



WS-SecurityPolicy 1.3

OASIS Committee Specification 01

29 November 2008

Specification URIs:

This Version:

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

Previous Version:

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

Latest Version:

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

Technical Committee:

[OASIS Web Services Secure Exchange TC](#)

Chair(s):

Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):

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

Related work:

N/A

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/ws-sx>.

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

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

Notices

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

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

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

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

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

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

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

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

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

Table of Contents

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

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

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

1 Introduction

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

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

1.1 Example

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

Table 1: Example security policy.

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

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

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

1.2 Namespaces

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

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

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

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

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

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

1.3 Schema Files

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

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

1.4 Terminology

Policy - A collection of policy alternatives.

Policy Alternative - A collection of policy assertions.

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

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

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

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

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

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

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

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

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

Supporting Token - A token used to provide additional claims.

1.4.1 Notational Conventions

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

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

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

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

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

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

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

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

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

1.5 Normative References

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

148	[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
149		http://www.ietf.org/rfc/rfc3986.txt
150		
151		
152		
153	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
154		http://www.ietf.org/rfc/rfc2068.txt
155		
156		
157	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
158		http://www.ietf.org/rfc/rfc2246.txt
159		
160	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
161		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
162		
163	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
164		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
165		
166		
167	[WS-Policy]	W3C Recommendation, "Web Services Policy 1.5 - Framework", 04 September 2007.
168		http://www.w3.org/TR/2007/REC-ws-policy-20070904/
169		
170		W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
171		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
172		
173		
174	[WS-PolicyAttachment]	W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04 September 2007.
175		http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
176		
177		W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
178		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
179		
180		
181		
182	[WS-Trust]	OASIS Committee Specification 01, " WS-Trust 1.4 ", November 2008
183		http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/cd/ws-trust-1.4-spec-cs-01.doc
184		
185		OASIS Standard, "WS-Trust 1.3", March 2007
186		http://docs.oasis-open.org/ws-sx/ws-trust/200512
187		
188	[WS-SecureConversation]	OASIS Committee Specification 01, " WS-SecureConversation 1.4 ", November 2008
189		http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/cd/ws-secureconversation-1.4-spec-cs-01.doc
190		
191		
192		

193	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
194		
195		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
196		
197		
198	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
199		
200		http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
201		
202		
203	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004
204		
205		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
206		
207		
208	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
209		
210		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
211		
212		
213	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
214		
215		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
216		
217		
218	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
219		
220		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
221		
222		
223	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
224		
225		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
226		
227		
228	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
229		
230		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
231		
232	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
233		
234		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
235		
236		
237	[WSS:RELTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
238		
239		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf

240		
241	[WSS:RELTOKENProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language
242		(REL) Token Profile 1.1", February 2006
243		http://www.oasis-open.org/committees/download.php/16687/oasis-
244		wss-rel-token-profile-1.1.pdf
245		
246	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with
247		Attachments (SwA) Profile 1.1", February 2006
248		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-
249		spec-os-SwAProfile.pdf
250		
251	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10
252		December 2002.
253		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
254		
255	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12
256		February 2002.
257		http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/
258		
259		W3C Recommendation, D. Eastlake et al. XML Signature Syntax and
260		Processing (Second Edition). 10 June 2008.
261		http://www.w3.org/TR/2008/REC-xmlsig-core-20080610/
262		
263		
264	[XPath]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16
265		November 1999.
266		http://www.w3.org/TR/1999/REC-xpath-19991116
267		
268	[XPath 2.0 Filter]	W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November
269		2002.
270		http://www.w3.org/TR/2002/REC-xmlsig-filter2-20021108/
271		
272	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second
273		Edition", 28 October 2004.
274		http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
275		
276	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second
277		Edition", 28 October 2004.
278		http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
279		

280 1.6 Non-Normative References

281 None.

282

2 Security Policy Model

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

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

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

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

2.1 Security Assertion Model

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

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

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

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

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

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

2.2 Nested Policy Assertions

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

2.3 Security Binding Abstraction

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

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

A binding defines the following security characteristics:

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

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

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

- 372 • TransportBinding (Section 7.3)
- 373 • SymmetricBinding (Section 7.4)
- 374 • AsymmetricBinding (Section 7.5)

3 Policy Considerations

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

3.1 Nested Policy

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

3.2 Policy Subjects

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

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

[Message Policy Subject]

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

wsdl:message

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

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

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

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

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

[Operation Policy Subject]

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

wsdl:portType/wsdl:operation

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

wsdl:binding/wsdl:operation

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

[Endpoint Policy Subject]

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

wsdl:portType

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

416 wsdl:binding

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

419 wsdl:port

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

4 Protection Assertions

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

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

4.1 Integrity Assertions

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

4.1.1 SignedParts Assertion

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

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

Syntax

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

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

/sp:SignedParts

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

/sp:SignedParts/sp:Body

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

/sp:SignedParts/sp:Header

465 Presence of this OPTIONAL element indicates a specific SOAP header, its attributes and content
 466 (or set of such headers) needs to be protected. There may be multiple sp:Header elements within
 467 a single sp:SignedParts element. If multiple SOAP headers with the same local name but
 468 different namespace names are to be integrity protected multiple sp:Header elements are
 469 needed, either as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts
 470 assertions.
 471 This element only applies to SOAP header elements targeted to the same actor/role as the
 472 Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
 473 SOAP Header elements targeted to a different actor/role, that may be accomplished using the
 474 sp:SignedElements assertion.

475 /sp:SignedParts/sp:Header/@Name
 476 This OPTIONAL attribute indicates the local name of the SOAP header to be integrity protected. If
 477 this attribute is not specified, all SOAP headers whose namespace matches the Namespace
 478 attribute are to be protected.

479 /sp:SignedParts/sp:Header/@Namespace
 480 This REQUIRED attribute indicates the namespace of the SOAP header(s) to be integrity
 481 protected.

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

487 /sp:SignedParts/sp:Attachments/sp13:ContentSignatureTransform
 488 Presence of this OPTIONAL empty element indicates that the
 489 AttachmentContentSignatureTransform must be used as part of attachment protection.

490 /sp:SignedParts/sp:Attachments/sp13:AttachmentCompleteSignatureTransform
 491 Presence of this OPTIONAL empty element indicates that the
 492 AttachmentCompleteSignatureTransform must be used as part of attachment protection.
 493 This is the default if neither sp13:ContentSignatureTransform or
 494 sp13:AttachmentCompleteSignatureTransform are specified.

495 4.1.2 SignedElements Assertion

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

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

505 Syntax

```
506 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
507   <sp:XPath>xs:string</sp:XPath>+
508   <sp13:Xpath2 Filter="xs:string">xs:string</sp13:Xpath2>+
509   ...
510 </sp:SignedElements>
```

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

529 4.2 Confidentiality Assertions

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

533 4.2.1 EncryptedParts Assertion

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

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

544 Syntax

```
545 <sp:EncryptedParts xmlns:sp="..." ... >
546   <sp:Body/>?
547   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
548   <sp:Attachments />?
549   ...
550 </sp:EncryptedParts>
```

551
 552 The following describes the attributes and elements listed in the schema outlined above:
 553 /sp:EncryptedParts

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

585 4.2.2 EncryptedElements Assertion

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

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

595 Syntax

```
596 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
597   <sp:XPath>xs:string</sp:XPath>+
598   ...
599 </sp:EncryptedElements>
```

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

612 4.2.3 ContentEncryptedElements Assertion

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

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

622 Syntax

```
623 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
624   <sp:XPath>xs:string</sp:XPath>+  
625   ...  
626 </sp:ContentEncryptedElements>
```

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

639 4.3 Required Elements Assertion

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

642

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

4.3.1 RequiredElements Assertion

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

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

Syntax

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

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

/sp:RequiredElements

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

/sp:RequiredElements/@XPathVersion

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

/sp:RequiredElements/sp:XPath

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

4.3.2 RequiredParts Assertion

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

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

Syntax

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

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

/sp:RequiredParts/sp:Header

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

/sp:RequiredParts/sp:Header/@Name

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

5 Token Assertions

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

5.1 Token Inclusion

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

5.1.1 Token Inclusion Values

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

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

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

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

5.1.2 Token Inclusion and Token References

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

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

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

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

5.2 Token Issuer and Required Claims

5.2.1 Token Issuer

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

5.2.2 Token Issuer Name

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

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

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

5.2.3 Required Claims

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

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

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

5.2.4 Processing Rules and Token Matching

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

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

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

5.3 Token Properties

5.3.1 [Derived Keys] Property

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

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

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

5.3.2 [Explicit Derived Keys] Property

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

5.3.3 [Implied Derived Keys] Property

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

5.4 Token Assertion Types

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

5.4.1 UsernameToken Assertion

This element represents a requirement to include a username token.

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

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

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

Syntax

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

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

/sp:UsernameToken

This identifies a UsernameToken assertion.

/sp:UsernameToken/@sp:IncludeToken

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

/sp:UsernameToken/sp:Issuer

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

/sp:UsernameToken/sp:IssuerName

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

/sp:UsernameToken/wst:Claims

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

850 /sp:UsernameToken/wsp:Policy

851 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken
 852 assertion.

853 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

856 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

861 /sp13:UsernameToken/wsp:Policy/sp13:Created

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

865 /sp13:UsernameToken/wsp:Policy/sp13:Nonce

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

869 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

872 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

876 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

880 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

881 This OPTIONAL element is a policy assertion that indicates that a Username token should be
 882 used as defined in [\[WSS:UsernameTokenProfile1.0\]](#).

883 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

884 This OPTIONAL element is a policy assertion that indicates that a Username token should be
 885 used as defined in [\[WSS:UsernameTokenProfile1.1\]](#).

886 5.4.2 ICreatessuedToken Assertion

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

891 Syntax

```

892 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
893 (
894   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
895   <sp:IssuerName>xs:anyURI</sp:IssuerName>
896 ) ?
897 <wst:Claims Dialect="..."> ... </wst:Claims> ?
898 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
899   ...
900 </sp:RequestSecurityTokenTemplate>
901 <wsp:Policy xmlns:wsp="...">
902   (
903     <sp:RequireDerivedKeys ... /> |
904     <sp:RequireImpliedDerivedKeys ... /> |
905     <sp:RequireExplicitDerivedKeys ... />
906   ) ?
907   <sp:RequireExternalReference ... /> ?
908   <sp:RequireInternalReference ... /> ?
909   ...
910 </wsp:Policy>
911 ...
912 </sp:IssuedToken>

```

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

914 /sp:IssuedToken

915 This identifies an IssuedToken assertion.

916 /sp:IssuedToken/@sp:IncludeToken

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

918 /sp:IssuedToken/sp:Issuer

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

921 /sp:IssuedToken/sp:IssuerName

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

924 /sp:IssuedToken/wst:Claims

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

927 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

932 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

933 This OPTIONAL attribute contains a WS-Trust specification namespace URI identifying the
934 version of WS-Trust referenced by the contents of this element. For example, when using Trust
935 1.3 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200512> should be used and when using
936 Trust 1.4 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200802> should be used.

