



WS-SecurityPolicy 1.2

OASIS Standard incorporating Proposed Errata

30 April 2008

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-errata-cd-01.doc>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-errata-cd-01.pdf>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-errata-cd-01.html>

Previous Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.doc>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf>
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html>

Latest 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>

Artifact Type:

specification

Technical Committee:

OASIS Web Services Secure Exchange TC

Chair(s):

Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):

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

Related work:

N/A

Declared XML Namespace(s):

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

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. OASIS trademark, IPR and other policies apply.

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	22
4.3	Required Elements Assertion	23
4.3.1	RequiredElements Assertion	23
4.3.2	RequiredParts Assertion	24
5	Token Assertions	25
5.1	Token Inclusion	25
5.1.1	Token Inclusion Values	25
5.1.2	Token Inclusion and Token References	26
5.2	Token Issuer and Required Claims	26
5.2.1	Token Issuer	26
5.2.2	Token Issuer Name	26
5.2.3	Required Claims	26
5.2.4	Processing Rules and Token Matching	27
5.3	Token Properties	27
5.3.1	[Derived Keys] Property	27
5.3.2	[Explicit Derived Keys] Property	27
5.3.3	[Implied Derived Keys] Property	27
5.4	Token Assertion Types	27
5.4.1	UsernameToken Assertion	27

5.4.2	IssuedToken Assertion	29
5.4.3	X509Token Assertion	31
5.4.4	KerberosToken Assertion	33
5.4.5	SpnegoContextToken Assertion	34
5.4.6	SecurityContextToken Assertion	36
5.4.7	SecureConversationToken Assertion	37
5.4.8	SamlToken Assertion	40
5.4.9	RelToken Assertion	42
5.4.10	HttpsToken Assertion	43
5.4.11	KeyValueToken Assertion	44
6	Security Binding Properties	47
6.1	[Algorithm Suite] Property	47
6.2	[Timestamp] Property	49
6.3	[Protection Order] Property	49
6.4	[Signature Protection] Property	49
6.5	[Token Protection] Property	49
6.6	[Entire Header and Body Signatures] Property	50
6.7	[Security Header Layout] Property	50
6.7.1	Strict Layout Rules for WSS 1.0	50
7	Security Binding Assertions	52
7.1	AlgorithmSuite Assertion	52
7.2	Layout Assertion	54
7.3	TransportBinding Assertion	55
7.4	SymmetricBinding Assertion	56
7.5	AsymmetricBinding Assertion	58
8	Supporting Tokens	61
8.1	SupportingTokens Assertion	62
8.2	SignedSupportingTokens Assertion	63
8.3	EndorsingSupportingTokens Assertion	65
8.4	SignedEndorsingSupportingTokens Assertion	67
8.5	SignedEncryptedSupportingTokens Assertion	69
8.6	EncryptedSupportingTokens Assertion	69
8.7	EndorsingEncryptedSupportingTokens Assertion	69
8.8	SignedEndorsingEncryptedSupportingTokens Assertion	69
8.9	Interaction between [Token Protection] property and supporting token assertions	69
8.10	Example	70
9	WSS: SOAP Message Security Options	71
9.1	Wss10 Assertion	72
9.2	Wss11 Assertion	73
10	WS-Trust Options	75
10.1	Trust13 Assertion	76
11	Guidance on creating new assertions and assertion extensibility	78
11.1	General Design Points	78

11.2 Detailed Design Guidance	78
12 Security Considerations.....	80
A. Assertions and WS-PolicyAttachment	81
A.1 Endpoint Policy Subject Assertions	81
A.1.1 Security Binding Assertions	81
A.1.2 Token Assertions	81
A.1.3 WSS: SOAP Message Security 1.0 Assertions	81
A.1.4 WSS: SOAP Message Security 1.1 Assertions	81
A.1.5 Trust 1.0 Assertions	81
A.2 Operation Policy Subject Assertions	81
A.2.1 Security Binding Assertions	81
A.2.2 Supporting Token Assertions	81
A.3 Message Policy Subject Assertions	82
A.3.1 Supporting Token Assertions	82
A.3.2 Protection Assertions	82
A.4 Assertions With Undefined Policy Subject	82
A.4.1 General Assertions	82
A.4.2 Token Usage Assertions	82
A.4.3 Token Assertions	82
B. Issued Token Policy	84
C. Strict Security Header Layout Examples	86
C.1 Transport Binding	86
C.1.1 Policy	86
C.1.2 Initiator to Recipient Messages	87
C.1.3 Recipient to Initiator Messages	88
C.2 Symmetric Binding	89
C.2.1 Policy	90
C.2.2 Initiator to Recipient Messages	91
C.2.3 Recipient to Initiator Messages	95
C.3 Asymmetric Binding.....	98
C.3.1 Policy	98
C.3.2 Initiator to Recipient Messages	100
C.3.3 Recipient to Initiator Messages	104
D. Signed and Encrypted Elements in the Security Header	108
D.1 Elements signed by the message signature	108
D.2 Elements signed by all endorsing signatures	108
D.3 Elements signed by a specific endorsing signature	108
D.4 Elements that are encrypted	108
E. Acknowledgements.....	109

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]. The assertions defined within this specification have been designed to work independently of a specific version of WS-Policy. At the time of the publication of this specification the versions of WS-Policy known to correctly compose with this specification are WS-Policy 1.2 and 1.5. Within this specification the use of the namespace prefix `wsp` refers generically to the WS-Policy namespace, not a specific version. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

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

1.1 Example

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

Table 1: Example security policy.

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

```

(20) </sp:SignedParts>
(21) <sp:EncryptedParts>
(22)   <sp:Body/>
(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 URI that MUST be used by implementations of this specification is:

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

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]
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

1.3 Schema Files

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

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2.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~~**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~~**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/ |
| [URI] | T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
http://www.ietf.org/rfc/rfc3986.txt |

153		
154	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
155		
156		http://www.ietf.org/rfc/rfc2068.txt
157		
158	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
159		http://www.ietf.org/rfc/rfc2246.txt
160		
161	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
162		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
163		
164	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
165		
166		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
167		
168	[WS-Policy]	W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
169		
170		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
171		W3C Candidate Recommendation "Web Services Policy 1.5 – Framework", 28 February 2007
172		
173		http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/
174		
175	[WS-PolicyAttachment]	W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
176		
177		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
178		
179		W3C Candidate Recommendation "Web Services Policy 1.5 – Attachment", 28 February 2007
180		
181		http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/
182		
183	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.3", September 2006
184		http://docs.oasis-open.org/ws-sx/ws-trust/200512
185		
186	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.3", September 2006
187		
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",
201		March 2004
202		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
203		token-profile-1.0.pdf
204		
205	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile
206		1.1", February 2006
207		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-
208		spec-os-UsernameTokenProfile.pdf
209		
210	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token
211		Profile", March 2004
212		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
213		profile-1.0.pdf
214		
215	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token
216		Profile", February 2006
217		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-
218		spec-os-x509TokenProfile.pdf
219		
220	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1",
221		February 2006
222		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-
223		spec-os-KerberosTokenProfile.pdf
224		
225	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile",
226		December 2004
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",
230		February 2006
231		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-
232		spec-os-SAMLTokenProfile.pdf
233		
234	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language
235		(REL) Token Profile", December 2004
236		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
237		
238	[WSS:RELTTokenProfile1.1]	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:SwAPProfile1.1]	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-](http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf)
 246 [spec-os-SwAProfile.pdf](http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-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 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second
 261 Edition", 28 October 2004.
 262 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
 263
 264 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second
 265 Edition", 28 October 2004.
 266 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
 267

268 1.6 Non-Normative References

269 None.
 270

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.

2.2 Nested Policy Assertions

Assertions **may-MAY** be used to further qualify a specific aspect of another assertion. For example, an assertion describing the set of algorithms to use **may-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~~ **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-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-MAY** choose to accept messages that do not conform to its policy.

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

- 358 • TransportBinding (Section 7.3)
- 359 • SymmetricBinding (Section 7.4)
- 360 • 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-RECOMMENDED~~ or ~~required~~ **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:

400 wsdl:portType
401 A policy expression containing one or more assertions with Endpoint Policy Subject MUST NOT
402 be attached to a wsdl:portType.
403 wsdl:binding
404 A policy expression containing one or more of the assertions with Endpoint Policy Subject
405 SHOULD be attached to a wsdl:binding.
406 wsdl:port
407 A policy expression containing one or more of the assertions with Endpoint Policy Subject MAY
408 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 />?
  ...
</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~~**OPTIONAL** empty element indicates that the entire body, that is the soap:Body element, its attributes and content, of the message needs to be integrity protected.

/sp:SignedParts/sp:Header

Presence of this ~~optional~~**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~~**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~~**REQUIRED** attribute indicates the namespace of the SOAP header(s) to be integrity protected.

`/sp:SignedParts/sp:Attachments`

Presence of this ~~optional~~**OPTIONAL** empty 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].

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>+
  ...
</sp:SignedElements>
```

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

`/sp:SignedElements`

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

`/sp:SignedElements/@XPathVersion`

This ~~optional~~**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:SignedElements/sp:XPath`

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

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

4.2 Confidentiality Assertions

Two mechanisms are defined for specifying the set of message parts to confidentiality 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.

4.2.1 EncryptedParts Assertion

The EncryptedParts assertion is used to specify the parts of the message that require confidentiality. This assertion can be satisfied with 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 EncryptedParts assertions present. Multiple EncryptedParts assertions present within a policy alternative are equivalent to a single EncryptedParts 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 confidentiality protection.

Syntax

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

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

/sp:EncryptedParts

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

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

/sp:EncryptedParts/sp:Body

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

/sp:EncryptedParts/sp:Header

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

541 /sp:EncryptedParts/sp:Header/@Name
 542 | This ~~optional~~**OPTIONAL** attribute indicates the local name of the SOAP header to be
 543 confidentiality protected. If this attribute is not specified, all SOAP headers whose namespace
 544 matches the Namespace attribute are to be protected.
 545 /sp:EncryptedParts/sp:Header/@Namespace
 546 | This ~~required~~**REQUIRED** attribute indicates the namespace of the SOAP header(s) to be
 547 confidentiality protected.
 548 /sp:EncryptedParts/sp:Attachments
 549 | Presence of this ~~optional~~**OPTIONAL** empty element indicates that all SwA (SOAP Messages with
 550 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message
 551 Security is used to accomplish this, all message parts other than the part containing the primary
 552 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security
 553 [WSS:SwAProfile1.1].

554 4.2.2 EncryptedElements Assertion

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

560

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

564 Syntax

```
565 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
566   <sp:XPath>xs:string</sp:XPath>+
567   ...
568 </sp:EncryptedElements>
```

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

570 /sp:EncryptedElements

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

573 /sp:EncryptedElements/@XPathVersion

574 | This ~~optional~~**OPTIONAL** attribute contains a URI which indicates the version of XPath to use. If
 575 no attribute is provided, then XPath 1.0 is assumed.

576 /sp:EncryptedElements/sp:XPath

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

581 4.2.3 ContentEncryptedElements Assertion

582 The ContentEncryptedElements assertion is used to specify arbitrary elements in the message that
 583 require confidentiality protection of their content. This assertion can be satisfied using WSS: SOAP
 584 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~~**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~~**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

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

631 /sp:RequiredElements/@XPathVersion

632 | This ~~optional~~**OPTIONAL** attribute contains a URI which indicates the version of XPath to use. If

633 no attribute is provided, then XPath 1.0 is assumed.

634 /sp:RequiredElements/sp:XPath

635 This element contains a string specifying an XPath expression that identifies the header elements

636 that a message MUST contain. The XPath expression is evaluated against the

637 S:Envelope/S:Header element node of the message. Multiple instances of this element ~~may~~**MAY**

638 appear within this assertion and ~~should~~**SHOULD** be treated as a combined XPath expression.

639 4.3.2 RequiredParts Assertion

640 RequiredParts is a QName based alternative to the RequiredElements assertion (which is based on

641 XPATH) for specifying header elements that MUST be present in the message. This assertion specifies

642 no security requirements.

643

644 There MAY be multiple RequiredParts assertions present. Multiple RequiredParts assertions present

645 within a policy alternative are equivalent to a single RequiredParts assertion containing the union of all

646 specified Header elements.

647 Syntax

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

651

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

653 /sp:RequiredParts/sp:Header

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

655 /sp:RequiredParts/sp:Header/@Name

656 | This ~~required~~**REQUIRED** attribute indicates the local name of the SOAPHeader that needs to be

657 present in the message.

658 /sp:RequiredParts/sp:Header/@Namespace

659 | This ~~required~~**REQUIRED** attribute indicates the namespace of the SOAP header that needs to be

660 present 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~~**MAY** also carry an ~~optional~~**OPTIONAL** `sp:IncludeToken` attribute. The schema type of this attribute is `xs:anyURI`. This attribute indicates whether the token ~~should~~**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:

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

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

682 5.1.2 Token Inclusion and Token References

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

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

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

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

699 5.2 Token Issuer and Required Claims

700 5.2.1 Token Issuer

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

705 5.2.2 Token Issuer Name

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

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

713 | While both sp:Issuer and sp:IssuerName elements are ~~optional~~**OPTIONAL** they are also mutually
714 exclusive and cannot be specified both at the same time.

715 5.2.3 Required Claims

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

720 | This element indicates the ~~required~~**REQUIRED** claims that the security token must contain in order to
721 satisfy the requirements of the token assertion.

723 | Individual token assertions ~~may~~**MAY** further limit what claims ~~may~~**MAY** be specified for that specific
724 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~~**REQUIRED** set of claims from ~~required~~**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~~**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 ... />
    ) ?
    (
      <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~~**OPTIONAL** attribute identifies the token inclusion value for this token assertion.

/sp:UsernameToken/sp:Issuer

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

/sp:UsernameToken/sp:IssuerName

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

/sp:UsernameToken/wst:Claims

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

/sp:UsernameToken/wsp:Policy

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

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

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

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

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

827 | /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

830 | /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

831 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Explicit
832 | Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this
833 | token to 'false'.

834 | /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

835 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Implied
836 | Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
837 | token to 'false'.

838 | /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

839 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a Username token
840 | should be used as defined in [WSS:UsernameTokenProfile1.0].

841 | /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

842 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a Username token
843 | should be used as defined in [WSS:UsernameTokenProfile1.1].

