119 Corinna Witt, BEA Systems, Inc.
4120 Kyle Young, Microsoft Corporation
4121
4122