937 /sp:IssuedToken/wsp:Policy

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

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

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

943 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

947 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

951 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

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

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

955 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

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

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

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

964 5.4.3 X509Token Assertion

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

966 Syntax

```
967 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
968   (  
969     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
970     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
971   ) ?  
972   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```



```

973 <wsp:Policy xmlns:wsp="...">
974   (
975     <sp:RequireDerivedKeys ... /> |
976     <sp:RequireExplicitDerivedKeys ... /> |
977     <sp:RequireImpliedDerivedKeys ... />
978   ) ?
979   <sp:RequireKeyIdentifierReference ... /> ?
980   <sp:RequireIssuerSerialReference ... /> ?
981   <sp:RequireEmbeddedTokenReference ... /> ?
982   <sp:RequireThumbprintReference ... /> ?
983   (
984     <sp:WssX509V3Token10 ... /> |
985     <sp:WssX509Pkcs7Token10 ... /> |
986     <sp:WssX509PkiPathV1Token10 ... /> |
987     <sp:WssX509V1Token11 ... /> |
988     <sp:WssX509V3Token11 ... /> |
989     <sp:WssX509Pkcs7Token11 ... /> |
990     <sp:WssX509PkiPathV1Token11 ... />
991   ) ?
992   ...
993 </wsp:Policy>
994 ...
995 </sp:X509Token>

```

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

/sp:X509Token

This identifies an X509Token assertion.

/sp:X509Token/@sp:IncludeToken

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

/sp:X509Token/sp:Issuer

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

/sp:X509Token/sp:IssuerName

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

/sp:X509Token/wst:Claims

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

/sp:X509Token/wsp:Policy

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

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

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

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

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

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

1022 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1023 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1024 'false'.

1025 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

1026 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1027 REQUIRED when referencing this token.

1028 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

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

1031 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

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

1034 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1035 This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is
1036 REQUIRED when referencing this token.

1037 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1038 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
1039 be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1040 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1041 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
1042 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1043 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1044 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
1045 token should be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1046 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

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

1049 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1050 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should
1051 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1052 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1053 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be
1054 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1055 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1056 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1
1057 token should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1058 5.4.4 KerberosToken Assertion

1059 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

1060 Syntax

```
1061 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1062 (
1063   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1064   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1065 ) ?
```

```

1066 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1067 <wsp:Policy xmlns:wsp="...">
1068   (
1069     <sp:RequireDerivedKeys ... /> |
1070     <sp:RequireImpliedDerivedKeys ... /> |
1071     <sp:RequireExplicitDerivedKeys ... />
1072   ) ?
1073   <sp:RequireKeyIdentifierReference ... /> ?
1074   (
1075     <sp:WssKerberosV5ApReqToken11 ... /> |
1076     <sp:WssGssKerberosV5ApReqToken11 ... />
1077   ) ?
1078
1079   ...
1080 </wsp:Policy>
1081 ...
1082 </sp:KerberosToken>

```

1083

1084 The following describes the attributes and elements listed in the schema outlined above:

1085 /sp:KerberosToken

1086 This identifies a KerberosV5ApReqToken assertion.

1087 /sp:KerberosToken/@sp:IncludeToken

1088 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1089 /sp:KerberosToken/sp:Issuer

1090 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1091 of the sp:KerberosToken.

1092 /sp:KerberosToken/sp:IssuerName

1093 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken
1094 issuer.

1095 /sp:KerberosToken/wst:Claims

1096 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1097 order to satisfy the token assertion requirements.

1098 /sp:KerberosToken/wsp:Policy

1099 This REQUIRED element identifies additional requirements for use of the sp:KerberosToken
1100 assertion.

1101 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

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

1104 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1105 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1106 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1107 'false'.

1108 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1109 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1110 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1111 'false'.

1112 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1113 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1114 REQUIRED when referencing this token.

1115 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

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

1118 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1119 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-
 1120 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

1121 5.4.5 SpnegoContextToken Assertion

1122 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
 1123 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1124 Syntax

```

1125 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1126 (
1127   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1128   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1129 ) ?
1130 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1131 <wsp:Policy xmlns:wsp="...">
1132   (
1133     <sp:RequireDerivedKeys ... /> |
1134     <sp:RequireImpliedDerivedKeys ... /> |
1135     <sp:RequireExplicitDerivedKeys ... />
1136   ) ?
1137   <sp:MustNotSendCancel ... /> ?
1138   <sp:MustNotSendAmend ... /> ?
1139   <sp:MustNotSendRenew ... /> ?
1140   ...
1141 </wsp:Policy>
1142 ...
1143 </sp:SpnegoContextToken>

```

1144

1145 The following describes the attributes and elements listed in the schema outlined above:

1146 /sp:SpnegoContextToken

1147 This identifies a SpnegoContextToken assertion.

1148 /sp:SpnegoContextToken/@sp:IncludeToken

1149 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1150 /sp:SpnegoContextToken/sp:Issuer

1151 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
 1152 for the Spnego Context Token.

1153 /sp:SpnegoContextToken/sp:IssuerName

1154 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
 1155 sp:SpnegoContextToken issuer.

1156 /sp:SpnegoContextToken/wst:Claims

1157 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1158 order to satisfy the token assertion requirements.

1159 /sp:SpnegoContextToken/wsp:Policy

1160 This REQUIRED element identifies additional requirements for use of the
 1161 sp:SpnegoContextToken assertion.

1162 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

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

1165 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1166 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1167 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1168 'false'.

1169 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1170 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1171 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1172 'false'.

1173 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

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

1177 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1178 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1179 token does not support SCT/Amend RST messages. If this assertion is missing it means that
 1180 SCT/Amend RST messages are supported by the STS.

1181 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1182 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego
 1183 token does not support SCT/Renew RST messages. If this assertion is missing it means that
 1184 SCT/Renew RST messages are supported by the STS.

1185 5.4.6 SecurityContextToken Assertion

1186 This element represents a requirement for a SecurityContextToken token.

1187 Syntax

```

1188 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1189 (
1190   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1191   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1192 ) ?
1193 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1194 <wsp:Policy xmlns:wsp="...">
1195   (
1196     <sp:RequireDerivedKeys ... /> |
1197     <sp:RequireImpliedDerivedKeys ... /> |
1198     <sp:RequireExplicitDerivedKeys ... />
1199   ) ?
1200   <sp:RequireExternalUriReference ... /> ?
1201   <sp:SC13SecurityContextToken... /> ?
1202   ...
1203 </wsp:Policy>
1204 ...
1205 </sp:SecurityContextToken>

```

1206

1207 The following describes the attributes and elements listed in the schema outlined above:

1208 /sp:SecurityContextToken

1209 This identifies a SecurityContextToken assertion.

1210 /sp:SecurityContextToken/@sp:IncludeToken

1211 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1212 /sp:SecurityContextToken/sp:Issuer

1213 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer

1214 of the sp:SecurityContextToken.

1215 /sp:SecurityContextToken/sp:IssuerName

1216 This OPTIONAL element, of type xs:anyURI, contains the logical name of the

1217 sp:SecurityContextToken issuer.

1218 /sp:SecurityContextToken/wst:Claims

1219 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in

1220 order to satisfy the token assertion requirements.

1221 /sp:SecurityContextToken/wsp:Policy

1222 This REQUIRED element identifies additional requirements for use of the

1223 sp:SecurityContextToken assertion.

1224 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1225 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]

1226 and [Implied Derived Keys] properties for this token to 'true'.

1227 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1228 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived

1229 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to

1230 'false'.

1231 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1232 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived

1233 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to

1234 'false'.

1235 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1236 This OPTIONAL element is a policy assertion that indicates that an external URI reference is

1237 REQUIRED when referencing this token.

1238 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1239 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should

1240 be used as defined in [\[WS-SecureConversation\]](#).

1241

1242 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that

1243 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If

1244 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the

1245 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

1246 5.4.7 SecureConversationToken Assertion

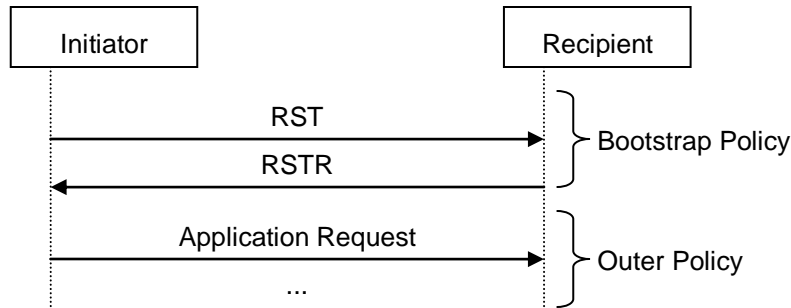
1247 This element represents a requirement for a Security Context Token retrieved from the indicated issuer

1248 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the

1249 service endpoint address.

1250

1251 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1252 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD
1253 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1254 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1255 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1256 below.



1257

1258

1259 If the bootstrap policy assertion is used to indicate the security binding and policy in effect when
1260 requesting a secure conversation token from the target service, then subsequent Amend, Renew and
1261 Cancel messages MUST comply with the following rules.

1262 **Amending Context**

1263 To amend an existing secure conversation token, a requestor uses the context amending mechanism as
1264 described by the WS-SecureConversation specification. The message exchange MUST be secured
1265 using the existing (to be amended) SCT in accordance with the target service (outer) policy, combined
1266 with endorsing supporting tokens carrying the new claims to be associated with the amended context with
1267 the inclusion mode set to:

1268 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1269 See the EndorsingSupportingTokens Assertion section for more details on the usage of the endorsing
1270 supporting tokens.

1271 **Renewing Context**

1272 To renew an existing secure conversation token, a requestor uses the context renewal mechanism as
1273 described by the WS-SecureConversation specification. The message exchange MUST be secured
1274 according to the requirements of the bootstrap policy assertion, combined with the existing (to be
1275 renewed) SCT used as an endorsing supporting token with the inclusion mode set to:

1276 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1277 See the EndorsingSupportingTokens Assertion section for more details on the usage of endorsing
1278 support tokens.

1279 **Canceling Context**

1280 To cancel an existing secure conversation token, a requestor uses the context cancelling mechanism as
1281 described by the WS-SecureConversation specification. The message exchange MUST be secured
1282 using the existing (to be cancelled) SCT in accordance with the target service (outer) policy.

1283 **Handling Policy Alternatives**

1284 If there are policy alternatives present in either the bootstrap policy assertion or the target service (outer)
1285 policy assertion, the following rules MUST be followed.

- 1286 • The policy alternative used as a basis for the context renewal MUST be the same as the policy
1287 alternative which was previously used for the context issuance.

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

Policy

Policy-alternative-1

SecureConversationToken-assertion-1

Policy-alternative-2

SecureConversationToken-assertion-2

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

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

Policy

SecureConversationToken-assertion-1

Policy-alternative-1

Policy-alternative-2

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

Syntax

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

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

/sp:SecureConversationToken

This identifies a SecureConversationToken assertion.

/sp:SecureConversationToken/@sp:IncludeToken

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

1338 /sp:SecureConversationToken/sp:Issuer
 1339 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer
 1340 for the Security Context Token.

1341 /sp:SecureConversationToken/sp:IssuerName
 1342 This OPTIONAL element, of type xs:anyURI, contains the logical name of the
 1343 sp:SecureConversationToken issuer.

1344 /sp:SpnegoContextToken/wst:Claims
 1345 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1346 order to satisfy the token assertion requirements.

1347 /sp:SecureConversationToken/wsp:Policy
 1348 This REQUIRED element identifies additional requirements for use of the
 1349 sp:SecureConversationToken assertion.

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

1353 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1354 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1355 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1356 'false'.

1357 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1358 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1359 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1360 'false'.

1361 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference
 1362 This OPTIONAL element is a policy assertion that indicates that an external URI reference is
 1363 REQUIRED when referencing this token.

1364 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken
 1365 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should
 1366 be used as obtained using the protocol defined in [\[WS-SecureConversation\]](#).

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

1371 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend
 1372 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
 1373 conversation token does not support SCT/Amend RST messages. If this assertion is missing it
 1374 means that SCT/Amend RST messages are supported by the STS.

1375 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew
 1376 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure
 1377 conversation token does not support SCT/Renew RST messages. If this assertion is missing it
 1378 means that SCT/Renew RST messages are supported by the STS.

1379 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy
 1380 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements
 1381 for obtaining the Security Context Token.

1382 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy
1383 This element contains the security binding requirements for obtaining the Security Context Token.
1384 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
1385 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
1386 are to be protected.

1387 Example

```
1388 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1389   <sp:SymmetricBinding>
1390     <wsp:Policy>
1391       <sp:ProtectionToken>
1392         <wsp:Policy>
1393           <sp:SecureConversationToken>
1394             <sp:Issuer>
1395               <wsa:Address>http://example.org/sts</wsa:Address>
1396             </sp:Issuer>
1397             <wsp:Policy>
1398               <sp:SC13SecurityContextToken />
1399               <sp:BootstrapPolicy>
1400                 <wsp:Policy>
1401                   <sp:AsymmetricBinding>
1402                     <wsp:Policy>
1403                       <sp:InitiatorToken>
1404                         ...
1405                       </sp:InitiatorToken>
1406                       <sp:RecipientToken>
1407                         ...
1408                       </sp:RecipientToken>
1409                     </wsp:Policy>
1410                   </sp:AsymmetricBinding>
1411                   <sp:SignedParts>
1412                     ...
1413                   </sp:SignedParts>
1414                   ...
1415                 </wsp:Policy>
1416               </sp:BootstrapPolicy>
1417             </wsp:Policy>
1418           </sp:SecureConversationToken>
1419         </wsp:Policy>
1420       </sp:ProtectionToken>
1421       ...
1422     </wsp:Policy>
1423   </sp:SymmetricBinding>
1424   <sp:SignedParts>
1425     ...
1426   </sp:SignedParts>
1427   ...
1428 </wsp:Policy>
```

1429 5.4.8 SamlToken Assertion

1430 This element represents a requirement for a SAML token.

1431 Syntax

```
1432 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1433   (
1434     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1435     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1436   ) ?
1437   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```