844 5.4.2 IssuedToken Assertion

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

849 Syntax

```
850 | <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
851 | (
852 |   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
853 |   <sp:IssuerName>xs:anyURI</sp:IssuerName>
854 | ) ?
```

```

855 <wst:Claims Dialect="..."> ... </wst:Claims> ?
856 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
857   ...
858 </sp:RequestSecurityTokenTemplate>
859 <wsp:Policy xmlns:wsp="...">
860   (
861     <sp:RequireDerivedKeys ... /> |
862     <sp:RequireImpliedDerivedKeys ... /> |
863     <sp:RequireExplicitDerivedKeys ... />
864   ) ?
865   <sp:RequireExternalReference ... /> ?
866   <sp:RequireInternalReference ... /> ?
867   ...
868 </wsp:Policy>
869 ...
870 </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~~**OPTIONAL** attribute identifies the token inclusion value for this token assertion.

/sp:IssuedToken/sp:Issuer

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

/sp:IssuedToken/sp:IssuerName

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

/sp:IssuedToken/wst:Claims

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

/sp:IssuedToken/sp:RequestSecurityTokenTemplate

This ~~required~~**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~~**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~~**OPTIONAL** attribute contains a WS-Trust specification namespace URI identifying the version of WS-Trust referenced by the contents of this element.

/sp:IssuedToken/wsp:Policy

This ~~required~~**REQUIRED** element identifies additional requirements for use of the `sp:IssuedToken` assertion.

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

This ~~optional~~**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:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

This ~~optional~~**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'.

903 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

904 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Implied

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

906 | token to 'false'.

907 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

908 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates whether an internal reference

909 | is ~~required~~**REQUIRED** when referencing this token.

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

911 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

912 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates whether an external

913 | reference is ~~required~~**REQUIRED** when referencing this token.

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

915 | Note: The IssuedToken ~~may~~**MAY** or ~~may not~~**MAY NOT** be associated with key material and such key

916 | material may be symmetric or asymmetric. The Binding assertion will imply the type of key associated

917 | with this token. Services ~~may~~**MAY** also include information in the

918 | sp:RequestSecurityTokenTemplate element to explicitly define the expected key type. See

919 | [Appendix B](#) for details of the sp:RequestSecurityTokenTemplate element.

920 5.4.3 X509Token Assertion

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

922 Syntax

```

923 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
924   (
925     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
926     <sp:IssuerName>xs:anyURI</sp:IssuerName>
927   ) ?
928   <wst:Claims Dialect="..."> ... </wst:Claims> ?
929   <wsp:Policy xmlns:wsp="...">
930     (
931       <sp:RequireDerivedKeys ... /> |
932       <sp:RequireExplicitDerivedKeys ... /> |
933       <sp:RequireImpliedDerivedKeys ... />
934     ) ?
935     <sp:RequireKeyIdentifierReference ... /> ?
936     <sp:RequireIssuerSerialReference ... /> ?
937     <sp:RequireEmbeddedTokenReference ... /> ?
938     <sp:RequireThumbprintReference ... /> ?
939     (
940       <sp:WssX509V3Token10 ... /> |
941       <sp:WssX509Pkcs7Token10 ... /> |
942       <sp:WssX509PkiPathV1Token10 ... /> |
943       <sp:WssX509V1Token11 ... /> |
944       <sp:WssX509V3Token11 ... /> |
945       <sp:WssX509Pkcs7Token11 ... /> |
946       <sp:WssX509PkiPathV1Token11 ... />
947     ) ?
948     ...
949   </wsp:Policy>
950   ...
951 </sp:X509Token>

```

952

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

954 /sp:X509Token

955 This identifies an X509Token assertion.

956 /sp:X509Token/@sp:IncludeToken

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

958 /sp:X509Token/sp:Issuer

959 This ~~optional~~OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the

960 issuer of the sp:X509Token.

961 /sp:X509Token/sp:IssuerName

962 This ~~optional~~OPTIONAL element, of type xs:anyURI, contains the logical name of the

963 sp:X509Token issuer.

964 /sp:X509Token/wst:Claims

965 This ~~optional~~OPTIONAL element identifies the ~~required~~REQUIRED claims that a security token

966 must contain in order to satisfy the token assertion requirements.

967 /sp:X509Token/wsp:Policy

968 This ~~required~~REQUIRED element identifies additional requirements for use of the sp:X509Token

969 assertion.

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

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

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

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

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

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

976 token to 'false'.

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

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

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

980 token to 'false'.

981 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

982 This ~~optional~~OPTIONAL element is a policy assertion that indicates that a key identifier reference

983 is ~~required~~REQUIRED when referencing this token.

984 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

985 This ~~optional~~OPTIONAL element is a policy assertion that indicates that an issuer serial reference

986 is ~~required~~REQUIRED when referencing this token.

987 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

988 This ~~optional~~OPTIONAL element is a policy assertion that indicates that an embedded token

989 reference is ~~required~~REQUIRED when referencing this token.

990 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

991 This ~~optional~~OPTIONAL element is a policy assertion that indicates that a thumbprint reference is

992 ~~required~~REQUIRED when referencing this token.

993 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

994 This ~~optional~~OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token

995 should be used as defined in [[WSS:X509TokenProfile1.0](#)].

996 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

997 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 PKCS7 token
998 | should be used as defined in [[WSS:X509TokenProfile1.0](#)].

999 | /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1000 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 PKI Path
1001 | Version 1 token should be used as defined in [[WSS:X509TokenProfile1.0](#)].

1002 | /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1003 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 Version 1 token
1004 | should be used as defined in [[WSS:X509TokenProfile1.1](#)].

1005 | /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1006 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 Version 3 token
1007 | should be used as defined in [[WSS:X509TokenProfile1.1](#)].

1008 | /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1009 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 PKCS7 token
1010 | should be used as defined in [[WSS:X509TokenProfile1.1](#)].

1011 | /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1012 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an X509 PKI Path
1013 | Version 1 token should be used as defined in [[WSS:X509TokenProfile1.1](#)].

1014 5.4.4 KerberosToken Assertion

1015 This element represents a requirement for a Kerberos token [[WSS:KerberosToken1.1](#)].

1016 Syntax

```

1017 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1018   (
1019     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1020     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1021   ) ?
1022   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1023   <wsp:Policy xmlns:wsp="...">
1024     (
1025       <sp:RequireDerivedKeys ... /> |
1026       <sp:RequireImpliedDerivedKeys ... /> |
1027       <sp:RequireExplicitDerivedKeys ... />
1028     ) ?
1029     <sp:RequireKeyIdentifierReference ... /> ?
1030     (
1031       <sp:WssKerberosV5ApReqToken11 ... /> |
1032       <sp:WssGssKerberosV5ApReqToken11 ... />
1033     ) ?
1034     ...
1035   </wsp:Policy>
1036   ...
1037 </sp:KerberosToken>

```

1039

1040 The following describes the attributes and elements listed in the schema outlined above:

1041 /sp:KerberosToken

1042 This identifies a KerberosV5ApReqToken assertion.

1043 /sp:KerberosToken/@sp:IncludeToken

1044 This ~~optional~~**OPTIONAL** attribute identifies the token inclusion value for this token assertion.

1045 /sp:KerberosToken/sp:Issuer
 1046 | This ~~optional~~**OPTIONAL** element, of type wsa:EndpointReferenceType, contains reference to the
 1047 | issuer of the sp:KerberosToken.

1048 /sp:KerberosToken/sp:IssuerName
 1049 | This ~~optional~~**OPTIONAL** element, of type xs:anyURI, contains the logical name of the
 1050 | sp:KerberosToken issuer.

1051 /sp:KerberosToken/wst:Claims
 1052 | This ~~optional~~**OPTIONAL** element identifies the ~~required~~**REQUIRED** claims that a security token
 1053 | must contain in order to satisfy the token assertion requirements.

1054 /sp:KerberosToken/wsp:Policy
 1055 | This ~~required~~**REQUIRED** element identifies additional requirements for use of the
 1056 | sp:KerberosToken assertion.

1057 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys
 1058 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys], [Explicit
 1059 | Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

1060 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1061 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Explicit
 1062 | Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this
 1063 | token to 'false'.

1064 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1065 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Implied
 1066 | Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
 1067 | token to 'false'.

1068 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1069 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a key identifier reference
 1070 | is ~~required~~**REQUIRED** when referencing this token.

1071 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11
 1072 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a Kerberos Version 5
 1073 | AP-REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

1074 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11
 1075 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a GSS Kerberos Version
 1076 | 5 AP-REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

1077 5.4.5 SpnegoContextToken Assertion

1078 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
 1079 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1080 Syntax

```
1081 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1082 (
1083   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1084   <sp:IssuerName>xs:anyURI</sp:IssuerName>
```

```

1085 ) ?
1086 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1087 <wsp:Policy xmlns:wsp="...">
1088   (
1089     <sp:RequireDerivedKeys ... /> |
1090     <sp:RequireImpliedDerivedKeys ... /> |
1091     <sp:RequireExplicitDerivedKeys ... />
1092   ) ?
1093   <sp:MustNotSendCancel ... /> ?
1094   <sp:MustNotSendAmend ... /> ?
1095   <sp:MustNotSendRenew ... /> ?
1096   ...
1097 </wsp:Policy>
1098 ...
1099 </sp:SpnegoContextToken>

```

1100

1101 The following describes the attributes and elements listed in the schema outlined above:

1102 /sp:SpnegoContextToken

1103 This identifies a SpnegoContextToken assertion.

1104 /sp:SpnegoContextToken/@sp:IncludeToken

1105 | This ~~optional~~OPTIONAL attribute identifies the token inclusion value for this token assertion.

1106 /sp:SpnegoContextToken/sp:Issuer

1107 | This ~~optional~~OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to
1108 the issuer for the Spnego Context Token.

1109 /sp:SpnegoContextToken/sp:IssuerName

1110 | This ~~optional~~OPTIONAL element, of type xs:anyURI, contains the logical name of the
1111 sp:SpnegoContextToken issuer.

1112 /sp:SpnegoContextToken/wst:Claims

1113 | This ~~optional~~OPTIONAL element identifies the ~~required~~REQUIRED claims that a security token
1114 must contain in order to satisfy the token assertion requirements.

1115 /sp:SpnegoContextToken/wsp:Policy

1116 | This ~~required~~REQUIRED element identifies additional requirements for use of the
1117 sp:SpnegoContextToken assertion.

1118 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1119 | This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit
1120 Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

1121 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1122 | This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit
1123 Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this
1124 token to 'false'.

1125 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1126 | This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied
1127 Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1128 token to 'false'.

1129 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1130 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the STS issuing the
1131 SP/Nego token does not support SCT/Cancel RST messages. If this assertion is missing it
1132 means that SCT/Cancel RST messages are supported by the STS.

1133 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend
1134 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the STS issuing the
1135 SP/Nego token does not support SCT/Amend RST messages. If this assertion is missing it
1136 means that SCT/Amend RST messages are supported by the STS.

1137 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew
1138 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the STS issuing the
1139 SP/Nego token does not support SCT/Renew RST messages. If this assertion is missing it
1140 means that SCT/Renew RST messages are supported by the STS.

1141 5.4.6 SecurityContextToken Assertion

1142 This element represents a requirement for a SecurityContextToken token.

1143 Syntax

```
1144 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1145 (  
1146   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1147   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1148 ) ?  
1149 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1150 <wsp:Policy xmlns:wsp="...">  
1151   (  
1152     <sp:RequireDerivedKeys ... /> |  
1153     <sp:RequireImpliedDerivedKeys ... /> |  
1154     <sp:RequireExplicitDerivedKeys ... />  
1155   ) ?  
1156   <sp:RequireExternalUriReference ... /> ?  
1157   <sp:SC13SecurityContextToken... /> ?  
1158   ...  
1159 </wsp:Policy>  
1160 ...  
1161 </sp:SecurityContextToken>
```

1162
1163 The following describes the attributes and elements listed in the schema outlined above:

1164 /sp:SecurityContextToken

1165 | This identifies a SecurityContextToken assertion.

1166 /sp:SecurityContextToken/@sp:IncludeToken

1167 | This ~~optional~~**OPTIONAL** attribute identifies the token inclusion value for this token assertion.

1168 /sp:SecurityContextToken/sp:Issuer

1169 | This ~~optional~~**OPTIONAL** element, of type wsa:EndpointReferenceType, contains reference to the
1170 issuer of the sp:SecurityContextToken.

1171 /sp:SecurityContextToken/sp:IssuerName

1172 | This ~~optional~~**OPTIONAL** element, of type xs:anyURI, contains the logical name of the
1173 sp:SecurityContextToken issuer.

1174 /sp:SecurityContextToken/wst:Claims

1175 | This ~~optional~~**OPTIONAL** element identifies the ~~required~~**REQUIRED** claims that a security token
1176 must contain in order to satisfy the token assertion requirements.

1177 /sp:SecurityContextToken/wsp:Policy

1178 | This ~~required~~**REQUIRED** element identifies additional requirements for use of the
1179 sp:SecurityContextToken assertion.

1180 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1181 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys], [Explicit

1182 | Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

1183 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1184 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Explicit

1185 | Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this

1186 | token to 'false'.

1187 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1188 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Implied

1189 | Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this

1190 | token to 'false'.

1191 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1192 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that an external URI

1193 | reference is ~~required~~**REQUIRED** when referencing this token.

1194 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1195 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a Security Context Token

1196 | should be used as defined in [WS-SecureConversation].

1197

1198 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that

1199 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If

1200 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the

1201 | sp:SecureConversationToken or the sp:IssuedToken assertion ~~should~~**SHOULD** be used instead.

1202 5.4.7 SecureConversationToken Assertion

1203 This element represents a requirement for a Security Context Token retrieved from the indicated issuer

1204 address. If the sp:Issuer address is absent, the protocol **MUST** be executed at the same address as the

1205 service endpoint address.

1206

1207 Note: This assertion describes the token accepted by the target service. Because this token is issued by

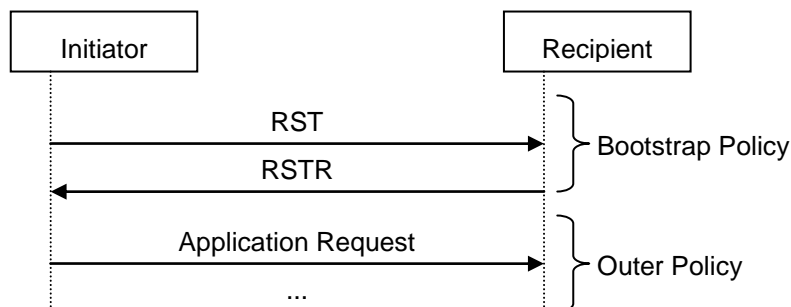
1208 | the target service and ~~may not~~**MAY NOT** have a separate port (with separate policy), this assertion

1209 | **SHOULD** contain a bootstrap policy indicating the security binding and policy that is used when

1210 | requesting this token from the target service. That is, the bootstrap policy is used to obtain the token and

1211 | then the current (outer) policy is used when making requests with the token. This is illustrated in the

1212 | diagram below.



1214 Syntax

1215 `<sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >`

1216 `(`

```

1217 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1218 <sp:IssuerName>xs:anyURI</sp:IssuerName>
1219 ) ?
1220 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1221 <wsp:Policy xmlns:wsp="...">
1222 (
1223   <sp:RequireDerivedKeys ... /> |
1224   <sp:RequireImpliedDerivedKeys ... /> |
1225   <sp:RequireExplicitDerivedKeys ... />
1226 ) ?
1227 <sp:RequireExternalUriReference ... /> ?
1228 <sp:SC13SecurityContextToken ... /> ?
1229 <sp:MustNotSendCancel ... /> ?
1230 <sp:MustNotSendAmend ... /> ?
1231 <sp:MustNotSendRenew ... /> ?
1232 <sp:BootstrapPolicy ... >
1233   <wsp:Policy> ... </wsp:Policy>
1234 </sp:BootstrapPolicy> ?
1235 </wsp:Policy>
1236 ...
1237 </sp:SecureConversationToken>

```

1238

1239 The following describes the attributes and elements listed in the schema outlined above:

1240 /sp:SecureConversationToken

1241 This identifies a SecureConversationToken assertion.

1242 /sp:SecureConversationToken/@sp:IncludeToken

1243 This ~~optional~~OPTIONAL attribute identifies the token inclusion value for this token assertion.

1244 /sp:SecureConversationToken/sp:Issuer

1245 This ~~optional~~OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to
1246 the issuer for the Security Context Token.

1247 /sp:SecureConversationToken/sp:IssuerName

1248 This ~~optional~~OPTIONAL element, of type xs:anyURI, contains the logical name of the
1249 sp:SecureConversationToken issuer.

1250 /sp:SpnegoContextToken/wst:Claims

1251 This ~~optional~~OPTIONAL element identifies the ~~required~~REQUIRED claims that a security token
1252 must contain in order to satisfy the token assertion requirements.

1253 /sp:SecureConversationToken/wsp:Policy

1254 This ~~required~~REQUIRED element identifies additional requirements for use of the
1255 sp:SecureConversationToken assertion.

1256 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1257 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit
1258 Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

1259 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1260 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit
1261 Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this
1262 token to 'false'.

1263 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1264 | This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied
1265 | Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1266 | token to 'false'.

1267 | /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1268 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that an external URI
1269 | reference is ~~required~~REQUIRED when referencing this token.

1270 | /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken

1271 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that a Security Context Token
1272 | should be used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1273 | /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

1274 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the STS issuing the
1275 | secure conversation token does not support SCT/Cancel RST messages. If this assertion is
1276 | missing it means that SCT/Cancel RST messages are supported by the STS.

1277 | /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

1278 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the STS issuing the
1279 | secure conversation token does not support SCT/Amend RST messages. If this assertion is
1280 | missing it means that SCT/Amend RST messages are supported by the STS.

1281 | /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1282 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the STS issuing the
1283 | secure conversation token does not support SCT/Renew RST messages. If this assertion is
1284 | missing it means that SCT/Renew RST messages are supported by the STS.

1285 | /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1286 | This ~~optional~~OPTIONAL element is a policy assertion that contains the policy indicating the
1287 | requirements for obtaining the Security Context Token.

1288 | /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1289 | This element contains the security binding requirements for obtaining the Security Context Token.
1290 | It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
1291 | protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
1292 | are to be protected.

1293 **Example**

```
1294 | <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1295 |   <sp:SymmetricBinding>
1296 |     <wsp:Policy>
1297 |       <sp:ProtectionToken>
1298 |         <wsp:Policy>
1299 |           <sp:SecureConversationToken>
1300 |             <sp:Issuer>
1301 |               <wsa:Address>http://example.org/sts</wsa:Address>
1302 |             </sp:Issuer>
1303 |           </sp:Policy>
1304 |         </sp:ProtectionToken>
1305 |       </wsp:Policy>
1306 |     </sp:SymmetricBinding>
1307 |   </wsp:Policy>
1308 | </sp:SecureConversationToken>
```

```

1304         <sp:SC134SecurityContextToken />
1305         <sp:BootstrapPolicy>
1306             <wsp:Policy>
1307                 <sp:AsymmetricBinding>
1308                     <wsp:Policy>
1309                         <sp:InitiatorToken>
1310                             ...
1311                         </sp:InitiatorToken>
1312                         <sp:RecipientToken>
1313                             ...
1314                         </sp:RecipientToken>
1315                     </wsp:Policy>
1316                 </sp:AsymmetricBinding>
1317                 <sp:SignedParts>
1318                     ...
1319                 </sp:SignedParts>
1320                 ...
1321             </wsp:Policy>
1322         </sp:BootstrapPolicy>
1323     </wsp:Policy>
1324 </sp:SecureConversationToken>
1325 </wsp:Policy>
1326 </sp:ProtectionToken>
1327 ...
1328 </wsp:Policy>
1329 </sp:SymmetricBinding>
1330 <sp:SignedParts>
1331 ...
1332 </sp:SignedParts>
1333 ...
1334 </wsp:Policy>

```

5.4.8 SamlToken Assertion

This element represents a requirement for a SAML token.

Syntax

```

1338 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1339 (
1340     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1341     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1342 ) ?
1343 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1344 <wsp:Policy xmlns:wsp="...">
1345 (
1346     <sp:RequireDerivedKeys ... /> |
1347     <sp:RequireImpliedDerivedKeys ... /> |
1348     <sp:RequireExplicitDerivedKeys ... />
1349 ) ?
1350 <sp:RequireKeyIdentifierReference ... /> ?
1351 (
1352     <sp:WssSamlV11Token10 ... /> |
1353     <sp:WssSamlV11Token11 ... /> |
1354     <sp:WssSamlV20Token11 ... />
1355 ) ?
1356 ...
1357 </wsp:Policy>
1358 ...
1359 </sp:SamlToken>

```

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

1362 /sp:SamIToken
 1363 This identifies a SamIToken assertion.

1364 /sp:SamIToken/@sp:IncludeToken
 1365 This ~~optional~~OPTIONAL attribute identifies the token inclusion value for this token assertion.

1366 /sp:SamIToken/sp:Issuer
 1367 This ~~optional~~OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the
 1368 issuer of the sp:SamIToken.

1369 /sp:SamIToken/sp:IssuerName
 1370 This ~~optional~~OPTIONAL element, of type xs:anyURI, contains the logical name of the
 1371 sp:SamIToken issuer.

1372 /sp:SamIToken/wst:Claims
 1373 This ~~optional~~OPTIONAL element identifies the ~~required~~REQUIRED claims that a security token
 1374 must contain in order to satisfy the token assertion requirements.

1375 /sp:SamIToken/wsp:Policy
 1376 This ~~required~~REQUIRED element identifies additional requirements for use of the sp:SamIToken
 1377 assertion.

1378 /sp:SamIToken/wsp:Policy/sp:RequireDerivedKeys
 1379 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit
 1380 Derived Keys] and [Implied Derived Keys] properties for this token to 'true'.

1381 /sp:SamIToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1382 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit
 1383 Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this
 1384 token to 'false'.

1385 /sp:SamIToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1386 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied
 1387 Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
 1388 token to 'false'.

1389 /sp:SamIToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1390 This ~~optional~~OPTIONAL element is a policy assertion that indicates that a key identifier reference
 1391 is ~~required~~REQUIRED when referencing this token.

1392 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10
 1393 This ~~optional~~OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1
 1394 token should be used as defined in [[WSS:SAMLTokenProfile1.0](#)].

1395 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11
 1396 This ~~optional~~OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1
 1397 token should be used as defined in [[WSS:SAMLTokenProfile1.1](#)].

1398 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11
 1399 This ~~optional~~OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0
 1400 token should be used as defined in [[WSS:SAMLTokenProfile1.1](#)].

1401
 1402 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
 1403 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition

1404 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion ~~should~~
1405 SHOULD be used instead.

1406 5.4.9 RelToken Assertion

1407 This element represents a requirement for a REL token.

1408 Syntax

```
1409 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1410   (  
1411     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1412     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1413   ) ?  
1414   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1415   <wsp:Policy xmlns:wsp="...">  
1416     (  
1417       <sp:RequireDerivedKeys ... /> |  
1418       <sp:RequireImpliedDerivedKeys ... /> |  
1419       <sp:RequireExplicitDerivedKeys ... />  
1420     ) ?  
1421     <sp:RequireKeyIdentifierReference ... /> ?  
1422     (  
1423       <sp:WssRelV10Token10 ... /> |  
1424       <sp:WssRelV20Token10 ... /> |  
1425       <sp:WssRelV10Token11 ... /> |  
1426       <sp:WssRelV20Token11 ... />  
1427     ) ?  
1428     ...  
1429   </wsp:Policy>  
1430   ...  
1431 </sp:RelToken>
```

1432
1433 The following describes the attributes and elements listed in the schema outlined above:

1434 /sp:RelToken

1435 This identifies a RelToken assertion.

1436 /sp:RelToken/@sp:IncludeToken

1437 This ~~optional~~OPTIONAL attribute identifies the token inclusion value for this token assertion.

1438 /sp:RelToken/sp:Issuer

1439 This ~~optional~~OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the
1440 issuer of the sp:RelToken.

1441 /sp:RelToken/sp:IssuerName

1442 This ~~optional~~OPTIONAL element, of type xs:anyURI, contains the logical name of the
1443 sp:RelToken issuer.

1444 /sp:RelToken/wst:Claims

1445 This ~~optional~~OPTIONAL element identifies the ~~required~~REQUIRED claims that a security token
1446 must contain in order to satisfy the token assertion requirements.

1447 /sp:RelToken/wsp:Policy

1448 This ~~required~~REQUIRED element identifies additional requirements for use of the sp:RelToken
1449 assertion.

1450 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1451 This ~~optional~~OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit
1452 Derived Keys] and [Implied Derived Keys] property for this token to 'true'.

1453 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1454 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Explicit

1455 | Derived Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this

1456 | token to 'false'.

1457 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1458 | This ~~optional~~**OPTIONAL** element is a policy assertion that sets the [Derived Keys] and [Implied

1459 | Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this

1460 | token to 'false'.

1461 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1462 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that a key identifier reference

1463 | is ~~required~~**REQUIRED** when referencing this token.

1464 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1465 | This ~~optional~~**OPTIONAL** element is a policy assertion that identifies that a REL Version 1.0 token

1466 | should be used as defined in [WSS:RELTokenProfile1.0].

1467 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1468 | This ~~optional~~**OPTIONAL** element is a policy assertion that identifies that a REL Version 2.0 token

1469 | should be used as defined in [WSS:RELTokenProfile1.0].

1470 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1471 | This ~~optional~~**OPTIONAL** element is a policy assertion that identifies that a REL Version 1.0 token

1472 | should be used as defined in [WSS:RELTokenProfile1.1].

1473 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1474 | This ~~optional~~**OPTIONAL** element is a policy assertion that identifies that a REL Version 2.0 token

1475 | should be used as defined in [WSS:RELTokenProfile1.1].

1476

1477 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties

1478 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition

1479 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion ~~should~~

1480 **SHOULD** be used instead.

1481 5.4.10 HttpsToken Assertion

1482 This element represents a requirement for a transport binding to support the use of HTTPS.

1483 Syntax

```

1484 <sp:HttpsToken xmlns:sp="..." ... >
1485 (
1486   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1487   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1488 ) ?
1489 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1490 <wsp:Policy xmlns:wsp="...">
1491 (
1492   <sp:HttpBasicAuthentication /> |
1493   <sp:HttpDigestAuthentication /> |
1494   <sp:RequireClientCertificate /> |
1495   ...
1496 ) ?
1497 ...
1498 </wsp:Policy>
1499 ...
1500 </sp:HttpsToken>

```

1501 The following describes the attributes and elements listed in the schema outlined above:

1502 /sp:HttpsToken

1503 This identifies an Https assertion stating that use of the HTTPS protocol specification is

1504 supported.

1505 /sp:HttpsToken/sp:Issuer

1506 This ~~optional~~**OPTIONAL** element, of type wsa:EndpointReferenceType, contains reference to the

1507 issuer of the sp:HttpsToken.

1508 /sp:HttpsToken/sp:IssuerName

1509 This ~~optional~~**OPTIONAL** element, of type xs:anyURI, contains the logical name of the

1510 sp:HttpsToken issuer.

1511 /sp:HttpsToken/wst:Claims

1512 This ~~optional~~**OPTIONAL** element identifies the ~~required~~**REQUIRED** claims that a security token

1513 must contain in order to satisfy the token assertion requirements.

1514 /sp:HttpsToken/wsp:Policy

1515 This ~~required~~**REQUIRED** element identifies additional requirements for use of the sp:HttpsToken

1516 assertion.

1517 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1518 This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the client MUST use

1519 HTTP Basic Authentication [RFC2068] to authenticate to the service.

1520 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1521 This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the client MUST use

1522 HTTP Digest Authentication [RFC2068] to authenticate to the service.

1523 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1524 This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the client MUST provide

1525 a certificate when negotiating the HTTPS session.

1526 5.4.11 KeyValueToken Assertion

1527 This element represents a requirement for a KeyValue token. The next section defines the KeyValue

1528 security token abstraction for purposes of this token assertion.

1529 This document defines requirements for KeyValue token when used in combination with RSA

1530 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by

1531 introducing new nested assertions besides *sp:RsaKeyValue*.

1533 Syntax

```
1534 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1535   <wsp:Policy xmlns:wsp="...">
1536     <sp:RsaKeyValue ... /> ?
1537     ...
1538   </wsp:Policy>
1539   ...
1540 </sp:KeyValueToken>
```

1541 The following describes the attributes listed in the schema outlined above:

1542 /sp:KeyValueToken

1543 This identifies a RsaToken assertion.

1544 /sp:KeyValueToken/@sp:IncludeToken

1545 This ~~optional~~**OPTIONAL** attribute identifies the token inclusion value for this token assertion.

1546 /sp:KeyValueToken/wsp:Policy

1547 | This ~~required~~**REQUIRED** element identifies additional requirements for use of the
1548 sp:KeyValueToken assertion.

1549 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1550 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the ds:RSAKeyValue
1551 element must be present in the KeyValue token. This indicates that an RSA key pair must be
1552 used.

1553 5.4.11.1 KeyValue Token

1554 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key
1555 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this
1556 section is to define the KeyValue token abstraction that represents such key pair referencing mechanism.

1557
1558 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be
1559 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*
1560 element in combination with RSA cryptographic algorithm.

1561
1562 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the
1563 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1564 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1565   <ds:KeyValue>  
1566     <ds:RSAKeyValue>  
1567       <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1568       <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1569     </ds:RSAKeyValue>  
1570   </ds:KeyValue>  
1571 </ds:KeyInfo>
```

1572
1573 When the KeyValue token is used the corresponding public key value appears directly in the signature or
1574 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValue token
1575 manifestation outside the *ds:KeyInfo* element.

```
1576 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1577   <SignedInfo>  
1578     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1579 c14n#" />  
1580     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1581     <Reference URI="#_1">  
1582       <Transforms>  
1583         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1584       </Transforms>  
1585       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
1586       <DigestValue>...</DigestValue>  
1587     </Reference>  
1588   </SignedInfo>  
1589   <SignatureValue>...</SignatureValue>  
1590   <KeyInfo>  
1591     <KeyValue>  
1592       <RSAKeyValue>  
1593         <Modulus>...</Modulus>  
1594         <Exponent>...</Exponent>  
1595       </RSAKeyValue>  
1596     </KeyValue>  
1597   </KeyInfo>  
1598 </Signature>
```