1438 <wsp:Policy xmlns:wsp="...">
1439   (
1440     <sp:RequireDerivedKeys ... /> |
1441     <sp:RequireImpliedDerivedKeys ... /> |
1442     <sp:RequireExplicitDerivedKeys ... />
1443   ) ?
1444   <sp:RequireKeyIdentifierReference ... /> ?
1445   (
1446     <sp:WssSamlV11Token10 ... /> |
1447     <sp:WssSamlV11Token11 ... /> |
1448     <sp:WssSamlV20Token11 ... />
1449   ) ?
1450   ...
1451 </wsp:Policy>
1452 ...
1453 </sp:SamlToken>

```

1454

1455 The following describes the attributes and elements listed in the schema outlined above:

1456 /sp:SamlToken

1457 This identifies a SamlToken assertion.

1458 /sp:SamlToken/@sp:IncludeToken

1459 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1460 /sp:SamlToken/sp:Issuer

1461 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1462 of the sp:SamlToken.

1463 /sp:SamlToken/sp:IssuerName

1464 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken
1465 issuer.

1466 /sp:SamlToken/wst:Claims

1467 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1468 order to satisfy the token assertion requirements.

1469 /sp:SamlToken/wsp:Policy

1470 This REQUIRED element identifies additional requirements for use of the sp:SamlToken
1471 assertion.

1472 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

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

1475 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1476 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
1477 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
1478 'false'.

1479 /sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1480 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1481 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1482 'false'.

1483 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1484 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
1485 REQUIRED when referencing this token.

1486 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10
1487 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1488 should be used as defined in [[WSS:SAMLTokenProfile1.0](#)].

1489 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11
1490 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token
1491 should be used as defined in [[WSS:SAMLTokenProfile1.1](#)].

1492 /sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11
1493 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token
1494 should be used as defined in [[WSS:SAMLTokenProfile1.1](#)].

1495
1496 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1497 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1498 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion
1499 SHOULD be used instead.

1500 5.4.9 RelToken Assertion

1501 This element represents a requirement for a REL token.

1502 Syntax

```
1503 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1504   (  
1505     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1506     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1507   ) ?  
1508   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1509   <wsp:Policy xmlns:wsp="...">  
1510     (  
1511       <sp:RequireDerivedKeys ... /> |  
1512       <sp:RequireImpliedDerivedKeys ... /> |  
1513       <sp:RequireExplicitDerivedKeys ... />  
1514     ) ?  
1515     <sp:RequireKeyIdentifierReference ... /> ?  
1516     (  
1517       <sp:WssRelV10Token10 ... /> |  
1518       <sp:WssRelV20Token10 ... /> |  
1519       <sp:WssRelV10Token11 ... /> |  
1520       <sp:WssRelV20Token11 ... />  
1521     ) ?  
1522     ...  
1523   </wsp:Policy>  
1524   ...  
1525 </sp:RelToken>
```

1526
1527 The following describes the attributes and elements listed in the schema outlined above:

1528 /sp:RelToken

1529 This identifies a RelToken assertion.

1530 /sp:RelToken/@sp:IncludeToken

1531 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1532 /sp:RelToken/sp:Issuer

1533 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1534 of the sp:RelToken.

1535 /sp:RelToken/sp:IssuerName
 1536 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken
 1537 issuer.

1538 /sp:RelToken/wst:Claims
 1539 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
 1540 order to satisfy the token assertion requirements.

1541 /sp:RelToken/wsp:Policy
 1542 This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

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

1546 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys
 1547 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived
 1548 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to
 1549 'false'.

1550 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1551 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1552 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1553 'false'.

1554 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1555 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is
 1556 REQUIRED when referencing this token.

1557 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10
 1558 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1559 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1560 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10
 1561 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1562 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1563 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11
 1564 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should
 1565 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1566 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11
 1567 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should
 1568 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1569

1570 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
 1571 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
 1572 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion
 1573 SHOULD be used instead.

1574 **5.4.10 HttpsToken Assertion**

1575 This element represents a requirement for a transport binding to support the use of HTTPS.

1576 **Syntax**

```

1577 <sp:HttpsToken xmlns:sp="..." ... >
1578 (
1579   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1580   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1581 ) ?
1582 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1583 <wsp:Policy xmlns:wsp="...">
1584 (
1585   <sp:HttpBasicAuthentication /> |
1586   <sp:HttpDigestAuthentication /> |
1587   <sp:RequireClientCertificate /> |
1588   ...
1589 ) ?
1590 ...
1591 </wsp:Policy>
1592 ...
1593 </sp:HttpsToken>

```

1594 The following describes the attributes and elements listed in the schema outlined above:

1595 /sp:HttpsToken

1596 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1597 supported.

1598 /sp:HttpsToken/sp:Issuer

1599 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer
1600 of the sp:HttpsToken.

1601 /sp:HttpsToken/sp:IssuerName

1602 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken
1603 issuer.

1604 /sp:HttpsToken/wst:Claims

1605 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in
1606 order to satisfy the token assertion requirements.

1607 /sp:HttpsToken/wsp:Policy

1608 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken
1609 assertion.

1610 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

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

1613 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1614 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP
1615 Digest Authentication [[RFC2068](#)] to authenticate to the service.

1616 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1617 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a
1618 certificate when negotiating the HTTPS session.

1619 **5.4.11 KeyValueToken Assertion**

1620 This element represents a requirement for a KeyValue token. The next section defines the KeyValue
1621 security token abstraction for purposes of this token assertion.
1622

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

Syntax

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

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

/sp:KeyValueToken

This identifies a RsaToken assertion.

/sp:KeyValueToken/@sp:IncludeToken

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

/sp:KeyValueToken/wsp:Policy

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

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

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

5.4.11.1 Key Value Token

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

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

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

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

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

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

```

1675     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1676   </Transforms>
1677   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1678   <DigestValue>...</DigestValue>
1679 </Reference>
1680 </SignedInfo>
1681 <SignatureValue>...</SignatureValue>
1682 <KeyInfo>
1683   <KeyValue>
1684     <RSAKeyValue>
1685       <Modulus>...</Modulus>
1686       <Exponent>...</Exponent>
1687     </RSAKeyValue>
1688   </KeyValue>
1689 </KeyInfo>
1690 </Signature>

```

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

```

1696 <t:RequestSecurityToken xmlns:t="...">
1697   <t:RequestType>...</t:RequestType>
1698   ...
1699   <t:UseKey>
1700     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1701       <KeyValue>
1702         <RSAKeyValue>
1703           <Modulus>...</Modulus>
1704           <Exponent>...</Exponent>
1705         </RSAKeyValue>
1706       </KeyValue>
1707     </KeyInfo>
1708   </t:UseKey>
1709 </t:RequestSecurityToken>

```

6 Security Binding Properties

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

6.1 [Algorithm Suite] Property

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

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

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

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

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

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

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

1746

1747 The tables below show all the base algorithm suites defined by this specification. This table defines
 1748 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

1749 This table defines additional properties whose values can be specified along with the default value for that
 1750 property.

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

1751 This table defines values for the remaining components for each algorithm suite.

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

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

6.2 [Timestamp] Property

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

6.3 [Protection Order] Property

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

EncryptBeforeSigning	Signature MUST be computed over ciphertext. Encryption key and signing key MUST be derived from the same source key unless distinct keys are provided, see Section 7.5 on the AsymmetricBinding.
SignBeforeEncrypting	Signature MUST be computed over plaintext. The resulting signature SHOULD be encrypted. Supporting signatures MUST be over the plaintext signature.

The default value for this property is 'SignBeforeEncrypting'.

6.4 [Signature Protection] Property

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

6.5 [Token Protection] Property

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

6.6 [Entire Header and Body Signatures] Property

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

6.7 [Security Header Layout] Property

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

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

6.7.1 Strict Layout Rules for WSS 1.0

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

- 1803 b. A Username token (usually in encrypted form) MUST occur before the signature that
1804 signs it.
- 1805 c. A primary signature MUST occur before the supporting token signature that signs the
1806 primary signature's signature value element.
- 1807 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1808 has the same order requirements as the source plain text element, unless requirement 4
1809 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1810 supporting token signature per 2.c above and an encrypted token has the same ordering
1811 requirements as the unencrypted token.
- 1812 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1813 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1814 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1815 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1816 Layout Rules for WSS 1.1
- 1817 1. Tokens that are included in the message MUST be declared before use. For example:
- 1818 a. A local signing token MUST occur before the signature that uses it.
- 1819 b. A local token serving as the source token for a derived key token MUST occur before that
1820 derived key token.
- 1821 c. A local encryption token MUST occur before the reference list that points to
1822 xenc:EncryptedData elements that use it.
- 1823 d. If the same token is used for both signing and encryption, then it SHOULD appear before
1824 the ds:Signature and xenc:ReferenceList elements in the security header that are
1825 generated using the token.
- 1826 2. Signed elements inside the security header MUST occur before the signature that signs them.
1827 For example:
- 1828 a. A timestamp MUST occur before the signature that signs it.
- 1829 b. A Username token (usually in encrypted form) MUST occur before the signature that
1830 signs it.
- 1831 c. A primary signature MUST occur before the supporting token signature that signs the
1832 primary signature's signature value element.
- 1833 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1834 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1835 has the same order requirements as the source plain text element, unless requirement 4
1836 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1837 supporting token signature per 2.c above and an encrypted token has the same ordering
1838 requirements as the unencrypted token.
- 1839 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1840 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1841 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1842 However, the xenc:ReferenceList is NOT REQUIRED to appear before independently encrypted
1843 tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1844 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1845 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>InclusiveC14N11 ... /> ?
    <sp>SOAPNormalization10 ... /> ?
    <sp>STRTransform10 ... /> ?
    (<sp>XPath10 ... /> |
    <sp>XPathFilter20 ... /> |
    <sp>AbsXPath ... /> |
    ...) ?
    ...
  </wsp:Policy>
  ...
</sp:AlgorithmSuite>
```

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

/sp:AlgorithmSuite

This identifies an AlgorithmSuite assertion.

/sp:AlgorithmSuite/wsp:Policy

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

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

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

1895 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192
1896 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1897 set to 'Basic192'.

1898 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128
1899 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1900 set to 'Basic128'.

1901 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes
1902 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1903 set to 'TripleDes'.

1904 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15
1905 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1906 set to 'Basic256Rsa15'.

1907 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15
1908 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1909 set to 'Basic192Rsa15'.

1910 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15
1911 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1912 set to 'Basic128Rsa15'.

1913 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15
1914 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1915 set to 'TripleDesRsa15'.

1916 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256
1917 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1918 set to 'Basic256Sha256'.

1919 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256
1920 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1921 set to 'Basic192Sha256'.

1922 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256
1923 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1924 set to 'Basic128Sha256'.

1925 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256
1926 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1927 set to 'TripleDesSha256'.

1928 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15
1929 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1930 set to 'Basic256Sha256Rsa15'.

1931 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15
1932 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1933 set to 'Basic192Sha256Rsa15'.

1934 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15
1935 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1936 set to 'Basic128Sha256Rsa15'.

1937 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1938 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is
1939 set to 'TripleDesSha256Rsa15'.

1940 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1941 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an
1942 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]
1943 property is 'ExC14N'.

1944 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N11

1945
1946 This optional element is a policy assertion that indicates that the
1947 [C14N] property of an algorithm suite is set to 'C14N11'. Note: as
1948 indicated in Section 6.1 the default value of the [C14N] property is
1949 'ExC14N'.

1950 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1951 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set
1952 to 'SNT'.

1953 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1954 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is
1955 set to 'STRT10'.

1956 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

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

1959 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1960 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1961 'XPath20'.

1962 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1963 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to
1964 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1965

1966 7.2 Layout Assertion

1967 This assertion indicates a requirement for a particular security header layout as defined under the
1968 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1969 containing assertion.

1970 Syntax

```
1971 <sp:Layout xmlns:sp="..." ... >  
1972   <wsp:Policy xmlns:wsp="...">  
1973     <sp:Strict ... /> |  
1974     <sp:Lax ... /> |  
1975     <sp:LaxTsFirst ... /> |  
1976     <sp:LaxTsLast ... /> |  
1977     ...  
1978   </wsp:Policy>  
1979   ...  
1980 </sp:Layout>
```

1981

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

1983 /sp:Layout

1984 This identifies a Layout assertion.

1985 /sp:Layout/wsp:Policy

1986 This REQUIRED element contains one or more policy assertions that indicate the specific security
1987 header layout to use.

1988 /sp:Layout/wsp:Policy/sp:Strict

1989 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1990 property is set to 'Strict'.

1991 /sp:Layout/wsp:Policy/sp:Lax

1992 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1993 property is set to 'Lax'.

1994 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1995 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
1996 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1997 'true' by the presence of an sp:IncludeTimestamp assertion.

1998 /sp:Layout/wsp:Policy/sp:LaxTsLast

1999 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]
2000 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
2001 'true' by the presence of an sp:IncludeTimestamp assertion.

2002 7.3 TransportBinding Assertion

2003 The TransportBinding assertion is used in scenarios in which message protection and security correlation
2004 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
2005 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
2006 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
2007 MUST apply to [Endpoint Policy Subject].