1599
1600 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no
1601 identifier can be associated with the token, the KeyValue token cannot be referenced by using

1602 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the *KeyValue*
1603 token can be used whenever a security token can be used as illustrated on the following example:

```
1604 <t:RequestSecurityToken xmlns:t="...">  
1605   <t:RequestType>...</t:RequestType>  
1606   ...  
1607   <t:UseKey>  
1608     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1609       <KeyValue>  
1610         <RSAKeyValue>  
1611           <Modulus>...</Modulus>  
1612           <Exponent>...</Exponent>  
1613         </RSAKeyValue>  
1614       </KeyValue>  
1615     </KeyInfo>  
1616   </t:UseKey>  
1617 </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~~**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>

1654

1655 The tables below show all the base algorithm suites defined by this specification. This table defines
 1656 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

1657 This table defines additional properties whose values can be specified along with the default value for that
 1658 property.

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

1659 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~~**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~~**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~~**NOT REQUIRED** to be encrypted if the value is 'true' when there is nothing ~~else~~ 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~~**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~~**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~~ **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 ~~recommended~~ **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 wsuse wsuse :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 wsuse wsuse :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~~ **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.

- 1712 b. A Username token (usually in encrypted form) MUST occur before the signature that
1713 signs it.
- 1714 c. A primary signature MUST occur before the supporting token signature that signs the
1715 primary signature's signature value element.
- 1716 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1717 has the same order requirements as the source plain text element, unless requirement 4
1718 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1719 supporting token signature per 2.c above and an encrypted token has the same ordering
1720 requirements as the unencrypted token.
- 1721 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1722 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1723 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1724 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1725 Layout Rules for WSS 1.1
- 1726 1. Tokens that are included in the message MUST be declared before use. For example:
- 1727 a. A local signing token MUST occur before the signature that uses it.
- 1728 b. A local token serving as the source token for a derived key token MUST occur before that
1729 derived key token.
- 1730 c. A local encryption token MUST occur before the reference list that points to
1731 xenc:EncryptedData elements that use it.
- 1732 d. If the same token is used for both signing and encryption, then it ~~should~~ **SHOULD** appear
1733 before the ds:Signature and xenc:ReferenceList elements in the security header that are
1734 generated using the token.
- 1735 2. Signed elements inside the security header MUST occur before the signature that signs them.
1736 For example:
- 1737 a. A timestamp MUST occur before the signature that signs it.
- 1738 b. A Username token (usually in encrypted form) MUST occur before the signature that
1739 signs it.
- 1740 c. A primary signature MUST occur before the supporting token signature that signs the
1741 primary signature's signature value element.
- 1742 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1743 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1744 has the same order requirements as the source plain text element, unless requirement 4
1745 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1746 supporting token signature per 2.c above and an encrypted token has the same ordering
1747 requirements as the unencrypted token.
- 1748 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1749 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1750 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1751 However, the xenc:ReferenceList is ~~not required~~ **NOT REQUIRED** to appear before independently
1752 encrypted tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1753 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1754 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~~**REQUIRED** element contains one or more policy assertions that indicate the specific algorithm suite to use.

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

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

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

1804 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1805 | property is set to 'Basic192'.

1806 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1807 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1808 | property is set to 'Basic128'.

1809 | /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1810 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1811 | property is set to 'TripleDes'.

1812 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1813 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1814 | property is set to 'Basic256Rsa15'.

1815 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1816 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1817 | property is set to 'Basic192Rsa15'.

1818 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1819 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1820 | property is set to 'Basic128Rsa15'.

1821 | /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1822 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1823 | property is set to 'TripleDesRsa15'.

1824 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1825 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1826 | property is set to 'Basic256Sha256'.

1827 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1828 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1829 | property is set to 'Basic192Sha256'.

1830 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1831 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1832 | property is set to 'Basic128Sha256'.

1833 | /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1834 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1835 | property is set to 'TripleDesSha256'.

1836 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1837 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1838 | property is set to 'Basic256Sha256Rsa15'.

1839 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1840 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1841 | property is set to 'Basic192Sha256Rsa15'.

1842 | /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1843 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite]
1844 | property is set to 'Basic128Sha256Rsa15'.

1845 | /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1846 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Algorithm Suite]
 1847 | property is set to 'TripleDesSha256Rsa15'.

1848 | /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1849 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [C14N] property of an
 1850 | algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]
 1851 | property is 'ExcC14N'.

1852 | /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1853 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [SOAP Norm]
 1854 | property is set to 'SNT'.

1855 | /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1856 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [STR Transform]
 1857 | property is set to 'STRT10'.

1858 | /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1859 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [XPath] property is
 1860 | set to 'XPath'.

1861 | /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1862 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [XPath] property is
 1863 | set to 'XPath20'.

1864 | /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1865 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [XPath] property is
 1866 | set to 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1867 |

1868 7.2 Layout Assertion

1869 | This assertion indicates a requirement for a particular security header layout as defined under the
 1870 | [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
 1871 | containing assertion.

1872 Syntax

```
1873 <sp:Layout xmlns:sp="..." ... >
1874   <wsp:Policy xmlns:wsp="...">
1875     <sp:Strict ... /> |
1876     <sp:Lax ... /> |
1877     <sp:LaxTsFirst ... /> |
1878     <sp:LaxTsLast ... /> |
1879     ...
1880   </wsp:Policy>
1881   ...
1882 </sp:Layout>
```

1883 |

1884 | The following describes the attributes and elements listed in the schema outlined above:

1885 | /sp:Layout

1886 | This identifies a Layout assertion.

1887 | /sp:Layout/wsp:Policy

1888 | This ~~required~~**REQUIRED** element contains one or more policy assertions that indicate the specific
 1889 | security header layout to use.

1890 | /sp:Layout/wsp:Policy/sp:Strict

1891 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Security Header
1892 Layout] property is set to 'Strict'.
1893 /sp:Layout/wsp:Policy/sp:Lax
1894 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Security Header
1895 Layout] property is set to 'Lax'.
1896 /sp:Layout/wsp:Policy/sp:LaxTsFirst
1897 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Security Header
1898 Layout] property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be
1899 set to 'true' by the presence of an sp:IncludeTimestamp assertion.
1900 /sp:Layout/wsp:Policy/sp:LaxTsLast
1901 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Security Header
1902 Layout] property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be
1903 set to 'true' by the presence of an sp:IncludeTimestamp assertion.

1904 7.3 TransportBinding Assertion

1905 The TransportBinding assertion is used in scenarios in which message protection and security correlation
1906 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
1907 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
1908 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
1909 MUST apply to [Endpoint Policy Subject].

1910 Syntax

```
1911 <sp:TransportBinding xmlns:sp="..." ... >
1912   <wsp:Policy xmlns:wsp="...">
1913     <sp:TransportToken ... >
1914       <wsp:Policy> ... </wsp:Policy>
1915     ...
1916   </sp:TransportToken>
1917   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1918   <sp:Layout ... > ... </sp:Layout> ?
1919   <sp:IncludeTimestamp ... /> ?
1920   ...
1921 </wsp:Policy>
1922 ...
1923 </sp:TransportBinding>
```

1924
1925 The following describes the attributes and elements listed in the schema outlined above:

1926 /sp:TransportBinding

1927 This identifies a TransportBinding assertion.

1928 /sp:TransportBinding/wsp:Policy

1929 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
1930 assertion.

1931 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1932 | This ~~required~~**REQUIRED** element is a policy assertion that indicates a requirement for a
1933 Transport Token. The specified token populates the [Transport Token] property and indicates
1934 how the transport is secured.

1935 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1936 This indicates a nested policy that identifies the type of Transport Token to use.

1937 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

1938 | This ~~required~~**REQUIRED** element is a policy assertion that indicates a value that populates the

1939 [Algorithm Suite] property. See Section 6.1 for more details.

1940 /sp:TransportBinding/wsp:Policy/sp:Layout

1941 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates a value that populates the

1942 [Security Header Layout] property. See Section 6.7 for more details.

1943 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

1944 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Timestamp] property

1945 is set to 'true'.

1946 7.4 SymmetricBinding Assertion

1947 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means

1948 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;

1949 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this

1950 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to

1951 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to

1952 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token

1953 properties and is used as the basis for both encryption and signature in both directions. This assertion

1954 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

1955 Syntax

```

1956 <sp:SymmetricBinding xmlns:sp="..." ... >
1957   <wsp:Policy xmlns:wsp="...">
1958     (
1959       <sp:EncryptionToken ... >
1960         <wsp:Policy> ... </wsp:Policy>
1961       </sp:EncryptionToken>
1962       <sp:SignatureToken ... >
1963         <wsp:Policy> ... </wsp:Policy>
1964       </sp:SignatureToken>
1965     ) | (
1966       <sp:ProtectionToken ... >
1967         <wsp:Policy> ... </wsp:Policy>
1968       </sp:ProtectionToken>
1969     )
1970     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1971     <sp:Layout ... > ... </sp:Layout> ?
1972     <sp:IncludeTimestamp ... /> ?
1973     <sp:EncryptBeforeSigning ... /> ?
1974     <sp:EncryptSignature ... /> ?
1975     <sp:ProtectTokens ... /> ?
1976     <sp:OnlySignEntireHeadersAndBody ... /> ?
1977     ...
1978   </wsp:Policy>
1979   ...
1980 </sp:SymmetricBinding>

```

1981

1982 The following describes the attributes and elements listed in the schema outlined above:

1983 /sp:SymmetricBinding

1984 This identifies a SymmetricBinding assertion.

1985 /sp:SymmetricBinding/wsp:Policy

1986 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding

1987 assertion.

1988	/sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
1989	This optional OPTIONAL element is a policy assertion that indicates a requirement for an Encryption Token. The specified token populates the [Encryption Token] property and is used for encryption. It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.
1990	
1991	
1992	
1993	/sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
1994	The policy contained here MUST identify exactly one token to use for encryption.
1995	/sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
1996	This optional OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token. The specified token populates the [Signature Token] property and is used for the message signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be specified.
1997	
1998	
1999	
2000	/sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
2001	The policy contained here MUST identify exactly one token to use for signatures.
2002	/sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
2003	This optional OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token. The specified token populates the [Encryption Token] and [Signature Token] properties] and is used for the message signature and for encryption. It is an error for both an sp:ProtectionToken assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be specified.
2004	
2005	
2006	
2007	
2008	/sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
2009	The policy contained here MUST identify exactly one token to use for protection.
2010	/sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
2011	This required REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm Suite] property. See Section 6.1 for more details.
2012	
2013	/sp:SymmetricBinding/wsp:Policy/sp:Layout
2014	This optional OPTIONAL element is a policy assertion that indicates a value that populates the [Security Header Layout] property. See Section 6.7 for more details.
2015	
2016	/sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
2017	This optional OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set to 'true'.
2018	
2019	/sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
2020	This optional OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is set to 'EncryptBeforeSigning'.
2021	
2022	/sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
2023	This optional OPTIONAL element is a policy assertion that indicates that the [Signature Protection] property is set to 'true'.
2024	
2025	/sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
2026	This optional OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is set to 'true'.
2027	
2028	/sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
2029	This optional OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body Signatures] property is set to 'true'.
2030	

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>
```

```

2083     </sp:RecipientEncryptionToken>
2084   )
2085   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2086   <sp:Layout ... > ... </sp:Layout> ?
2087   <sp:IncludeTimestamp ... /> ?
2088   <sp:EncryptBeforeSigning ... /> ?
2089   <sp:EncryptSignature ... /> ?
2090   <sp:ProtectTokens ... /> ?
2091   <sp:OnlySignEntireHeadersAndBody ... /> ?
2092   ...
2093 </wsp:Policy>
2094   ...
2095 </sp:AsymmetricBinding>

```

2096

2097 The following describes the attributes and elements listed in the schema outlined above:

2098 /sp:AsymmetricBinding

2099 This identifies a AsymmetricBinding assertion.

2100 /sp:AsymmetricBinding/wsp:Policy

2101 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
 2102 assertion.

2103 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2104 | This ~~optional~~OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
 2105 Token. The specified token populates the [Initiator Signature Token] and [Initiator Encryption
 2106 Token] properties and is used for the message signature from initiator to recipient, and encryption
 2107 from recipient to initiator.

2108 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2109 The policy contained here MUST identify one or more token assertions.

2110 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2111 | This ~~optional~~OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
 2112 Signature Token. The specified token populates the [Initiator Signature Token] property and is
 2113 used for the message signature from initiator to recipient.

2114 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2115 The policy contained here MUST identify one or more token assertions.

2116 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2117 | This ~~optional~~OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
 2118 Encryption Token. The specified token populates the [Initiator Encryption Token] property and is
 2119 used for the message encryption from recipient to initiator.

2120 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2121 The policy contained here MUST identify one or more token assertions.

2122 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2123 | This ~~optional~~OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
 2124 Token. The specified token populates the [Recipient Signature Token] and [Recipient Encryption
 2125 Token] property and is used for encryption from initiator to recipient, and for the message
 2126 signature from recipient to initiator.

2127 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2128 The policy contained here MUST identify one or more token assertions.

2129 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2130 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates a requirement for a Recipient
 2131 | Signature Token. The specified token populates the [Recipient Signature Token] property and is
 2132 | used for the message signature from ~~R~~recipient to ~~recipient~~**initiator**.

2133 | /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
 2134 | The policy contained here MUST identify one or more token assertions.

2135 | /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
 2136 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates a requirement for a Recipient
 2137 | Encryption Token. The specified token populates the [Recipient Encryption Token] property and
 2138 | is used for the message encryption from ~~recipient~~**initiator** to ~~R~~recipient.

2139 | /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
 2140 | The policy contained here MUST identify one or more token assertions.

2141 | /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2142 | This ~~required~~**REQUIRED** element is a policy assertion that indicates a value that populates the
 2143 | [Algorithm Suite] property. See Section 6.1 for more details.

2144 | /sp:AsymmetricBinding/wsp:Policy/sp:Layout
 2145 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates a value that populates the
 2146 | [Security Header Layout] property. See Section 6.7 for more details.

2147 | /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 2148 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Timestamp] property
 2149 | is set to 'true'.

2150 | /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 2151 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Protection Order]
 2152 | property is set to 'EncryptBeforeSigning'.

2153 | /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
 2154 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Signature
 2155 | Protection] property is set to 'true'.

2156 | /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
 2157 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Token Protection]
 2158 | property is set to 'true'.

2159 | /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 2160 | This ~~optional~~**OPTIONAL** element is a policy assertion that indicates that the [Entire Header And
 2161 | Body Signatures] property is set to 'true'.

8 Supporting Tokens

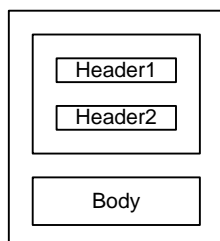
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. In case of Transport Binding the message is signed outside of the message XML by the underlying transport protocol and the signature itself is not part of the message. Additional tokens ~~may~~**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~~**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~~**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~~**SHOULD** merge the requirements by including all tokens from the outer scope and any additional tokens for a specific message from the inner scope.

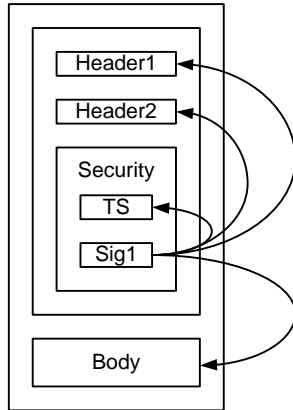
In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens ~~should~~**SHOULD** sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine 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~~**MAY** be bound to the message, let’s consider a message with three components: Header1, Header2, and Body.

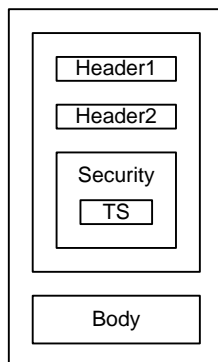


Even before any supporting tokens are added, each binding requires that the message is signed using a token satisfying the **required REQUIRED** usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection.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



Note: if **required 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 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> |
```

2227
2228
2229
2230
2231
2232
2233
2234

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

2235

2236 The following describes the attributes and elements listed in the schema outlined above:

2237 /sp:SupportingTokens

2238 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2239 Tokens] property.

2240 /sp:SupportingTokens/wsp:Policy

2241 This describes additional requirements for satisfying the SupportingTokens assertion.

2242 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2243 The policy MUST identify one or more token assertions.

2244 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2245 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2246 7.1 and describes the algorithms to use for cryptographic operations performed with the tokens
2247 identified by this policy assertion.

2248 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2249 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2250 4.1.1 and describes additional message parts that MUST be included in the signature generated
2251 with the token identified by this policy assertion.

2252 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2253 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2254 4.1.2 and describes additional message elements that MUST be included in the signature
2255 generated with the token identified by this policy assertion.

2256 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2257 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2258 4.2.1 and describes additional message parts that MUST be encrypted using the token identified
2259 by this policy assertion.

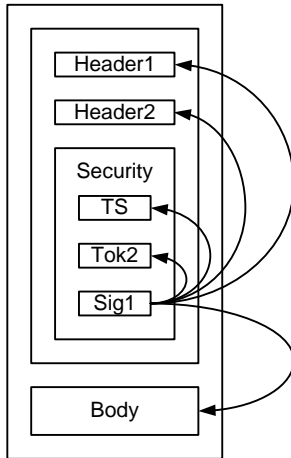
2260 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2261 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2262 4.2.2 and describes additional message elements that MUST be encrypted using the token
2263 identified by this policy assertion.

2264 8.2 SignedSupportingTokens Assertion

2265 Signed tokens are included in the “message signature” as defined above and ~~may optionally~~MAY
2266 OPTIONALLY include additional message parts to sign and/or encrypt. The diagram below illustrates how
2267 the attached token (Tok2) is signed by the message signature (Sig1):

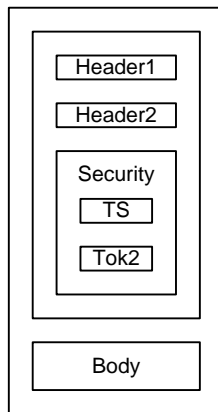
2268



2269

2270 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2271



2272

2273 Syntax

2274

2275

2276

2277

2278

2279

2280

2281

2282

2283

2284

2285

2286

2287

```
<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>
```

2288

2289 The following describes the attributes and elements listed in the schema outlined above:

2290 /sp:SignedSupportingTokens

2291 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed
2292 Supporting Tokens] property.

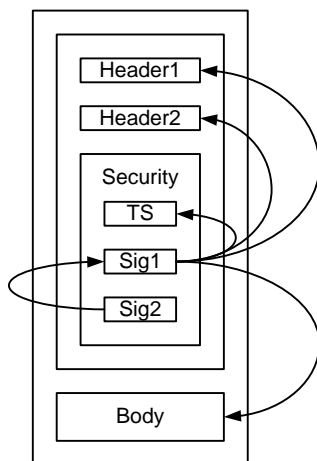
2293 /sp:SignedSupportingTokens/wsp:Policy

2294 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

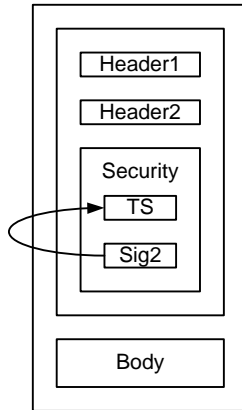
2295 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 2296 The policy MUST identify one or more token assertions.
 2297 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 2298 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
 2299 7.1 and describes the algorithms to use for cryptographic operations performed with the tokens
 2300 identified by this policy assertion.
 2301 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 2302 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
 2303 4.1.1 and describes additional message parts that MUST be included in the signature generated
 2304 with the token identified by this policy assertion.
 2305 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 2306 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
 2307 4.1.2 and describes additional message elements that MUST be included in the signature
 2308 generated with the token identified by this policy assertion.
 2309 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 2310 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
 2311 4.2.1 and describes additional message parts that MUST be encrypted using the token identified
 2312 by this policy assertion.
 2313 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 2314 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
 2315 4.2.2 and describes additional message elements that MUST be encrypted using the token
 2316 identified by this policy assertion.

2317 8.3 EndorsingSupportingTokens Assertion

2318 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 2319 | produced from the message signature and ~~may optionally~~MAY OPTIONALLY include additional message
 2320 parts to sign and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the
 2321 message signature (Sig1):
 2322



2323
 2324 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 2325 below:
 2326



Syntax

```

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

```

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

/sp:EndorsingSupportingTokens

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

/sp:EndorsingSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

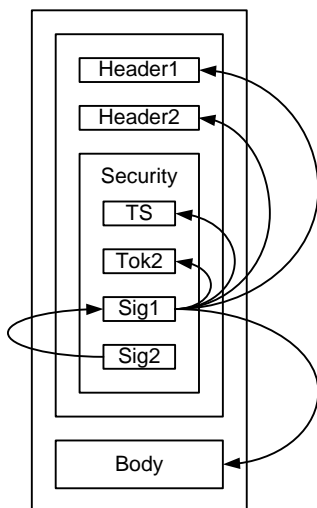
This ~~optional~~**OPTIONAL** element is a policy assertion that follows the schema outlined in Section 4.1.2 and describes additional message elements that MUST be included in the signature generated with the token identified by this policy assertion.

2364 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
 2365 | This ~~optional~~**OPTIONAL** element is a policy assertion that follows the schema outlined in Section
 2366 4.2.1 and describes additional message parts that **MUST** be encrypted using the token identified
 2367 by this policy assertion.

2368 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
 2369 | This ~~optional~~**OPTIONAL** element is a policy assertion that follows the schema outlined in Section
 2370 4.2.2 and describes additional message elements that **MUST** be encrypted using the token
 2371 identified by this policy assertion.

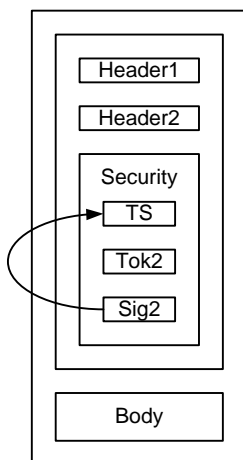
2372 8.4 SignedEndorsingSupportingTokens Assertion

2373 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
 2374 and are themselves signed by that message signature, that is both tokens (the token used for the
 2375 message signature and the signed endorsing token) sign each other. This assertion ~~may optionally~~**MAY**
 2376 **OPTIONALLY** include additional message parts to sign and/or encrypt. The diagram below illustrates how
 2377 the signed token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2)
 2378 signs the message signature (Sig1):
 2379



2380

2381 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
 2382 ~~should~~**SHOULD** cover the message timestamp as illustrated below:
 2383



2385 Syntax

```
2386 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2387   <wsp:Policy xmlns:wsp="...">
2388     [Token Assertion]+
2389     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2390     (
2391       <sp:SignedParts ... > ... </sp:SignedParts> |
2392       <sp:SignedElements ... > ... </sp:SignedElements> |
2393       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2394       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2395     ) *
2396     ...
2397   </wsp:Policy>
2398   ...
2399 </sp:SignedEndorsingSupportingTokens>
```

2400

2401 The following describes the attributes and elements listed in the schema outlined above:

2402 /sp:SignedEndorsingSupportingTokens

2403 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the
2404 [Signed Endorsing Supporting Tokens] property.

2405 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2406 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2407 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2408 The policy MUST identify one or more token assertions.

2409 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2410 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2411 7.1 and describes the algorithms to use for cryptographic operations performed with the tokens
2412 identified by this policy assertion.

2413 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2414 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2415 4.1.1 and describes additional message parts that MUST be included in the signature generated
2416 with the token identified by this policy assertion.

2417 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2418 | This ~~optional~~OPTIONAL element follows the schema outlined in Section 4.1.2 and describes
2419 additional message elements that MUST be included in the signature generated with the token
2420 identified by this policy assertion.

2421 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2422 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2423 4.2.1 and describes additional message parts that MUST be encrypted using the token identified
2424 by this policy assertion.

2425 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2426 | This ~~optional~~OPTIONAL element is a policy assertion that follows the schema outlined in Section
2427 4.2.2 and describes additional message elements that MUST be encrypted using the token
2428 identified by this policy assertion.

8.5 SignedEncryptedSupportingTokens Assertion

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

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

8.6 EncryptedSupportingTokens Assertion

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

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

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

8.7 EndorsingEncryptedSupportingTokens Assertion

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

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

8.8 SignedEndorsingEncryptedSupportingTokens Assertion

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

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

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

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

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

- 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~~ **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~~ **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~~ 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'.

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>
```

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

2609 /sp:Wss10
 2610 This identifies a WSS10 assertion.
 2611 /sp:Wss10/wsp:Policy
 2612 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.
 2613 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2614 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Key Identifier
 2615 References] property is set to 'true'.
 2616 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial
 2617 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
 2618 property is set to 'true'.
 2619 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI
 2620 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [External URI
 2621 References] property is set to 'true'.
 2622 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken
 2623 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Embedded Token
 2624 References] property is set to 'true'.

2625 9.2 Wss11 Assertion

2626 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
 2627 supported.

2628 Syntax

```

2629 <sp:Wss11 xmlns:sp="..." ... >
2630   <wsp:Policy xmlns:wsp="...">
2631     <sp:MustSupportRefKeyIdentifier ... /> ?
2632     <sp:MustSupportRefIssuerSerial ... /> ?
2633     <sp:MustSupportRefExternalURI ... /> ?
2634     <sp:MustSupportRefEmbeddedToken ... /> ?
2635     <sp:MustSupportRefThumbprint ... /> ?
2636     <sp:MustSupportRefEncryptedKey ... /> ?
2637     <sp:RequireSignatureConfirmation ... /> ?
2638     ...
2639   </wsp:Policy>
2640 </sp:Wss11>
  
```

2641
 2642 The following describes the attributes and elements listed in the schema outlined above:

2643 /sp:Wss11
 2644 This identifies an WSS11 assertion.
 2645 /sp:Wss11/wsp:Policy
 2646 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.
 2647 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2648 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Key Identifier
 2649 References] property is set to 'true'.
 2650 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial
 2651 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
 2652 property is set to 'true'.

2653 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI

2654 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [External URI

2655 | References] property is set to 'true'.

2656 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken

2657 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Embedded Token

2658 | References] property is set to 'true'.

2659 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint

2660 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Thumbprint References]

2661 | property is set to 'true'.

2662 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey

2663 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [EncryptedKey

2664 | References] property is set to 'true'.

2665 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation

2666 | This ~~optional~~OPTIONAL element is a policy assertion indicates that the [Signature Confirmation]

2667 | property 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~~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 1.3 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~~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~~REQUIRED to be used as key material for a requested proof token. A value of 'true' indicates that client entropy is ~~required~~REQUIRED. A value of 'false' indicates that client entropy is ~~not required~~NOT REQUIRED. This property has a default value of 'false'.

[Server Entropy]

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

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 />?
    ...
  </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.

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

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

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

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

/sp:Trust13/wsp:Policy/sp:RequireClientEntropy

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

/sp:Trust13/wsp:Policy/sp:RequireServerEntropy

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

/sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens

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

/sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2756 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the [Collection] property
2757 | is set to 'true'.

2758 | /sp:Trust1~~39~~/wsp:Policy/sp:RequireAppliesTo

2759 | This ~~optional~~OPTIONAL element is a policy assertion that indicates that the STS requires the
2760 | requestor to specify the scope for the issued token using wsp:AppliesTo in the RST.

11 Guidance on creating new assertions and assertion extensibility

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

11.1 General Design Points

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

11.2 Detailed Design Guidance

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

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

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

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

then design 1. would generally be 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

2805 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2806 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2807 was encoded into the assertion QNames, each existing token assertion would require five variants, one
2808 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2809
2810 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2811 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2812 assertion is parameterized by the nested token version assertions. Policy matching can use these
2813 parameters to find matches between policies where the broad token type is support by both parties but
2814 they might not support the same specific versions.

2815
2816 Note, when designing assertions for new token types such assertions SHOULD allow the
2817 sp:IncludeToken attribute and SHOULD allow nested policy.

2818

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.

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)

2868	EndorsingSupportingTokens Assertion	(Section 8.3)
2869	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2870	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2871	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2872	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2873 **A.3 Message Policy Subject Assertions**

2874 **A.3.1 Supporting Token Assertions**

2875	SupportingTokens Assertion	(Section 8.1)
2876	SignedSupportingTokens Assertion	(Section 8.2)
2877	EndorsingSupportingTokens Assertion	(Section 8.3)
2878	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
2879	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
2880	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
2881	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

2882 **A.3.2 Protection Assertions**

2883	SignedParts Assertion	(Section 4.1.1)
2884	SignedElements Assertion	(Section 4.1.2)
2885	EncryptedParts Assertion	(Section 4.2.1)
2886	EncryptedElements Assertion	(Section 4.2.2)
2887	ContentEncryptedElements Assertion	(Section 4.2.3)
2888	RequiredElements Assertion	(Section 4.3.1)
2889	RequiredParts Assertion	(Section 4.3.2)

2890 **A.4 Assertions With Undefined Policy Subject**

2891 The assertions listed in this section do not have a defined policy subject because they appear nested
 2892 inside some other assertion which does have a defined policy subject. This list is derived from nested
 2893 assertions in the specification that have independent sections. It is not a complete list of nested
 2894 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
 2895 additional nested assertions.

2896 **A.4.1 General Assertions**

2897	AlgorithmSuite Assertion	(Section 7.1)
2898	Layout Assertion	(Section 7.2)

2899 **A.4.2 Token Usage Assertions**

2900 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
 2901 assertions.