2008 Syntax

```
2009 <sp:TransportBinding xmlns:sp="..." ... >
2010   <wsp:Policy xmlns:wsp="...">
2011     <sp:TransportToken ... >
2012       <wsp:Policy> ... </wsp:Policy>
2013       ...
2014     </sp:TransportToken>
2015     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2016     <sp:Layout ... > ... </sp:Layout> ?
2017     <sp:IncludeTimestamp ... /> ?
2018     ...
2019   </wsp:Policy>
2020   ...
2021 </sp:TransportBinding>
```

2022

2023 The following describes the attributes and elements listed in the schema outlined above:

2024 /sp:TransportBinding

2025 This identifies a TransportBinding assertion.

2026 /sp:TransportBinding/wsp:Policy

2027 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
2028 assertion.

2029 /sp:TransportBinding/wsp:Policy/sp:TransportToken

2030 This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.
2031 The specified token populates the [Transport Token] property and indicates how the transport is
2032 secured.

2033 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

2034 This indicates a nested policy that identifies the type of Transport Token to use.

2035 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

2036 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
2037 Suite] property. See Section 6.1 for more details.

2038 /sp:TransportBinding/wsp:Policy/sp:Layout

2039 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
2040 Header Layout] property. See Section 6.7 for more details.

2041 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

2042 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
2043 to 'true'.

2044 7.4 SymmetricBinding Assertion

2045 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
2046 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
2047 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
2048 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
2049 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
2050 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
2051 properties and is used as the basis for both encryption and signature in both directions. This assertion
2052 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

2053 Syntax

```
2054 <sp:SymmetricBinding xmlns:sp="..." ... >  
2055   <wsp:Policy xmlns:wsp="...">  
2056     (  
2057       <sp:EncryptionToken ... >  
2058         <wsp:Policy> ... </wsp:Policy>  
2059       </sp:EncryptionToken>  
2060       <sp:SignatureToken ... >  
2061         <wsp:Policy> ... </wsp:Policy>  
2062       </sp:SignatureToken>  
2063     ) | (  
2064       <sp:ProtectionToken ... >  
2065         <wsp:Policy> ... </wsp:Policy>  
2066       </sp:ProtectionToken>  
2067     )  
2068     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
2069     <sp:Layout ... > ... </sp:Layout> ?  
2070     <sp:IncludeTimestamp ... /> ?  
2071     <sp:EncryptBeforeSigning ... /> ?  
2072     <sp:EncryptSignature ... /> ?  
2073     <sp:ProtectTokens ... /> ?  
2074     <sp:OnlySignEntireHeadersAndBody ... /> ?  
2075     ...  
2076   </wsp:Policy>  
2077   ...  
2078 </sp:SymmetricBinding>
```

2079
2080 The following describes the attributes and elements listed in the schema outlined above:

2081 /sp:SymmetricBinding
 2082 This identifies a SymmetricBinding assertion.

2083 /sp:SymmetricBinding/wsp:Policy
 2084 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
 2085 assertion.

2086 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
 2087 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption
 2088 Token. The specified token populates the [Encryption Token] property and is used for encryption.
 2089 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

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

2092 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
 2093 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.
 2094 The specified token populates the [Signature Token] property and is used for the message
 2095 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
 2096 specified.

2097 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
 2098 The policy contained here MUST identify exactly one token to use for signatures.

2099 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
 2100 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.
 2101 The specified token populates the [Encryption Token] and [Signature Token properties] and is
 2102 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
 2103 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
 2104 specified.

2105 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
 2106 The policy contained here MUST identify exactly one token to use for protection.

2107 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2108 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
 2109 Suite] property. See Section 6.1 for more details.

2110 /sp:SymmetricBinding/wsp:Policy/sp:Layout
 2111 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
 2112 Header Layout] property. See Section 6.7 for more details.

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

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

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

2122 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
 2123 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
 2124 set to 'true'.

2125 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
2126 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
2127 Signatures] property is set to 'true'.

2128 7.5 AsymmetricBinding Assertion

2129 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means
2130 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly
2131 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and
2132 signature. However it is also common practice to use distinct keys for encryption and signature, because
2133 of their different lifecycles.

2134
2135 This binding enables either of these practices by means of four binding specific token properties: [Initiator
2136 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption
2137 Token].

2138
2139 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator
2140 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient
2141 Encryption Token] will both refer to the same token.

2142
2143 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator
2144 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient
2145 Encryption Token] will refer to different tokens.

2146
2147 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the
2148 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response
2149 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the
2150 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for
2151 the message encryption from initiator to the recipient. Note that in each case, the token is associated with
2152 the party (initiator or recipient) who knows the secret.

2153 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy
2154 Subject].

2155 Syntax

```
2156 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2157   <wsp:Policy xmlns:wsp="...">  
2158     (  
2159       <sp:InitiatorToken>  
2160         <wsp:Policy> ... </wsp:Policy>  
2161       </sp:InitiatorToken>  
2162     ) | (  
2163       <sp:InitiatorSignatureToken>  
2164         <wsp:Policy> ... </wsp:Policy>  
2165       </sp:InitiatorSignatureToken>  
2166       <sp:InitiatorEncryptionToken>  
2167         <wsp:Policy> ... </wsp:Policy>  
2168       </sp:InitiatorEncryptionToken>  
2169     )  
2170     (  
2171       <sp:RecipientToken>  
2172         <wsp:Policy> ... </wsp:Policy>  
2173       </sp:RecipientToken>  
2174     ) | (  
2175
```

```

2175     <sp:RecipientSignatureToken>
2176         <wsp:Policy> ... </wsp:Policy>
2177     </sp:RecipientSignatureToken>
2178     <sp:RecipientEncryptionToken>
2179         <wsp:Policy> ... </wsp:Policy>
2180     </sp:RecipientEncryptionToken>
2181 )
2182 <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2183 <sp:Layout ... > ... </sp:Layout> ?
2184 <sp:IncludeTimestamp ... /> ?
2185 <sp:EncryptBeforeSigning ... /> ?
2186 <sp:EncryptSignature ... /> ?
2187 <sp:ProtectTokens ... /> ?
2188 <sp:OnlySignEntireHeadersAndBody ... /> ?
2189 ...
2190 </wsp:Policy>
2191 ...
2192 </sp:AsymmetricBinding>

```

2193

2194 The following describes the attributes and elements listed in the schema outlined above:

2195 /sp:AsymmetricBinding

2196 This identifies a AsymmetricBinding assertion.

2197 /sp:AsymmetricBinding/wsp:Policy

2198 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
2199 assertion.

2200 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2201 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.
2202 The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
2203 properties and is used for the message signature from initiator to recipient, and encryption from
2204 recipient to initiator.

2205 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2206 The policy contained here MUST identify one or more token assertions.

2207 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2208 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2209 Signature Token. The specified token populates the [Initiator Signature Token] property and is
2210 used for the message signature from initiator to recipient.

2211 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2212 The policy contained here MUST identify one or more token assertions.

2213 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2214 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator
2215 Encryption Token. The specified token populates the [Initiator Encryption Token] property and is
2216 used for the message encryption from recipient to initiator.

2217 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2218 The policy contained here MUST identify one or more token assertions.

2219 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2220 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.
2221 The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
2222 property and is used for encryption from initiator to recipient, and for the message signature from
2223 recipient to initiator.

2224 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy
 2225 The policy contained here MUST identify one or more token assertions.

2226 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken
 2227 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
 2228 Signature Token. The specified token populates the [Recipient Signature Token] property and is
 2229 used for the message signature from recipient to initiator.

2230 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
 2231 The policy contained here MUST identify one or more token assertions.

2232 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
 2233 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient
 2234 Encryption Token. The specified token populates the [Recipient Encryption Token] property and
 2235 is used for the message encryption from initiator to recipient.

2236 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
 2237 The policy contained here MUST identify one or more token assertions.

2238 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2239 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
 2240 Suite] property. See Section 6.1 for more details.

2241 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
 2242 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
 2243 Header Layout] property. See Section 6.7 for more details.

2244 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 2245 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
 2246 to 'true'.

2247 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 2248 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is
 2249 set to 'EncryptBeforeSigning'.

2250 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
 2251 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]
 2252 property is set to 'true'.

2253 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
 2254 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is
 2255 set to 'true'.

2256 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 2257 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body
 2258 Signatures] property is set to 'true'.

8 Supporting Tokens

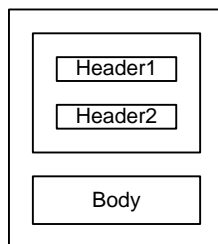
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. In case of Transport Binding the message is signed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

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

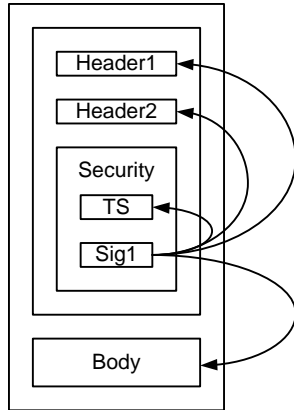
In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine which order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

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

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

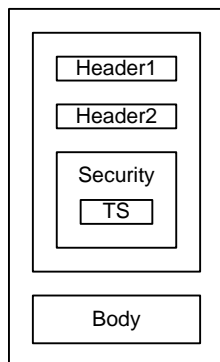


Even before any supporting tokens are added, each binding requires that the message is signed using a token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection. The signature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



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

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



8.1 SupportingTokens Assertion

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

Syntax

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


2323
2324
2325
2326
2327
2328
2329
2330

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

2331

2332 The following describes the attributes and elements listed in the schema outlined above:

2333 /sp:SupportingTokens

2334 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2335 Tokens] property.

2336 /sp:SupportingTokens/wsp:Policy

2337 This describes additional requirements for satisfying the SupportingTokens assertion.

2338 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2339 The policy MUST identify one or more token assertions.

2340 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2341 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2342 describes the algorithms to use for cryptographic operations performed with the tokens identified
2343 by this policy assertion.

2344 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2345 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2346 and describes additional message parts that MUST be included in the signature generated with
2347 the token identified by this policy assertion.

2348 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2349 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2350 and describes additional message elements that MUST be included in the signature generated
2351 with the token identified by this policy assertion.

2352 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2353 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2354 and describes additional message parts that MUST be encrypted using the token identified by
2355 this policy assertion.

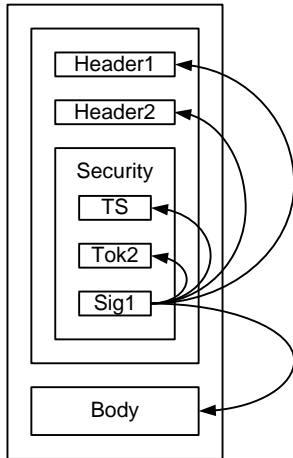
2356 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2357 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2358 and describes additional message elements that MUST be encrypted using the token identified
2359 by this policy assertion.

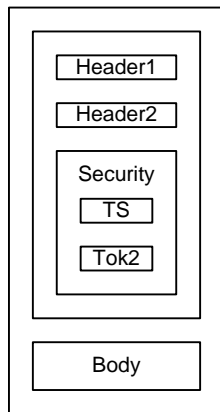
2360 8.2 SignedSupportingTokens Assertion

2361 Signed tokens are included in the “message signature” as defined above and MAY OPTIONALLY include
2362 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
2363 (Tok2) is signed by the message signature (Sig1):

2364



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



Syntax

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

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

/sp:SignedSupportingTokens

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

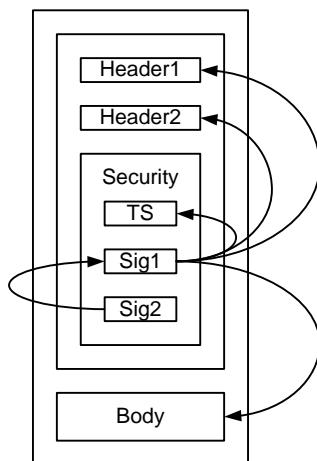
/sp:SignedSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the SignedSupportingTokens assertion.

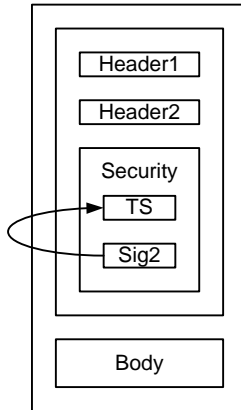
2391 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 2392 The policy MUST identify one or more token assertions.
 2393 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 2394 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
 2395 describes the algorithms to use for cryptographic operations performed with the tokens identified
 2396 by this policy assertion.
 2397 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 2398 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
 2399 and describes additional message parts that MUST be included in the signature generated with
 2400 the token identified by this policy assertion.
 2401 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 2402 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
 2403 and describes additional message elements that MUST be included in the signature generated
 2404 with the token identified by this policy assertion.
 2405 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 2406 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
 2407 and describes additional message parts that MUST be encrypted using the token identified by
 2408 this policy assertion.
 2409 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 2410 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
 2411 and describes additional message elements that MUST be encrypted using the token identified
 2412 by this policy assertion.