2902 **A.4.3 Token Assertions**

2903	UsernameToken Assertion	(Section 5.3.1)
------	---	-----------------

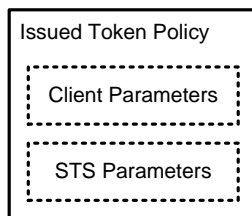
2904	IssuedToken Assertion	(Section 5.3.2)
2905	X509Token Assertion	(Section 5.3.3)
2906	KerberosToken Assertion	(Section 5.3.4)
2907	SpnegoContextToken Assertion	(Section 5.3.5)
2908	SecurityContextToken Assertion	(Section 5.3.6)
2909	SecureConversationToken Assertion	(Section 5.3.7)
2910	SamlToken Assertion	(Section 5.3.8)
2911	RelToken Assertion	(Section 5.3.9)
2912	HttpsToken Assertion	(Section 5.3.10)

B. Issued Token Policy

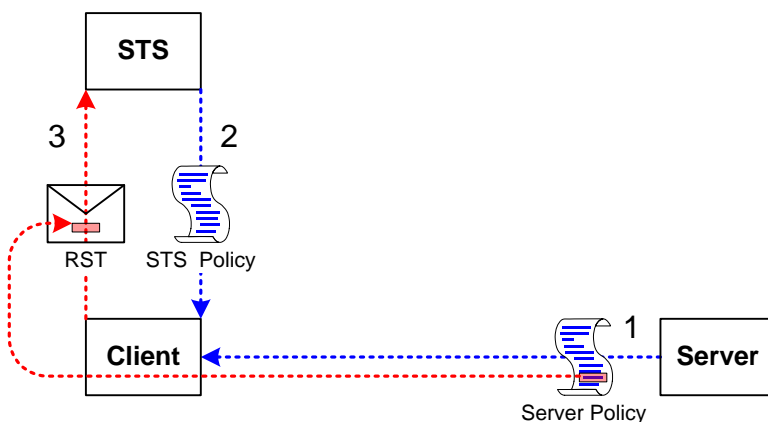
The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

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-MAY** be embedded inside an Issued Token assertion, or acquired out-of-band. There **may-MAY** be an explicit trust relationship between the Server and the STS. There **must-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-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-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.

2940 | The Issued Token Policy Assertion contains elements which ~~must~~**MUST** be understood by the Client.
2941 | The assertion contains one element which contains a list of arbitrary elements which ~~should~~**SHOULD** be
2942 | sent along to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of
2943 | the RST request sent by the Client to the STS following the protocol defined in WS-Trust.
2944 |
2945 | Elements inside the `sp:RequestSecurityTokenTemplate` element **MUST** conform to WS-Trust [WS-
2946 | Trust]. All items are ~~optional~~**OPTIONAL**, since the Server and STS may already have a pre-arranged
2947 | relationship 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>
```

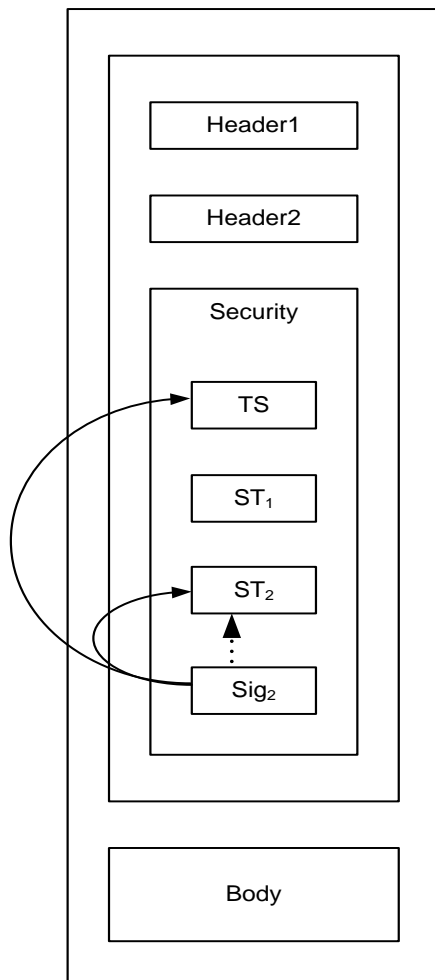
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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

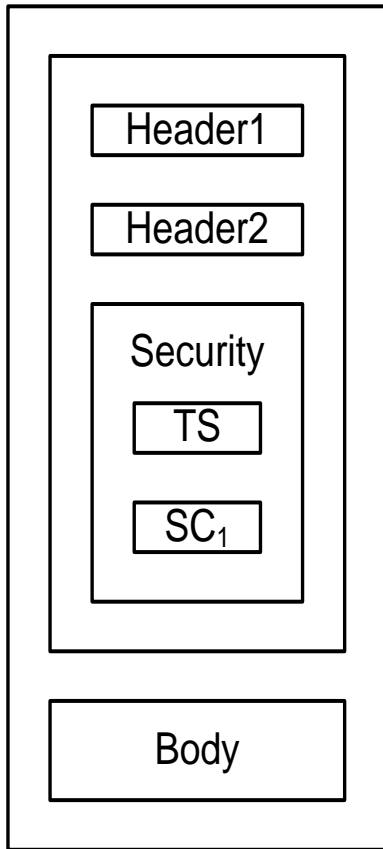
```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken 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:



3071

3072 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 3073 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 3074 message illustrated previously is included. In general, the ordering of the items in the security header
 3075 follows the most optimal layout for a receiver to process its contents.

3076 *Example:*

3077 Recipient to initiator message

```

3078 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3079   <S:Header>
3080     ...
3081     <wsse:Security>
3082       <wsu:Timestamp wsu:Id="timestamp">
3083         <wsu:Created>[datetime]</wsu:Created>
3084         <wsu:Expires>[datetime]</wsu:Expires>
3085       </wsu:Timestamp>
3086       <wsse11:SignatureConfirmation Value="..." />
3087     ...
3088   </wsse:Security>
3089   ...
3090 </S:Header>
3091 <S:Body>
3092   ...
3093 </S:Body>
3094 </S:Envelope>

```

3095 C.2 Symmetric Binding

3096 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

3097 C.2.1 Policy

3098 The following example shows a policy indicating a Symmetric Binding, a symmetric key based
3099 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message
3100 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in
3101 the message signature and the supporting signatures, a username token attached to the message, and
3102 finally an X509 token attached to the message and endorsing the message signature. Minimum message
3103 protection requirements are described as well.

```
3104 <!-- Example Endpoint Policy -->
3105 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3106   <sp:SymmetricBinding>
3107     <wsp:Policy>
3108       <sp:ProtectionToken>
3109         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3110           <sp:Issuer>...</sp:Issuer>
3111           <sp:RequestSecurityTokenTemplate>
3112             ...
3113           </sp:RequestSecurityTokenTemplate>
3114         </sp:IssuedToken>
3115       </sp:ProtectionToken>
3116       <sp:AlgorithmSuite>
3117         <wsp:Policy>
3118           <sp:Basic256 />
3119         </wsp:Policy>
3120       </sp:AlgorithmSuite>
3121       <sp:Layout>
3122         <wsp:Policy>
3123           <sp:Strict />
3124         </wsp:Policy>
3125       </sp:Layout>
3126       <sp:IncludeTimestamp />
3127       <sp:EncryptBeforeSigning />
3128       <sp:EncryptSignature />
3129       <sp:ProtectTokens />
3130     </wsp:Policy>
3131   </sp:SymmetricBinding>
3132   <sp:SignedSupportingTokens>
3133     <wsp:Policy>
3134       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3135     </wsp:Policy>
3136   </sp:SignedSupportingTokens>
3137   <sp:SignedEndorsingSupportingTokens>
3138     <wsp:Policy>
3139       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3140         <wsp:Policy>
3141           <sp:WssX509v3Token10 />
3142         </wsp:Policy>
3143       </sp:X509Token>
3144     </wsp:Policy>
3145   </sp:SignedEndorsingSupportingTokens>
3146   <sp:Wss11>
3147     <wsp:Policy>
3148       <sp:RequireSignatureConfirmation />
3149     </wsp:Policy>
3150   </sp:Wss11>
3151 </wsp:Policy>
3152
```

```

3153 <!-- Example Message Policy -->
3154 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3155   <sp:SignedParts>
3156     <sp:Header Name="Header1" Namespace="..." />
3157     <sp:Header Name="Header2" Namespace="..." />
3158     <sp:Body/>
3159   </sp:SignedParts>
3160   <sp:EncryptedParts>
3161     <sp:Header Name="Header2" Namespace="..." />
3162     <sp:Body/>
3163   </sp:EncryptedParts>
3164 </wsp:Policy>
3165

```

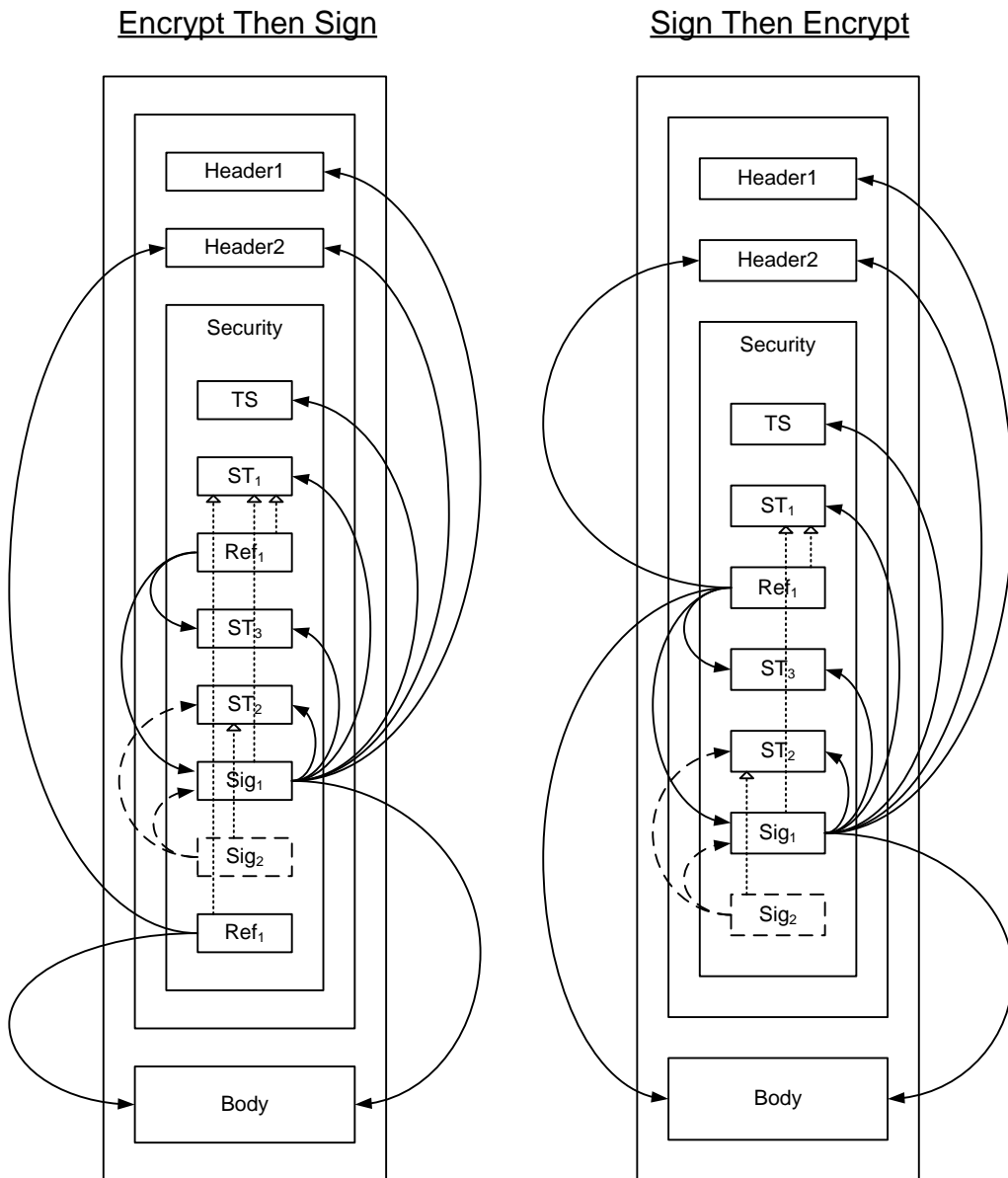
3166 This policy is used as the basis for the examples shown in the subsequent section describing the security
3167 header layout for this binding.

3168 C.2.2 Initiator to Recipient Messages

3169 Messages sent from initiator to recipient have the following layout for the security header:

- 3170 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3171 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
3172 `.../IncludeToken/Always`, then the [Encryption Token].
- 3173 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3174 Derived Key Token is used for encryption.
- 3175 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3176 reference list MUST include a reference to the message signature. If [Protection Order] is
3177 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3178 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3179 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3180 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3181 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
3182 `.../IncludeToken/Always`.
- 3183 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3184 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
3185 [Signature Token].
- 3186 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3187 Derived Key Token is used for signature.
- 3188 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3189 whether they are included in the message, and any message parts specified in SignedParts
3190 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3191 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3192 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3193 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3194 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3195 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3196 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3197 endorsing token, appears before the signature.
- 3198 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3199 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3200 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3201 above.