2413 8.3 EndorsingSupportingTokens Assertion

2414 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 2415 produced from the message signature and MAY OPTIONALLY include additional message parts to sign
 2416 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message
 2417 signature (Sig1):
 2418



2419
 2420 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 2421 below:
 2422



2423

2424 Syntax

```

2425 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2426   <wsp:Policy xmlns:wsp="...">
2427     [Token Assertion]+
2428     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2429     (
2430       <sp:SignedParts ... > ... </sp:SignedParts> |
2431       <sp:SignedElements ... > ... </sp:SignedElements> |
2432       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2433       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2434     ) *
2435     ...
2436   </wsp:Policy>
2437   ...
2438 </sp:EndorsingSupportingTokens>

```

2439

2440 The following describes the attributes and elements listed in the schema outlined above:

2441 /sp:EndorsingSupportingTokens

2442 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
2443 [Endorsing Supporting Tokens] property.

2444 /sp:EndorsingSupportingTokens/wsp:Policy

2445 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2446 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2447 The policy MUST identify one or more token assertions.

2448 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2449 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
2450 describes the algorithms to use for cryptographic operations performed with the tokens identified
2451 by this policy assertion.

2452 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2453 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
2454 and describes additional message parts that MUST be included in the signature generated with
2455 the token identified by this policy assertion.

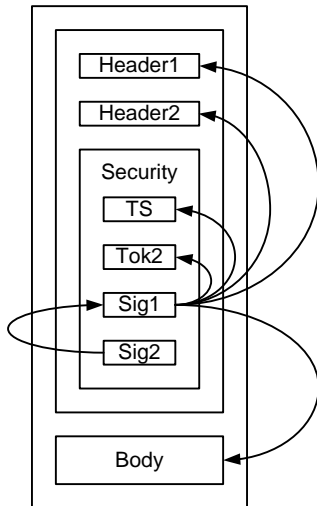
2456 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2457 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
2458 and describes additional message elements that MUST be included in the signature generated
2459 with the token identified by this policy assertion.

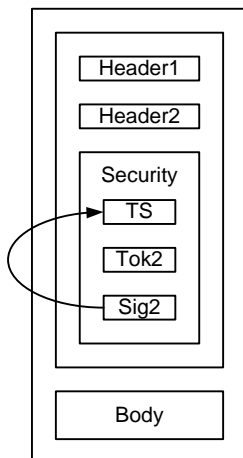
2460 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
2461 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
2462 and describes additional message parts that MUST be encrypted using the token identified by
2463 this policy assertion.
2464 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
2465 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
2466 and describes additional message elements that MUST be encrypted using the token identified
2467 by this policy assertion.

2468 8.4 SignedEndorsingSupportingTokens Assertion

2469 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
2470 and are themselves signed by that message signature, that is both tokens (the token used for the
2471 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY
2472 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
2473 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
2474 message signature (Sig1):
2475



2476
2477 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
2478 SHOULD cover the message timestamp as illustrated below:
2479



Syntax

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

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

/sp:SignedEndorsingSupportingTokens

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

/sp:SignedEndorsingSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

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

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

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

8.5 SignedEncryptedSupportingTokens Assertion

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

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

8.6 EncryptedSupportingTokens Assertion

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

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

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

8.7 EndorsingEncryptedSupportingTokens Assertion

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

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

8.8 SignedEndorsingEncryptedSupportingTokens Assertion

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

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

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

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

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

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

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

8.10 Example

Example policy containing supporting token assertions:

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

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

9 WSS: SOAP Message Security Options

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

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

Note: This approach is chosen because:

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

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

WSS: SOAP Message Security 1.0 Properties

[Direct References]

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

[Key Identifier References]

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

[Issuer Serial References]

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

[External URI References]

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

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

[Embedded Token References]

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

WSS: SOAP Message Security 1.1 Properties

[Thumbprint References]

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

[EncryptedKey References]

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

[Signature Confirmation]

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

9.1 Wss10 Assertion

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

Syntax

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

2705 The following describes the attributes and elements listed in the schema outlined above:
2706 /sp:Wss10
2707 This identifies a WSS10 assertion.
2708 /sp:Wss10/wsp:Policy
2709 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.
2710 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier
2711 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]
2712 property is set to 'true'.
2713 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial
2714 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
2715 property is set to 'true'.
2716 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI
2717 This OPTIONAL element is a policy assertion indicates that the [External URI References]
2718 property is set to 'true'.
2719 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken
2720 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2721 property is set to 'true'.

2722 9.2 Wss11 Assertion

2723 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
2724 supported.

2725 Syntax

```
2726 <sp:Wss11 xmlns:sp="..." ... >  
2727   <wsp:Policy xmlns:wsp="...">  
2728     <sp:MustSupportRefKeyIdentifier ... /> ?  
2729     <sp:MustSupportRefIssuerSerial ... /> ?  
2730     <sp:MustSupportRefExternalURI ... /> ?  
2731     <sp:MustSupportRefEmbeddedToken ... /> ?  
2732     <sp:MustSupportRefThumbprint ... /> ?  
2733     <sp:MustSupportRefEncryptedKey ... /> ?  
2734     <sp:RequireSignatureConfirmation ... /> ?  
2735     ...  
2736   </wsp:Policy>  
2737 </sp:Wss11>
```

2738
2739 The following describes the attributes and elements listed in the schema outlined above:

2740 /sp:Wss11
2741 This identifies an WSS11 assertion.
2742 /sp:Wss11/wsp:Policy
2743 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.
2744 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier
2745 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]
2746 property is set to 'true'.
2747 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial
2748 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]
2749 property is set to 'true'.

2750 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2751 This OPTIONAL element is a policy assertion indicates that the [External URI References]
2752 property is set to 'true'.
2753 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2754 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]
2755 property is set to 'true'.
2756 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2757 This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property
2758 is set to 'true'.
2759 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2760 This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]
2761 property is set to 'true'.
2762 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2763 This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property
2764 is set to 'true'.

10 WS-Trust Options

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

WS-Trust Properties

[Client Challenge]

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

[Server Challenge]

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

[Client Entropy]

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

[Server Entropy]

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

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

[Issued Tokens]

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

[Collection]

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

[Scope Policy 1.5]

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

[Interactive Challenge]

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

10.1 Trust13 Assertion

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

Syntax

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

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

/sp:Trust13

This identifies a Trust13 assertion.

/sp:Trust13/wsp:Policy

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

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

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

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

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

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

2870 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection
2871 This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to
2872 'true'.

2873 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo
2874 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor
2875 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2876 /sp:Trust13/wsp:Policy/sp13:ScopePolicy15
2877 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5]
2878 property is set to 'true'.

2879 /sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge
2880 This optional element is a policy assertion indicates that the [Interactive Challenge]
2881 property is set to 'true'.

11 Guidance on creating new assertions and assertion extensibility

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

11.1 General Design Points

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

11.2 Detailed Design Guidance

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

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

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

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

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

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

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

2926 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2927 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2928 was encoded into the assertion Qnames, each existing token assertion would require five variants, one
2929 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2930

2931 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2932 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2933 assertion is parameterized by the nested token version assertions. Policy matching can use these
2934 parameters to find matches between policies where the broad token type is support by both parties but
2935 they might not support the same specific versions.

2936

2937 Note, when designing assertions for new token types such assertions SHOULD allow the
2938 sp:IncludeToken attribute and SHOULD allow nested policy.

2939

12 Security Considerations

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

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

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

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

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

13 Conformance

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

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

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

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

A. Assertions and WS-PolicyAttachment

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

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

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

A.1.2 Token Assertions

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

A.1.3 WSS: SOAP Message Security 1.0 Assertions

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

A.1.4 WSS: SOAP Message Security 1.1 Assertions

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

A.1.5 Trust 1.0 Assertions

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

A.2 Operation Policy Subject Assertions

A.2.1 Security Binding Assertions

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

A.2.2 Supporting Token Assertions

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

3013	EndorsingSupportingTokens Assertion	(Section 8.3)
3014	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3015	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3016	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3017	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3018 **A.3 Message Policy Subject Assertions**

3019 **A.3.1 Supporting Token Assertions**

3020	SupportingTokens Assertion	(Section 8.1)
3021	SignedSupportingTokens Assertion	(Section 8.2)
3022	EndorsingSupportingTokens Assertion	(Section 8.3)
3023	SignedEndorsingSupportingTokens Assertion	(Section 8.4)
3024	SignedEncryptedSupportingTokens Assertion	(Section 8.5)
3025	EndorsingEncryptedSupportingTokens Assertion	(Section 8.6)
3026	SignedEndorsingEncryptedSupportingTokens Assertion	(Section 8.7)

3027 **A.3.2 Protection Assertions**

3028	SignedParts Assertion	(Section 4.1.1)
3029	SignedElements Assertion	(Section 4.1.2)
3030	EncryptedParts Assertion	(Section 4.2.1)
3031	EncryptedElements Assertion	(Section 4.2.2)
3032	ContentEncryptedElements Assertion	(Section 4.2.3)
3033	RequiredElements Assertion	(Section 4.3.1)
3034	RequiredParts Assertion	(Section 4.3.2)

3035 **A.4 Assertions With Undefined Policy Subject**

3036 The assertions listed in this section do not have a defined policy subject because they appear nested
3037 inside some other assertion which does have a defined policy subject. This list is derived from nested
3038 assertions in the specification that have independent sections. It is not a complete list of nested
3039 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
3040 additional nested assertions.

3041 **A.4.1 General Assertions**

3042	AlgorithmSuite Assertion	(Section 7.1)
3043	Layout Assertion	(Section 7.2)

3044 **A.4.2 Token Usage Assertions**

3045 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
3046 assertions.

3047 **A.4.3 Token Assertions**

3048	UsernameToken Assertion	(Section 5.3.1)
------	---	-----------------

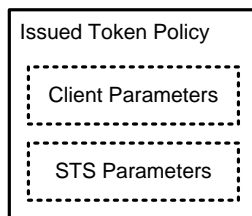
3049	IssuedToken Assertion	(Section 5.3.2)
3050	X509Token Assertion	(Section 5.3.3)
3051	KerberosToken Assertion	(Section 5.3.4)
3052	SpnegoContextToken Assertion	(Section 5.3.5)
3053	SecurityContextToken Assertion	(Section 5.3.6)
3054	SecureConversationToken Assertion	(Section 5.3.7)
3055	SamlToken Assertion	(Section 5.3.8)
3056	RelToken Assertion	(Section 5.3.9)
3057	HttpsToken Assertion	(Section 5.3.10)

B. Issued Token Policy

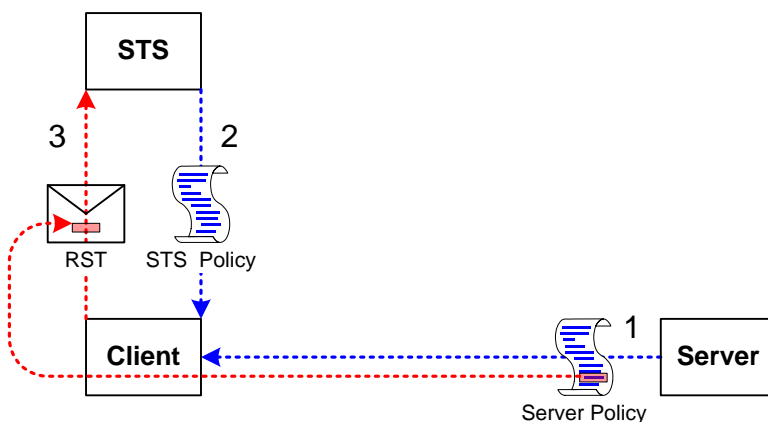
The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

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

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



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



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

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

3085 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The
3086 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along
3087 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
3088 request sent by the Client to the STS following the protocol defined in WS-Trust.
3089
3090 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)
3091 [Trust\]](#). All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship
3092 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>
```

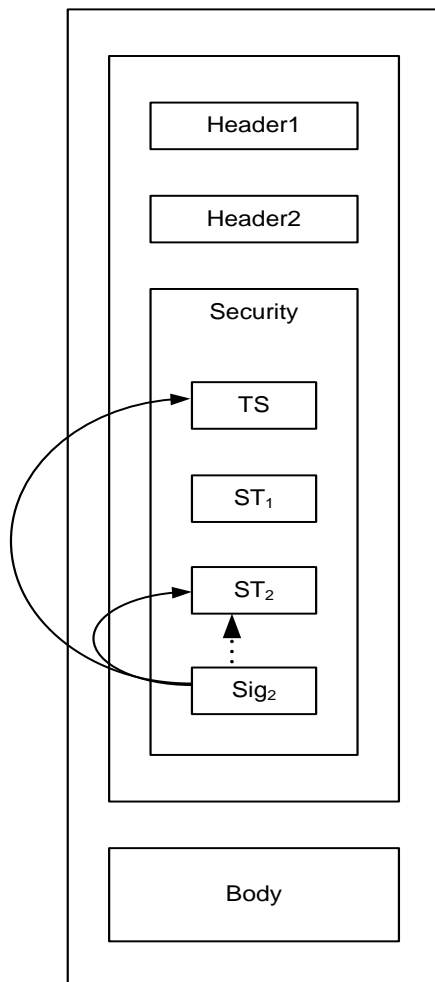
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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

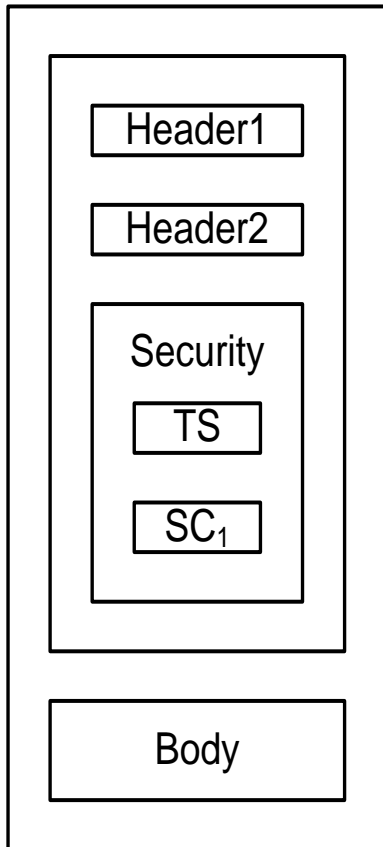
```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken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



3216

3217 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 3218 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 3219 message illustrated previously is included. In general, the ordering of the items in the security header
 3220 follows the most optimal layout for a receiver to process its contents.

3221 *Example:*

3222 Recipient to initiator message

```

3223 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3224   <S:Header>
3225     ...
3226     <wsse:Security>
3227       <wsu:Timestamp wsu:Id="timestamp">
3228         <wsu:Created>[datetime]</wsu:Created>
3229         <wsu:Expires>[datetime]</wsu:Expires>
3230       </wsu:Timestamp>
3231       <wsse11:SignatureConfirmation Value="..." />
3232     ...
3233   </wsse:Security>
3234   ...
3235 </S:Header>
3236 <S:Body>
3237   ...
3238 </S:Body>
3239 </S:Envelope>

```

3240 C.2 Symmetric Binding

3241 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

C.2.1 Policy

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

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

```

3298
3299 <!-- Example Message Policy -->
3300 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3301   <sp:SignedParts>
3302     <sp:Header Name="Header1" Namespace="..." />
3303     <sp:Header Name="Header2" Namespace="..." />
3304     <sp:Body/>
3305   </sp:SignedParts>
3306   <sp:EncryptedParts>
3307     <sp:Header Name="Header2" Namespace="..." />
3308     <sp:Body/>
3309   </sp:EncryptedParts>
3310 </wsp:Policy>

```

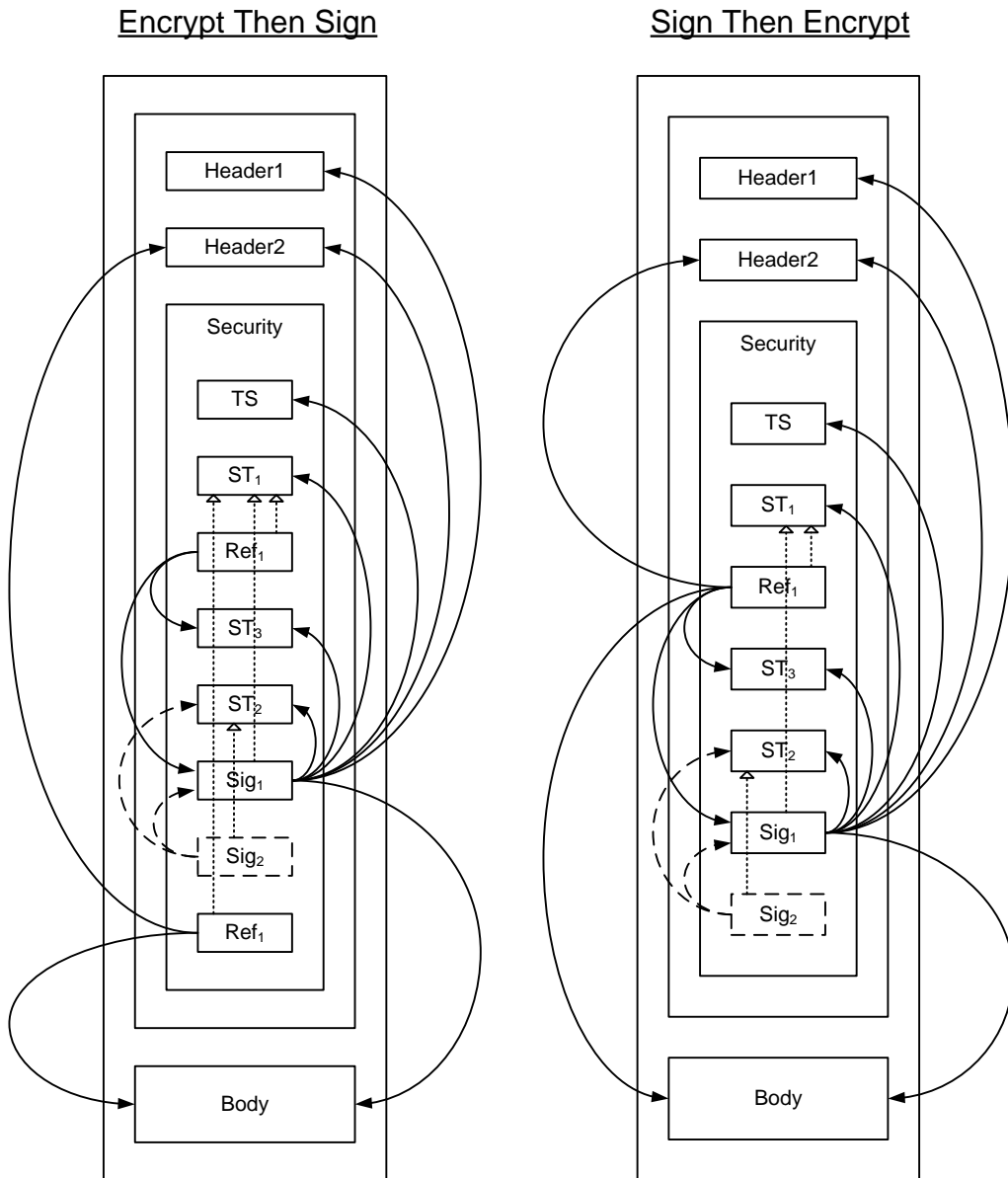
3311 This policy is used as the basis for the examples shown in the subsequent section describing the security
3312 header layout for this binding.

3313 C.2.2 Initiator to Recipient Messages

3314 Messages sent from initiator to recipient have the following layout for the security header:

- 3315 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3316 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
3317 `.../IncludeToken/Always`, then the [Encryption Token].
- 3318 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3319 Derived Key Token is used for encryption.
- 3320 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3321 reference list MUST include a reference to the message signature. If [Protection Order] is
3322 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3323 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3324 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3325 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3326 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
3327 `.../IncludeToken/Always`.
- 3328 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3329 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
3330 [Signature Token].
- 3331 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3332 Derived Key Token is used for signature.
- 3333 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3334 whether they are included in the message, and any message parts specified in SignedParts
3335 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3336 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3337 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3338 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3339 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3340 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3341 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3342 endorsing token, appears before the signature.
- 3343 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3344 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3345 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3346 above.

3347
 3348 The following diagram illustrates the security header layout for the initiator to recipient message:



3349
 3350 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 3351 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
 3352 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
 3353 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
 3354 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
 3355 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
 3356 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
 3357 process its contents.
 3358 *Example:*
 3359 Initiator to recipient message using EncryptBeforeSigning:

```
3360 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3361   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3362   xmlns:xenc="..." xmlns:ds="...">
3363   <S:Header>
3364     <x:Header1 wsu:Id="Header1" >
3365       ...
3366     </x:Header1>
3367
```



```

3368 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3369   <!-- Plaintext Header2
3370   <x:Header2 wsu:Id="Header2" >
3371     ...
3372   </x:Header2>
3373   -->
3374   ...
3375 </wsse1:EncryptedHeader>
3376 ...
3377 <wsse:Security>
3378   <wsu:Timestamp wsu:Id="Timestamp">
3379     <wsu:Created>...</wsu:Created>
3380     <wsu:Expires>...</wsu:Expires>
3381   </wsu:Timestamp>
3382   <saml:Assertion AssertionId="_SharedSecretToken" ...>
3383     ...
3384   </saml:Assertion>
3385   <xenc:ReferenceList>
3386     <xenc:DataReference URI="#enc_Signature" />
3387     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3388     ...
3389   </xenc:ReferenceList>
3390   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3391     <!-- Plaintext UsernameToken
3392     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3393       ...
3394     </wsse:UsernameToken>
3395     -->
3396     ...
3397     <ds:KeyInfo>
3398       <wsse:SecurityTokenReference>
3399         <wsse:Reference URI="#_SharedSecretToken" />
3400       </wsse:SecurityTokenReference>
3401     </ds:KeyInfo>
3402   </xenc:EncryptedData>
3403   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3404     ...
3405   </wsse:BinarySecurityToken>
3406   <xenc:EncryptedData ID="enc_Signature">
3407     <!-- Plaintext Signature
3408     <ds:Signature Id="Signature">
3409       <ds:SignedInfo>
3410         <ds:References>
3411           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3412           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3413           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3414           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3415           <ds:Reference URI="#Header1" >...</ds:Reference>
3416           <ds:Reference URI="#Header2" >...</ds:Reference>
3417           <ds:Reference URI="#Body" >...</ds:Reference>
3418         </ds:References>
3419       </ds:SignedInfo>
3420     <ds:SignatureValue>...</ds:SignatureValue>
3421     <ds:KeyInfo>
3422       <wsse:SecurityTokenReference>
3423         <wsse:Reference URI="#_SharedSecretToken" />
3424       </wsse:SecurityTokenReference>
3425     </ds:KeyInfo>
3426   </ds:Signature>
3427   -->
3428   ...
3429   <ds:KeyInfo>
3430     <wsse:SecurityTokenReference>
3431       <wsse:Reference URI="#_SharedSecretToken" />

```

```

3432         </wsse:SecurityTokenReference>
3433     </ds:KeyInfo>
3434 </xenc:EncryptedData>
3435 <ds:Signature>
3436     <ds:SignedInfo>
3437         <ds:References>
3438             <ds:Reference URI="#Signature" >...</ds:Reference>
3439             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3440         </ds:References>
3441     </ds:SignedInfo>
3442 <ds:SignatureValue>...</ds:SignatureValue>
3443 <ds:KeyInfo>
3444     <wsse:SecurityTokenReference>
3445         <wsse:Reference URI="#SomeSupportingToken" />
3446     </wsse:SecurityTokenReference>
3447 </ds:KeyInfo>
3448 </ds:Signature>
3449 <xenc:ReferenceList>
3450     <xenc:DataReference URI="#enc_Body" />
3451     <xenc:DataReference URI="#enc_Header2" />
3452     ...
3453 </xenc:ReferenceList>
3454 </wsse:Security>
3455 </S:Header>
3456 <S:Body wsu:Id="Body">
3457     <xenc:EncryptedData Id="enc_Body">
3458         ...
3459         <ds:KeyInfo>
3460             <wsse:SecurityTokenReference>
3461                 <wsse:Reference URI="#_SharedSecretToken" />
3462             </wsse:SecurityTokenReference>
3463         </ds:KeyInfo>
3464     </xenc:EncryptedData>
3465 </S:Body>
3466 </S:Envelope>