3202
 3203 The following diagram illustrates the security header layout for the initiator to recipient message:



3204
 3205 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 3206 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
 3207 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
 3208 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
 3209 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
 3210 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
 3211 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
 3212 process its contents.
 3213 *Example:*
 3214 Initiator to recipient message using EncryptBeforeSigning:

```
3215 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3216   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3217   xmlns:xenc="..." xmlns:ds="...">
3218   <S:Header>
3219     <x:Header1 wsu:Id="Header1" >
3220       ...
3221     </x:Header1>
3222
```

```

3223 <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3224 <!-- Plaintext Header2
3225 <x:Header2 wsu:Id="Header2" >
3226 ...
3227 </x:Header2>
3228 -->
3229 ...
3230 </wsse11:EncryptedHeader>
3231 ...
3232 <wsse:Security>
3233 <wsu:Timestamp wsu:Id="Timestamp">
3234 <wsu:Created>...</wsu:Created>
3235 <wsu:Expires>...</wsu:Expires>
3236 </wsu:Timestamp>
3237 <saml:Assertion AssertionId="_SharedSecretToken" ...>
3238 ...
3239 </saml:Assertion>
3240 <xenc:ReferenceList>
3241 <xenc:DataReference URI="#enc_Signature" />
3242 <xenc:DataReference URI="#enc_SomeUsernameToken" />
3243 ...
3244 </xenc:ReferenceList>
3245 <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3246 <!-- Plaintext UsernameToken
3247 <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3248 ...
3249 </wsse:UsernameToken>
3250 -->
3251 ...
3252 <ds:KeyInfo>
3253 <wsse:SecurityTokenReference>
3254 <wsse:Reference URI="#_SharedSecretToken" />
3255 </wsse:SecurityTokenReference>
3256 </ds:KeyInfo>
3257 </xenc:EncryptedData>
3258 <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3259 ...
3260 </wsse:BinarySecurityToken>
3261 <xenc:EncryptedData ID="enc_Signature">
3262 <!-- Plaintext Signature
3263 <ds:Signature Id="Signature">
3264 <ds:SignedInfo>
3265 <ds:References>
3266 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3267 <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3268 <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3269 <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3270 <ds:Reference URI="#Header1" >...</ds:Reference>
3271 <ds:Reference URI="#Header2" >...</ds:Reference>
3272 <ds:Reference URI="#Body" >...</ds:Reference>
3273 </ds:References>
3274 </ds:SignedInfo>
3275 <ds:SignatureValue>...</ds:SignatureValue>
3276 <ds:KeyInfo>
3277 <wsse:SecurityTokenReference>
3278 <wsse:Reference URI="#_SharedSecretToken" />
3279 </wsse:SecurityTokenReference>
3280 </ds:KeyInfo>
3281 </ds:Signature>
3282 -->
3283 ...
3284 <ds:KeyInfo>
3285 <wsse:SecurityTokenReference>
3286 <wsse:Reference URI="#_SharedSecretToken" />

```

```

3287         </wsse:SecurityTokenReference>
3288     </ds:KeyInfo>
3289 </xenc:EncryptedData>
3290 <ds:Signature>
3291     <ds:SignedInfo>
3292         <ds:References>
3293             <ds:Reference URI="#Signature" >...</ds:Reference>
3294             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3295         </ds:References>
3296     </ds:SignedInfo>
3297     <ds:SignatureValue>...</ds:SignatureValue>
3298     <ds:KeyInfo>
3299         <wsse:SecurityTokenReference>
3300             <wsse:Reference URI="#SomeSupportingToken" />
3301         </wsse:SecurityTokenReference>
3302     </ds:KeyInfo>
3303 </ds:Signature>
3304 <xenc:ReferenceList>
3305     <xenc:DataReference URI="#enc_Body" />
3306     <xenc:DataReference URI="#enc_Header2" />
3307     ...
3308 </xenc:ReferenceList>
3309 </wsse:Security>
3310 </S:Header>
3311 <S:Body wsu:Id="Body">
3312     <xenc:EncryptedData Id="enc_Body">
3313         ...
3314         <ds:KeyInfo>
3315             <wsse:SecurityTokenReference>
3316                 <wsse:Reference URI="#_SharedSecretToken" />
3317             </wsse:SecurityTokenReference>
3318         </ds:KeyInfo>
3319     </xenc:EncryptedData>
3320 </S:Body>
3321 </S:Envelope>

```

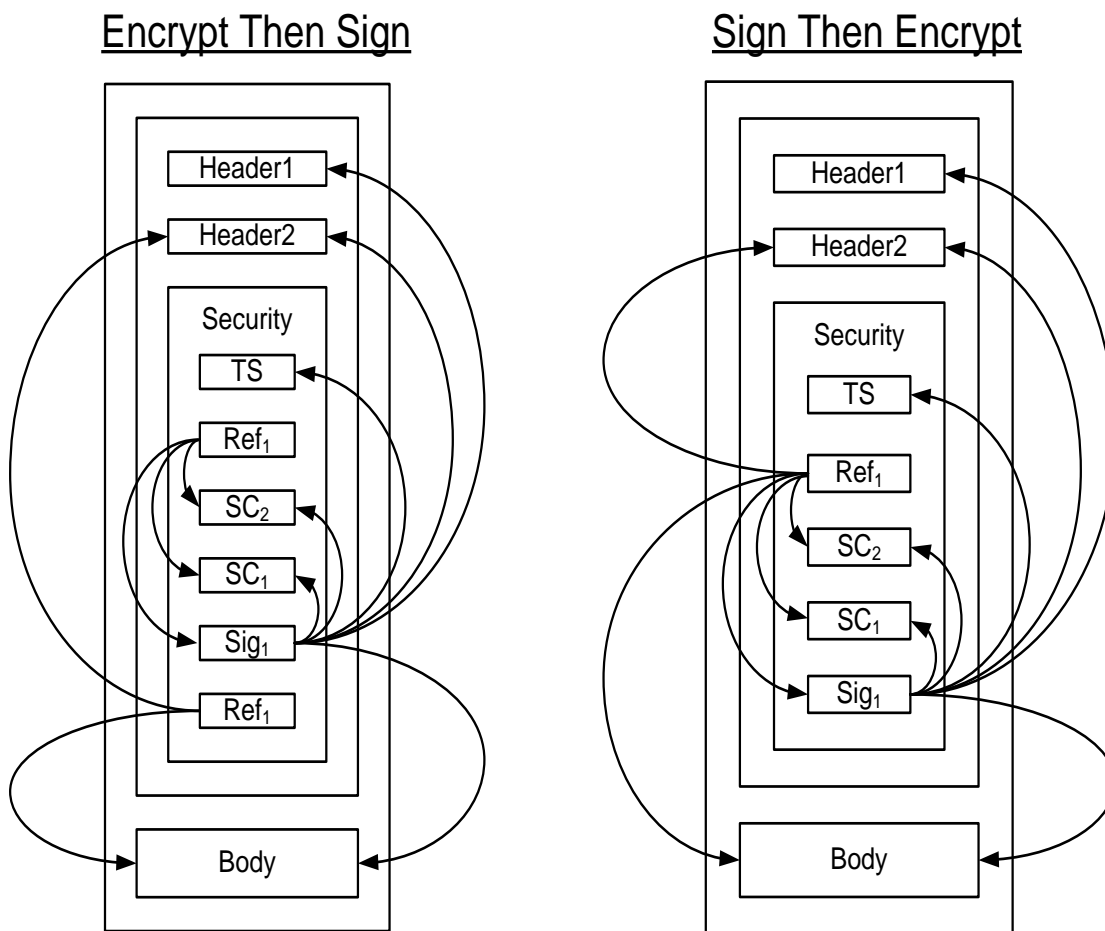
C.2.3 Recipient to Initiator Messages

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:

3363 Recipient to initiator message using EncryptBeforeSigning:

```
3364 <S:Envelope>
3365   <S:Header>
3366     <x:Header1 wsu:Id="Header1" >
3367       ...
3368     </x:Header1>
3369     <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3370       <!-- Plaintext Header2
3371       <x:Header2 wsu:Id="Header2" >
3372         ...
3373       </x:Header2>
3374       -->
3375       ...
3376     </wsse1:EncryptedHeader>
3377     ...
3378   <wsse:Security>
3379     <wsu:Timestamp wsu:Id="Timestamp">
3380       <wsu:Created>...</wsu:Created>
3381       <wsu:Expires>...</wsu:Expires>
3382     </wsu:Timestamp>
3383     <xenc:ReferenceList>
3384       <xenc:DataReference URI="#enc_Signature" />
3385       <xenc:DataReference URI="#enc_SigConf1" />
3386       <xenc:DataReference URI="#enc_SigConf2" />
3387       ...
3388     </xenc:ReferenceList>
3389     <xenc:EncryptedData ID="enc_SigConf1" >
3390       <!-- Plaintext SignatureConfirmation
3391       <wsse1:SignatureConfirmation wsu:Id="SigConf1" >
3392         ...
3393       </wsse1:SignatureConfirmation>
3394       -->
3395       ...
3396     </xenc:EncryptedData>
3397     <xenc:EncryptedData ID="enc_SigConf2" >
3398       <!-- Plaintext SignatureConfirmation
3399       <wsse1:SignatureConfirmation wsu:Id="SigConf2" >
3400         ...
3401       </wsse1:SignatureConfirmation>
3402       -->
3403       ...
3404     </xenc:EncryptedData>
```

```

3405 <xenc:EncryptedData Id="enc_Signature">
3406   <!-- Plaintext Signature
3407   <ds:Signature Id="Signature">
3408     <ds:SignedInfo>
3409       <ds:References>
3410         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3411         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3412         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3413         <ds:Reference URI="#Header1" >...</ds:Reference>
3414         <ds:Reference URI="#Header2" >...</ds:Reference>
3415         <ds:Reference URI="#Body" >...</ds:Reference>
3416       </ds:References>
3417     </ds:SignedInfo>
3418     <ds:SignatureValue>...</ds:SignatureValue>
3419     <ds:KeyInfo>
3420       <wsse:SecurityTokenReference>
3421         <wsse:Reference URI="#_SomeIssuedToken" />
3422       </wsse:SecurityTokenReference>
3423     </ds:KeyInfo>
3424   </ds:Signature>
3425   -->
3426 </xenc:EncryptedData>
3427 ...
3428 <ds:KeyInfo>
3429   <wsse:SecurityTokenReference>
3430     <wsse:Reference URI="#_SomeIssuedToken" />
3431   </wsse:SecurityTokenReference>
3432 </ds:KeyInfo>
3433 <xenc:EncryptedData>
3434 <xenc:ReferenceList>
3435   <xenc:DataReference URI="#enc_Body" />
3436   <xenc:DataReference URI="#enc_Header2" />
3437   ...
3438 </xenc:ReferenceList>
3439 </xenc:EncryptedData>
3440 </wsse:Security>
3441 </S:Header>
3442 <S:Body wsu:Id="Body">
3443   <xenc:EncryptedData Id="enc_Body">
3444     ...
3445     <ds:KeyInfo>
3446       <wsse:SecurityTokenReference>
3447         <wsse:Reference URI="#_SomeIssuedToken" />
3448       </wsse:SecurityTokenReference>
3449     </ds:KeyInfo>
3450   </xenc:EncryptedData>
3451 </S:Body>
3452 </S:Envelope>
3453

```

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>
```

```

3516 <!-- Example Message Policy -->
3517 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3518   <sp:SignedParts>
3519     <sp:Header Name="Header1" Namespace="..." />
3520     <sp:Header Name="Header2" Namespace="..." />
3521     <sp:Body/>
3522   </sp:SignedParts>
3523   <sp:EncryptedParts>
3524     <sp:Header Name="Header2" Namespace="..." />
3525     <sp:Body/>
3526   </sp:EncryptedParts>
3527 </wsp:All>

```

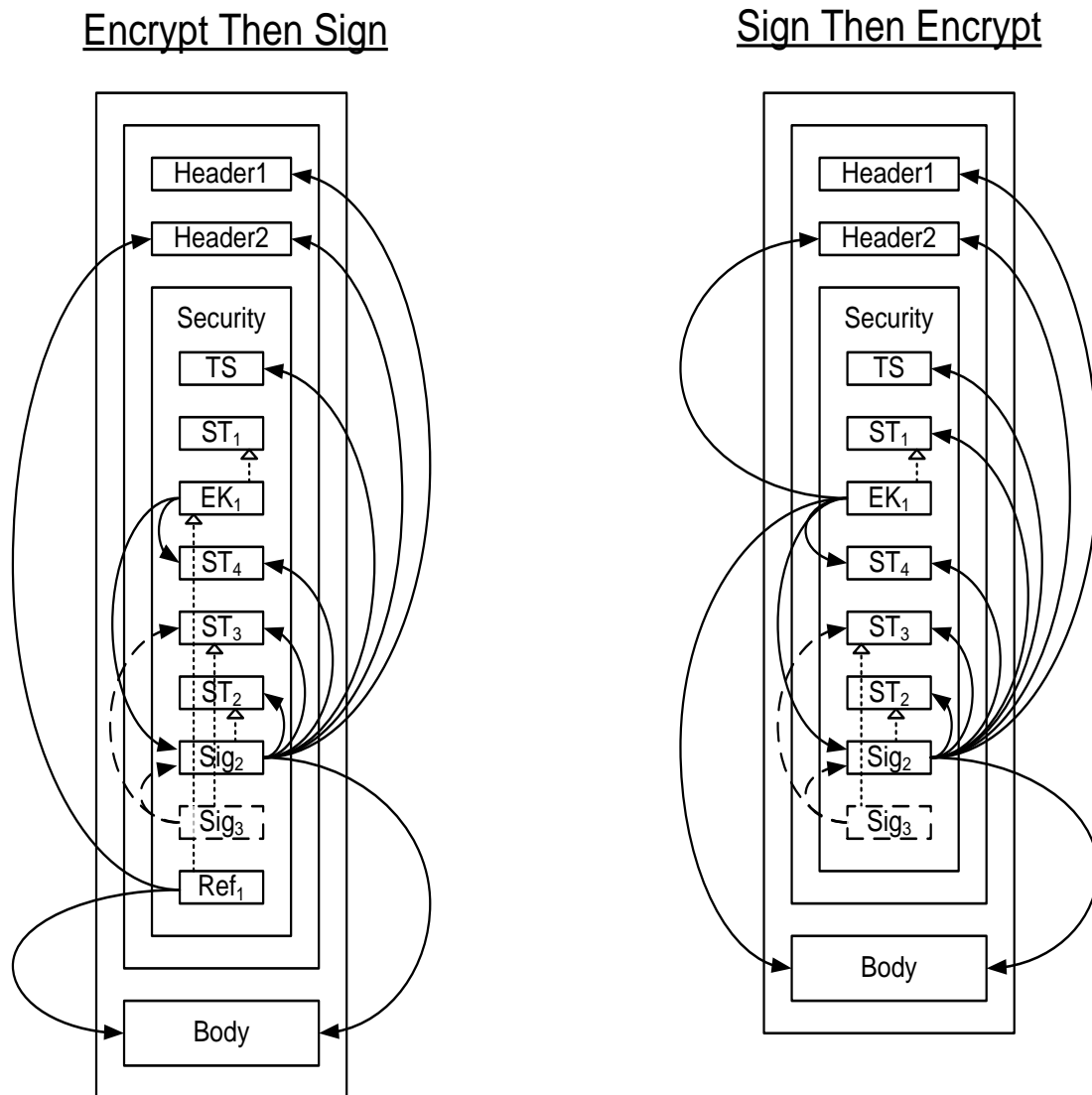
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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.

3564 The following diagram illustrates the security header layout for the initiator to recipient messages:



3565

3566 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂

3567 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate

3568 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as

3569 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references

3570 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left

3571 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the

3572 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as

3573 the basis for each cryptographic operation. In general, the ordering of the items in the security header

3574 follows the most optimal layout for a receiver to process its contents. The rules used to determine this

3575 ordering are described in Appendix C.

3576

3577 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted

3578 key contains an external reference to the token containing the encryption key. The diagram illustrates

3579 how one might attach a security token related to the encrypted key for completeness. One possible use-

3580 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3581 wishes to include the encryption token in the message signature.

3582 Initiator to recipient message *Example*

3583 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3584     xmlns:wsse1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3585 <S:Header>
3586   <x:Header1 wsu:Id="Header1" >
3587     ...
3588   </x:Header1>
3589   <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3590     <!-- Plaintext Header2
3591     <x:Header2 wsu:Id="Header2" >
3592       ...
3593     </x:Header2>
3594     -->
3595     ...
3596   </wsse1:EncryptedHeader>
3597   ...
3598   <wsse:Security>
3599     <wsu:Timestamp wsu:Id="Timestamp">
3600       <wsu:Created>...</wsu:Created>
3601       <wsu:Expires>...</wsu:Expires>
3602     </wsu:Timestamp>
3603     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3604       ...
3605     </wsse:BinarySecurityToken>
3606     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3607       ...
3608       <xenc:ReferenceList>
3609         <xenc:DataReference URI="#enc_Signature" />
3610         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3611         ...
3612       </xenc:ReferenceList>
3613     </xenc:EncryptedKey>
3614     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3615       <!-- Plaintext UsernameToken
3616       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3617         ...
3618       </wsse:UsernameToken>
3619       -->
3620       ...
3621     </xenc:EncryptedData>
3622     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3623       ...
3624     </wsse:BinarySecurityToken>
3625     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3626       ...
3627     </wsse:BinarySecurityToken>
3628     <xenc:EncryptedData ID="enc_Signature">
3629       <!-- Plaintext Signature
3630       <ds:Signature Id="Signature">
3631         <ds:SignedInfo>
3632           <ds:References>
3633             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3634             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3635             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3636             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3637             <ds:Reference URI="#Header1" >...</ds:Reference>
3638             <ds:Reference URI="#Header2" >...</ds:Reference>
3639             <ds:Reference URI="#Body" >...</ds:Reference>
3640           </ds:References>
3641         </ds:SignedInfo>
3642         <ds:SignatureValue>...</ds:SignatureValue>
3643         <ds:KeyInfo>
3644           <wsse:SecurityTokenReference>
3645             <wsse:Reference URI="#InitiatorToken" />
3646           </wsse:SecurityTokenReference>
3647         </ds:KeyInfo>