```

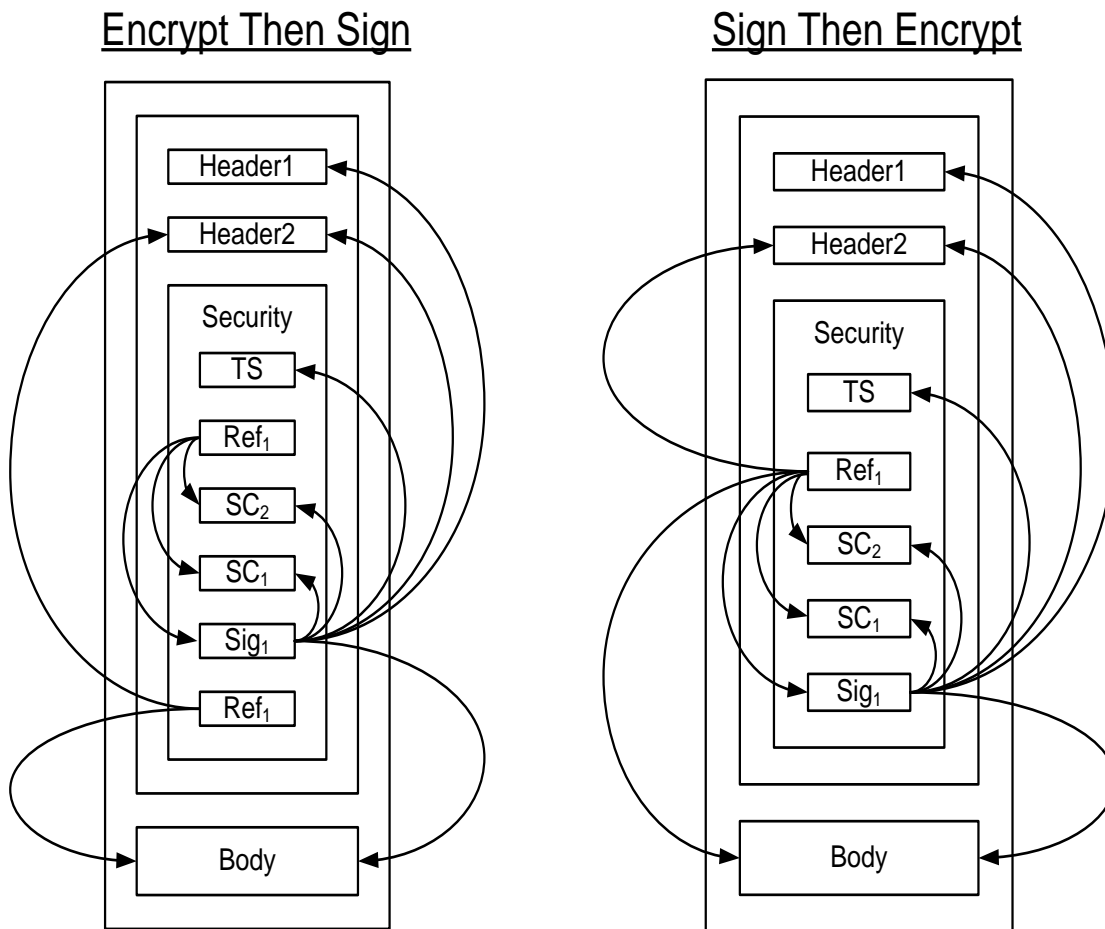
C.2.3 Recipient to Initiator Messages

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 `wsse11: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 `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.
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 `wssell: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 `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.

Example:

3508 Recipient to initiator message using EncryptBeforeSigning:

```
3509 <S:Envelope>
3510   <S:Header>
3511     <x:Header1 wsu:Id="Header1" >
3512       ...
3513     </x:Header1>
3514     <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3515       <!-- Plaintext Header2
3516       <x:Header2 wsu:Id="Header2" >
3517         ...
3518       </x:Header2>
3519       -->
3520     </wsse1:EncryptedHeader>
3521     ...
3522   <wsse:Security>
3523     <wsu:Timestamp wsu:Id="Timestamp">
3524       <wsu:Created>...</wsu:Created>
3525       <wsu:Expires>...</wsu:Expires>
3526     </wsu:Timestamp>
3527     <xenc:ReferenceList>
3528       <xenc:DataReference URI="#enc_Signature" />
3529       <xenc:DataReference URI="#enc_SigConf1" />
3530       <xenc:DataReference URI="#enc_SigConf2" />
3531       ...
3532     </xenc:ReferenceList>
3533     <xenc:EncryptedData ID="enc_SigConf1" >
3534       <!-- Plaintext SignatureConfirmation
3535       <wsse1:SignatureConfirmation wsu:Id="SigConf1" >
3536         ...
3537       </wsse1:SignatureConfirmation>
3538       -->
3539     </xenc:EncryptedData>
3540     <xenc:EncryptedData ID="enc_SigConf2" >
3541       <!-- Plaintext SignatureConfirmation
3542       <wsse1:SignatureConfirmation wsu:Id="SigConf2" >
3543         ...
3544       </wsse1:SignatureConfirmation>
3545       -->
3546     </xenc:EncryptedData>
```

```

3550
3551 <xenc:EncryptedData Id="enc_Signature">
3552   <!-- Plaintext Signature
3553   <ds:Signature Id="Signature">
3554     <ds:SignedInfo>
3555       <ds:References>
3556         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3557         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3558         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3559         <ds:Reference URI="#Header1" >...</ds:Reference>
3560         <ds:Reference URI="#Header2" >...</ds:Reference>
3561         <ds:Reference URI="#Body" >...</ds:Reference>
3562       </ds:References>
3563     </ds:SignedInfo>
3564     <ds:SignatureValue>...</ds:SignatureValue>
3565     <ds:KeyInfo>
3566       <wsse:SecurityTokenReference>
3567         <wsse:Reference URI="#_SomeIssuedToken" />
3568       </wsse:SecurityTokenReference>
3569     </ds:KeyInfo>
3570   </ds:Signature>
3571   -->
3572 </xenc:EncryptedData>
3573   ...
3574 <ds:KeyInfo>
3575   <wsse:SecurityTokenReference>
3576     <wsse:Reference URI="#_SomeIssuedToken" />
3577   </wsse:SecurityTokenReference>
3578 </ds:KeyInfo>
3579 <xenc:EncryptedData>
3580 <xenc:ReferenceList>
3581   <xenc:DataReference URI="#enc_Body" />
3582   <xenc:DataReference URI="#enc_Header2" />
3583   ...
3584 </xenc:ReferenceList>
3585 </xenc:EncryptedData>
3586 </wsse:Security>
3587 </S:Header>
3588 <S:Body wsu:Id="Body">
3589   <xenc:EncryptedData Id="enc_Body">
3590     ...
3591     <ds:KeyInfo>
3592       <wsse:SecurityTokenReference>
3593         <wsse:Reference URI="#_SomeIssuedToken" />
3594       </wsse:SecurityTokenReference>
3595     </ds:KeyInfo>
3596   </xenc:EncryptedData>
3597 </S:Body>
3598 </S:Envelope>

```

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>
```

```

3661 <!-- Example Message Policy -->
3662 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3663   <sp:SignedParts>
3664     <sp:Header Name="Header1" Namespace="..." />
3665     <sp:Header Name="Header2" Namespace="..." />
3666     <sp:Body/>
3667   </sp:SignedParts>
3668   <sp:EncryptedParts>
3669     <sp:Header Name="Header2" Namespace="..." />
3670     <sp:Body/>
3671   </sp:EncryptedParts>
3672 </wsp:All>

```

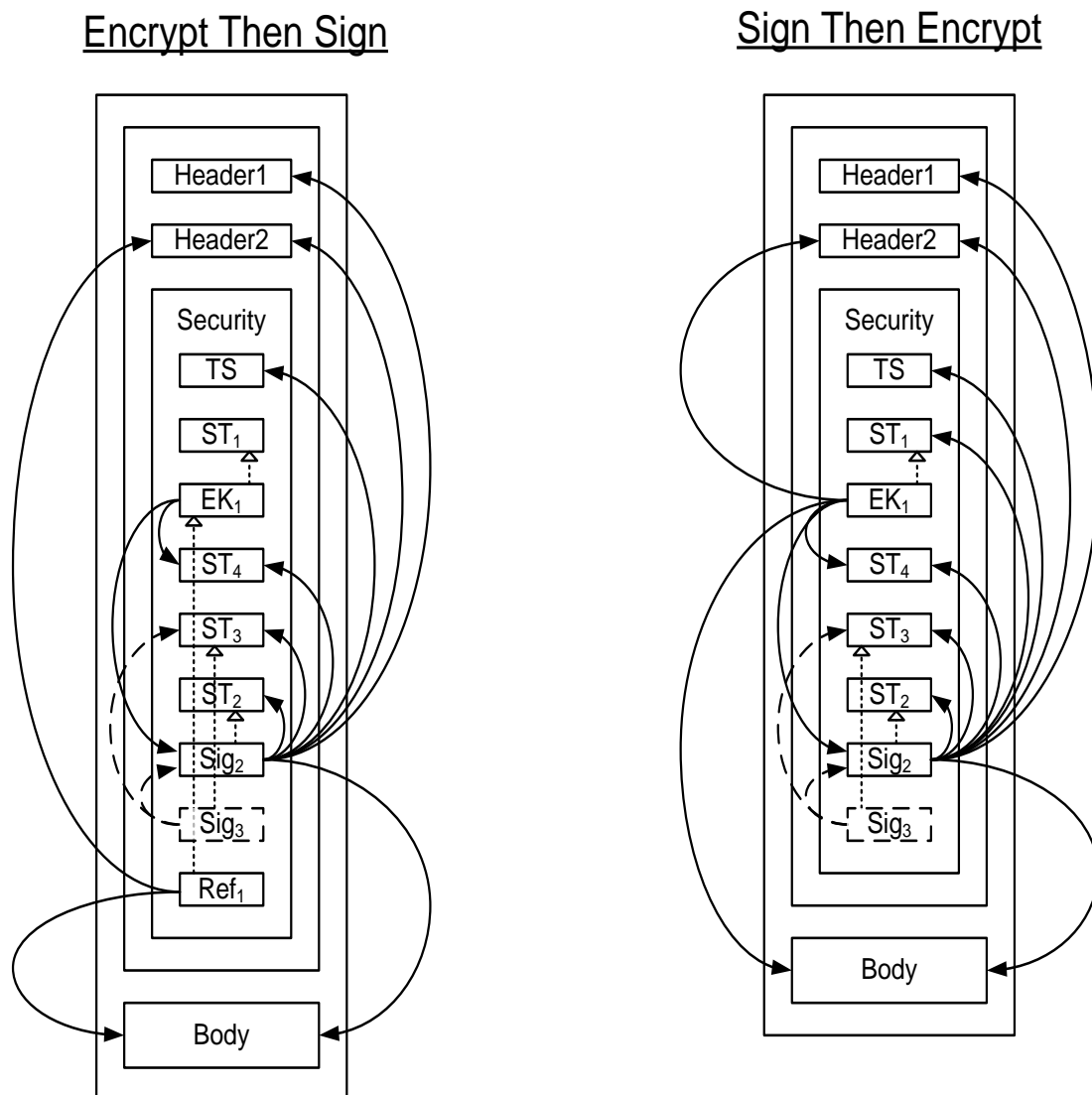
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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.

3709 The following diagram illustrates the security header layout for the initiator to recipient messages:



3710
3711 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂
3712 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate
3713 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as
3714 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled E_{K1} indicate references
3715 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left
3716 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the
3717 encrypted key labeled E_{K1}. The dotted arrows inside the box labeled Security indicate the token used as
3718 the basis for each cryptographic operation. In general, the ordering of the items in the security header
3719 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
3720 ordering are described in Appendix C.

3721
3722 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
3723 key contains an external reference to the token containing the encryption key. The diagram illustrates
3724 how one might attach a security token related to the encrypted key for completeness. One possible use-

3725 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3726 wishes to include the encryption token in the message signature.

3727 Initiator to recipient message *Example*

3728 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3729     xmlns:wsse1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3730 <S:Header>
3731   <x:Header1 wsu:Id="Header1" >
3732     ...
3733   </x:Header1>
3734   <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3735     <!-- Plaintext Header2
3736     <x:Header2 wsu:Id="Header2" >
3737       ...
3738     </x:Header2>
3739     -->
3740     ...
3741   </wsse1:EncryptedHeader>
3742   ...
3743   <wsse:Security>
3744     <wsu:Timestamp wsu:Id="Timestamp">
3745       <wsu:Created>...</wsu:Created>
3746       <wsu:Expires>...</wsu:Expires>
3747     </wsu:Timestamp>
3748     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3749       ...
3750     </wsse:BinarySecurityToken>
3751     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3752       ...
3753       <xenc:ReferenceList>
3754         <xenc:DataReference URI="#enc_Signature" />
3755         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3756         ...
3757       </xenc:ReferenceList>
3758     </xenc:EncryptedKey>
3759     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3760       <!-- Plaintext UsernameToken
3761       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3762         ...
3763       </wsse:UsernameToken>
3764       -->
3765       ...
3766     </xenc:EncryptedData>
3767     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3768       ...
3769     </wsse:BinarySecurityToken>
3770     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3771       ...
3772     </wsse:BinarySecurityToken>
3773     <xenc:EncryptedData ID="enc_Signature">
3774       <!-- Plaintext Signature
3775       <ds:Signature Id="Signature">
3776         <ds:SignedInfo>
3777           <ds:References>
3778             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3779             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3780             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3781             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3782             <ds:Reference URI="#Header1" >...</ds:Reference>
3783             <ds:Reference URI="#Header2" >...</ds:Reference>
3784             <ds:Reference URI="#Body" >...</ds:Reference>
3785           </ds:References>
3786         </ds:SignedInfo>
3787         <ds:SignatureValue>...</ds:SignatureValue>
3788       </ds:Signature>
3789       <ds:KeyInfo>
3790         <wsse:SecurityTokenReference>
3791           <wsse:Reference URI="#InitiatorToken" />
3792         </wsse:SecurityTokenReference>
3793       </ds:KeyInfo>