```

```

3648     </ds:Signature>
3649     -->
3650     ...
3651 </xenc:EncryptedData>
3652 <ds:Signature>
3653   <ds:SignedInfo>
3654     <ds:References>
3655       <ds:Reference URI="#Signature" >...</ds:Reference>
3656       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3657     </ds:References>
3658   </ds:SignedInfo>
3659   <ds:SignatureValue>...</ds:SignatureValue>
3660   <ds:KeyInfo>
3661     <wsse:SecurityTokenReference>
3662       <wsse:Reference URI="#SomeSupportingToken" />
3663     </wsse:SecurityTokenReference>
3664   </ds:KeyInfo>
3665 </ds:Signature>
3666 <xenc:ReferenceList>
3667   <xenc:DataReference URI="#enc_Body" />
3668   <xenc:DataReference URI="#enc_Header2" />
3669   ...
3670 </xenc:ReferenceList>
3671 </wsse:Security>
3672 </S:Header>
3673 <S:Body wsu:Id="Body">
3674   <xenc:EncryptedData Id="enc_Body">
3675     ...
3676     <ds:KeyInfo>
3677       <wsse:SecurityTokenReference>
3678         <wsse:Reference URI="#RecipientEncryptedKey" />
3679       </wsse:SecurityTokenReference>
3680     </ds:KeyInfo>
3681   </xenc:EncryptedData>
3682 </S:Body>
3683 </S:Envelope>

```

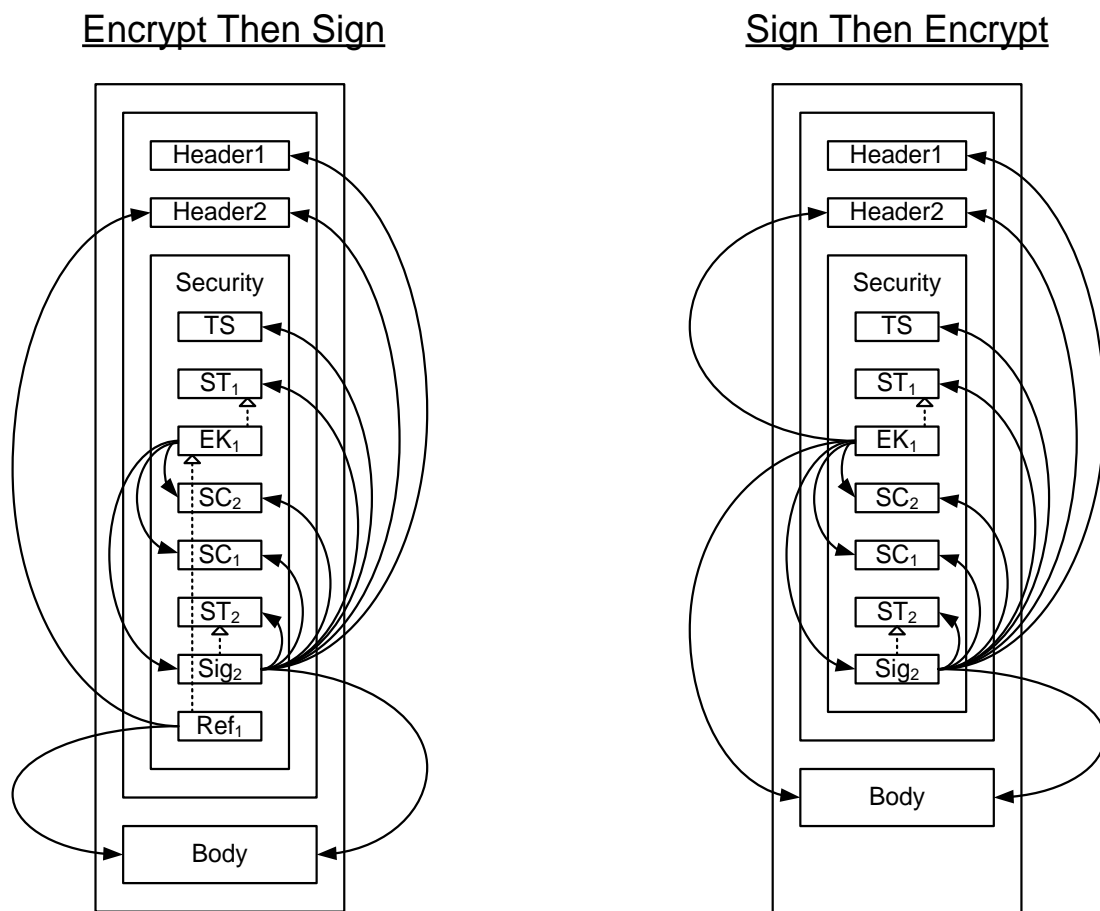
C.3.3 Recipient to Initiator Messages

Messages sent from recipient to initiator have the following layout:

1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is `.../IncludeToken/Always`, then the [Initiator Token].
3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for the initiator. 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 also contain a reference to the message signature from 6 below, if any and references to the `wssell:SignatureConfirmation` elements from 4 below, if any.
4. If [Signature Confirmation] is 'true', then a `wssell: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 `wssell:SignatureConfirmation` element with no Value attribute.
5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is `.../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 E_K₁ 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 E_K₁. 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*:

```

3725 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3726     xmlns:wssell="..." xmlns:wsse="..."
3727     xmlns:xenc="..." xmlns:ds="...">
3728 <S:Header>
3729     <x:Header1 wsu:Id="Header1" >
3730         ...
3731     </x:Header1>
3732     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3733         <!-- Plaintext Header2
3734         <x:Header2 wsu:Id="Header2" >
3735             ...
3736         </x:Header2>
3737         -->
3738         ...
3739     </wssell:EncryptedHeader>
3740     ...
3741 <wsse:Security>
3742     <wsu:Timestamp wsu:Id="Timestamp">
3743         <wsu:Created>...</wsu:Created>
3744         <wsu:Expires>...</wsu:Expires>
3745     </wsu:Timestamp>
3746     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3747         ...
3748     </wsse:BinarySecurityToken>
3749     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3750         ...
3751         <xenc:ReferenceList>
3752             <xenc:DataReference URI="#enc_Signature" />
3753             <xenc:DataReference URI="#enc_SigConf1" />
3754             <xenc:DataReference URI="#enc_SigConf2" />
3755             ...
3756         </xenc:ReferenceList>
3757     </xenc:EncryptedKey>
3758     <xenc:EncryptedData ID="enc_SigConf2" >
3759         <!-- Plaintext SignatureConfirmation
3760         <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3761             ...
3762         </wssell:SignatureConfirmation>
3763         -->
3764         ...
3765     </xenc:EncryptedData>
3766     <xenc:EncryptedData ID="enc_SigConf1" >
3767         <!-- Plaintext SignatureConfirmation
3768         <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3769             ...
3770         </wssell:SignatureConfirmation>
3771         -->
3772         ...
3773     </xenc:EncryptedData>
3774     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3775         ...
3776     </wsse:BinarySecurityToken>
3777

```

```

3778 <xenc:EncryptedData ID="enc_Signature">
3779   <!-- Plaintext Signature
3780   <ds:Signature Id="Signature">
3781     <ds:SignedInfo>
3782       <ds:References>
3783         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3784         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3785         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3786         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3787         <ds:Reference URI="#Header1" >...</ds:Reference>
3788         <ds:Reference URI="#Header2" >...</ds:Reference>
3789         <ds:Reference URI="#Body" >...</ds:Reference>
3790       </ds:References>
3791     </ds:SignedInfo>
3792     <ds:SignatureValue>...</ds:SignatureValue>
3793     <ds:KeyInfo>
3794       <wsse:SecurityTokenReference>
3795         <wsse:Reference URI="#RecipientToken" />
3796       </wsse:SecurityTokenReference>
3797     </ds:KeyInfo>
3798   </ds:Signature>
3799   -->
3800   ...
3801 </xenc:EncryptedData>
3802 <xenc:ReferenceList>
3803   <xenc:DataReference URI="#enc_Body" />
3804   <xenc:DataReference URI="#enc_Header2" />
3805   ...
3806 </xenc:ReferenceList>
3807 </wsse:Security>
3808 </S:Header>
3809 <S:Body wsu:Id="Body">
3810   <xenc:EncryptedData Id="enc_Body">
3811     ...
3812     <ds:KeyInfo>
3813       <wsse:SecurityTokenReference>
3814         <wsse:Reference URI="#InitiatorEncryptedKey" />
3815       </wsse:SecurityTokenReference>
3816     </ds:KeyInfo>
3817   </xenc:EncryptedData>
3818 </S:Body>
3819 </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~~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

3897 Lloyd Burch, Novell
 3898 Scott Cantor, Internet2
 3899 Greg Carpenter, Microsoft Corporation
 3900 Steve Carter, Novell
 3901 Symon Chang, BEA Systems, Inc.
 3902 Ching-Yun (C.Y.) Chao, IBM
 3903 Martin Chapman, Oracle Corporation
 3904 Kate Cherry, Lockheed Martin
 3905 Henry (Hyenvui) Chung, IBM
 3906 Luc Clement, Systinet Corp.
 3907 Paul Cotton, Microsoft Corporation
 3908 Glen Daniels, Sonic Software Corp.
 3909 Peter Davis, Neustar, Inc.
 3910 Martijn de Boer, SAP AG
 3911 Werner Dittmann, Siemens AG
 3912 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
 3913 Fred Dushin, IONA Technologies
 3914 Petr Dvorak, Systinet Corp.
 3915 Colleen Evans, Microsoft Corporation
 3916 Ruchith Fernando, WSO2
 3917 Mark Fussell, Microsoft Corporation
 3918 Vijay Gajjala, Microsoft Corporation
 3919 Marc Goodner, Microsoft Corporation
 3920 Hans Granqvist, VeriSign
 3921 Martin Gudgin, Microsoft Corporation
 3922 Tony Gullotta, SOA Software Inc.
 3923 Jiandong Guo, Sun Microsystems
 3924 Phillip Hallam-Baker, VeriSign
 3925 Patrick Harding, Ping Identity Corporation
 3926 Heather Hinton, IBM
 3927 Frederick Hirsch, Nokia Corporation
 3928 Jeff Hodges, Neustar, Inc.
 3929 Will Hopkins, BEA Systems, Inc.
 3930 Alex Hristov, Otecia Incorporated
 3931 John Hughes, PA Consulting
 3932 Diane Jordan, IBM
 3933 Venugopal K, Sun Microsystems
 3934 Chris Kaler, Microsoft Corporation
 3935 Dana Kaufman, Forum Systems, Inc.
 3936 Paul Knight, Nortel Networks Limited
 3937 Ramanathan Krishnamurthy, IONA Technologies
 3938 Christopher Kurt, Microsoft Corporation
 3939 Kelvin Lawrence, IBM
 3940 Hubert Le Van Gong, Sun Microsystems
 3941 Jong Lee, BEA Systems, Inc.
 3942 Rich Levinson, Oracle Corporation
 3943 Tommy Lindberg, Dajeil Ltd.
 3944 Mark Little, JBoss Inc.
 3945 Hal Lockhart, BEA Systems, Inc.
 3946 Mike Lyons, Layer 7 Technologies Inc.
 3947 Eve Maler, Sun Microsystems
 3948 Ashok Malhotra, Oracle Corporation
 3949 Anand Mani, CrimsonLogic Pte Ltd
 3950 Jonathan Marsh, Microsoft Corporation
 3951 Robin Martherus, Oracle Corporation
 3952 Miko Matsumura, Infravio, Inc.
 3953 Gary McAfee, IBM

3954 Michael McIntosh, IBM
3955 John Merrells, Sxip Networks SRL
3956 Jeff Mischkinsky, Oracle Corporation
3957 Prateek Mishra, Oracle Corporation
3958 Bob Morgan, Internet2
3959 Vamsi Motukuru, Oracle Corporation
3960 Raajmohan Na, EDS
3961 Anthony Nadalin, IBM
3962 Andrew Nash, Reactivity, Inc.
3963 Eric Newcomer, IONA Technologies
3964 Duane Nickull, Adobe Systems
3965 Toshihiro Nishimura, Fujitsu Limited
3966 Rob Philpott, RSA Security
3967 Denis Pilipchuk, BEA Systems, Inc.
3968 Darren Platt, Ping Identity Corporation
3969 Martin Raepple, SAP AG
3970 Nick Ragouzis, Enosis Group LLC
3971 Prakash Reddy, CA
3972 Alain Regnier, Ricoh Company, Ltd.
3973 Irving Reid, Hewlett-Packard
3974 Bruce Rich, IBM
3975 Tom Rutt, Fujitsu Limited
3976 Maneesh Sahu, Actional Corporation
3977 Frank Siebenlist, Argonne National Laboratory
3978 Joe Smith, Apani Networks
3979 Davanum Srinivas, WSO2
3980 Yakov Sverdlov, CA
3981 Gene Thurston, AmberPoint
3982 Victor Valle, IBM
3983 Asir Vedamuthu, Microsoft Corporation
3984 Greg Whitehead, Hewlett-Packard
3985 Ron Williams, IBM
3986 Corinna Witt, BEA Systems, Inc.
3987 Kyle Young, Microsoft Corporation