```

```

3793     </ds:Signature>
3794     -->
3795     ...
3796 </xenc:EncryptedData>
3797 <ds:Signature>
3798   <ds:SignedInfo>
3799     <ds:References>
3800       <ds:Reference URI="#Signature" >...</ds:Reference>
3801       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3802     </ds:References>
3803   </ds:SignedInfo>
3804   <ds:SignatureValue>...</ds:SignatureValue>
3805   <ds:KeyInfo>
3806     <wsse:SecurityTokenReference>
3807       <wsse:Reference URI="#SomeSupportingToken" />
3808     </wsse:SecurityTokenReference>
3809   </ds:KeyInfo>
3810 </ds:Signature>
3811 <xenc:ReferenceList>
3812   <xenc:DataReference URI="#enc_Body" />
3813   <xenc:DataReference URI="#enc_Header2" />
3814   ...
3815 </xenc:ReferenceList>
3816 </wsse:Security>
3817 </S:Header>
3818 <S:Body wsu:Id="Body">
3819   <xenc:EncryptedData Id="enc_Body">
3820     ...
3821     <ds:KeyInfo>
3822       <wsse:SecurityTokenReference>
3823         <wsse:Reference URI="#RecipientEncryptedKey" />
3824       </wsse:SecurityTokenReference>
3825     </ds:KeyInfo>
3826   </xenc:EncryptedData>
3827 </S:Body>
3828 </S:Envelope>

```

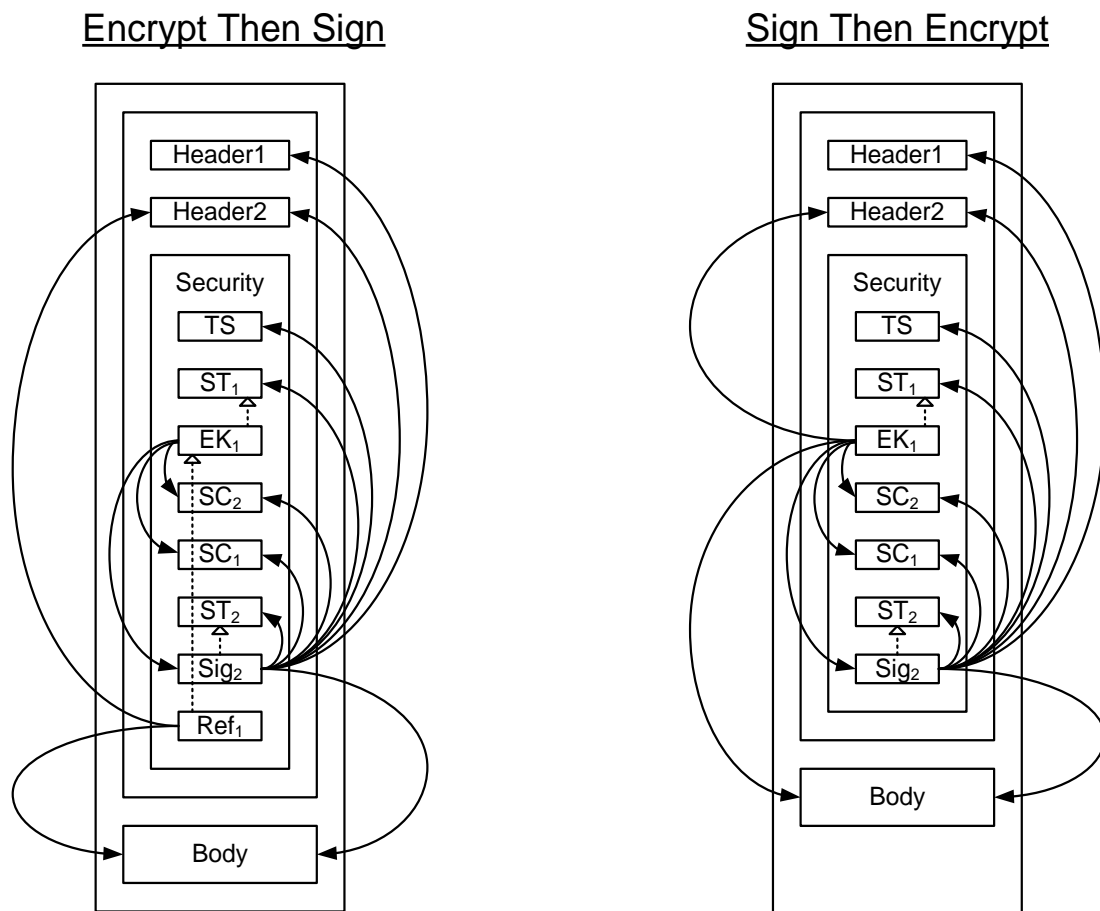
3829 C.3.3 Recipient to Initiator Messages

3830 Messages sent from recipient to initiator have the following layout:

- 3831 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3832 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3833 `.../IncludeToken/Always`, then the [Initiator Token].
- 3834 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3835 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for
3836 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a
3837 reference to all the message parts specified in EncryptedParts assertions in the policy. If
3838 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3839 message signature from 6 below, if any and references to the
3840 `wssell:SignatureConfirmation` elements from 4 below, if any.
- 3841 4. If [Signature Confirmation] is 'true', then a `wssell:SignatureConfirmation` element for each
3842 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3843 in the corresponding message from the initiator to the recipient, then a
3844 `wssell:SignatureConfirmation` element with no Value attribute.
- 3845 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3846 `.../IncludeToken/Always`, then the [Recipient Token].

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

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



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

Recipient to initiator message *Example*:

```

3870 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3871     xmlns:wssell="..." xmlns:wsse="..."
3872     xmlns:xenc="..." xmlns:ds="...">
3873 <S:Header>
3874     <x:Header1 wsu:Id="Header1" >
3875         ...
3876     </x:Header1>
3877     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3878         <!-- Plaintext Header2
3879         <x:Header2 wsu:Id="Header2" >
3880             ...
3881         </x:Header2>
3882         -->
3883         ...
3884     </wssell:EncryptedHeader>
3885     ...
3886 <wsse:Security>
3887     <wsu:Timestamp wsu:Id="Timestamp">
3888         <wsu:Created>...</wsu:Created>
3889         <wsu:Expires>...</wsu:Expires>
3890     </wsu:Timestamp>
3891     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3892         ...
3893     </wsse:BinarySecurityToken>
3894     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3895         ...
3896         <xenc:ReferenceList>
3897             <xenc:DataReference URI="#enc_Signature" />
3898             <xenc:DataReference URI="#enc_SigConf1" />
3899             <xenc:DataReference URI="#enc_SigConf2" />
3900             ...
3901         </xenc:ReferenceList>
3902     </xenc:EncryptedKey>
3903     <xenc:EncryptedData ID="enc_SigConf2" >
3904         <!-- Plaintext SignatureConfirmation
3905         <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3906             ...
3907         </wssell:SignatureConfirmation>
3908         -->
3909         ...
3910     </xenc:EncryptedData>
3911     <xenc:EncryptedData ID="enc_SigConf1" >
3912         <!-- Plaintext SignatureConfirmation
3913         <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3914             ...
3915         </wssell:SignatureConfirmation>
3916         -->
3917         ...
3918     </xenc:EncryptedData>
3919     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3920         ...
3921     </wsse:BinarySecurityToken>
3922

```

```

3923 <xenc:EncryptedData ID="enc_Signature">
3924 <!-- Plaintext Signature
3925 <ds:Signature Id="Signature">
3926 <ds:SignedInfo>
3927 <ds:References>
3928 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3929 <ds:Reference URI="#SigConf1" >...</ds:Reference>
3930 <ds:Reference URI="#SigConf2" >...</ds:Reference>
3931 <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3932 <ds:Reference URI="#Header1" >...</ds:Reference>
3933 <ds:Reference URI="#Header2" >...</ds:Reference>
3934 <ds:Reference URI="#Body" >...</ds:Reference>
3935 </ds:References>
3936 </ds:SignedInfo>
3937 <ds:SignatureValue>...</ds:SignatureValue>
3938 <ds:KeyInfo>
3939 <wsse:SecurityTokenReference>
3940 <wsse:Reference URI="#RecipientToken" />
3941 </wsse:SecurityTokenReference>
3942 </ds:KeyInfo>
3943 </ds:Signature>
3944 -->
3945 ...
3946 </xenc:EncryptedData>
3947 <xenc:ReferenceList>
3948 <xenc:DataReference URI="#enc_Body" />
3949 <xenc:DataReference URI="#enc_Header2" />
3950 ...
3951 </xenc:ReferenceList>
3952 </wsse:Security>
3953 </S:Header>
3954 <S:Body wsu:Id="Body">
3955 <xenc:EncryptedData Id="enc_Body">
3956 ...
3957 <ds:KeyInfo>
3958 <wsse:SecurityTokenReference>
3959 <wsse:Reference URI="#InitiatorEncryptedKey" />
3960 </wsse:SecurityTokenReference>
3961 </ds:KeyInfo>
3962 </xenc:EncryptedData>
3963 </S:Body>
3964 </S:Envelope>

```

D. Signed and Encrypted Elements in the Security Header

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

D.1 Elements signed by the message signature

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

D.2 Elements signed by all endorsing signatures

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

D.3 Elements signed by a specific endorsing signature

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

D.4 Elements that are encrypted

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

E. Acknowledgements

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

Original Authors of the initial contribution:

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

Original Acknowledgements of the initial contribution:

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

TC Members during the development of this specification:

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

4042 Geoff Bullen, Microsoft Corporation
4043 Lloyd Burch, Novell
4044 Scott Cantor, Internet2
4045 Greg Carpenter, Microsoft Corporation
4046 Steve Carter, Novell
4047 Symon Chang, Oracle Corporation Ching-Yun (C.Y.) Chao, IBM
4048 Martin Chapman, Oracle Corporation
4049 Kate Cherry, Lockheed Martin
4050 Henry (Hyenvui) Chung, IBM
4051 Luc Clement, Systinet Corp.
4052 Paul Cotton, Microsoft Corporation
4053 Glen Daniels, Sonic Software Corp.
4054 Peter Davis, Neustar, Inc.
4055 Duane DeCouteau, Veterans Health Administration
4056 Martijn de Boer, SAP AG
4057 Werner Dittmann, Siemens AG
4058 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
4059 Fred Dushin, IONA Technologies
4060 Petr Dvorak, Systinet Corp.
4061 Colleen Evans, Microsoft Corporation
4062 Ruchith Fernando, WSO2
4063 Mark Fussell, Microsoft Corporation
4064 Vijay Gajjala, Microsoft Corporation
4065 Marc Goodner, Microsoft Corporation
4066 Hans Granqvist, VeriSign
4067 Martin Gudgin, Microsoft Corporation
4068 Tony Gullotta, SOA Software Inc.
4069 Jiandong Guo, Sun Microsystems
4070 Phillip Hallam-Baker, VeriSign
4071 Patrick Harding, Ping Identity Corporation
4072 Heather Hinton, IBM
4073 Frederick Hirsch, Nokia Corporation
4074 Jeff Hodges, Neustar, Inc.
4075 Will Hopkins, Oracle Corporation
4076 Alex Hristov, Otecia Incorporated
4077 John Hughes, PA Consulting
4078 Diane Jordan, IBM
4079 Venugopal K, Sun Microsystems
4080 Chris Kaler, Microsoft Corporation
4081 Dana Kaufman, Forum Systems, Inc.
4082 Paul Knight, Nortel Networks Limited
4083 Ramanathan Krishnamurthy, IONA Technologies
4084 Christopher Kurt, Microsoft Corporation
4085 Kelvin Lawrence, IBM
4086 Hubert Le Van Gong, Sun Microsystems
4087 Jong Lee, Oracle Corporation
4088 Rich Levinson, Oracle Corporation
4089 Tommy Lindberg, Dajel Ltd.
4090 Mark Little, JBoss Inc.
4091 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.
4092 Eve Maler, Sun Microsystems
4093 Ashok Malhotra, Oracle Corporation
4094 Anand Mani, CrimsonLogic Pte Ltd
4095 Jonathan Marsh, Microsoft Corporation
4096 Robin Martherus, Oracle Corporation
4097 Miko Matsumura, Infravio, Inc.
4098 Gary McAfee, IBM

4099 Michael McIntosh, IBM
4100 John Merrells, Sxip Networks SRL
4101 Jeff Mischkinsky, Oracle Corporation
4102 Prateek Mishra, Oracle Corporation
4103 Bob Morgan, Internet2
4104 Vamsi Motukuru, Oracle Corporation
4105 Raajmohan Na, EDS
4106 Anthony Nadalin, IBM
4107 Andrew Nash, Reactivity, Inc.
4108 Eric Newcomer, IONA Technologies
4109 Duane Nickull, Adobe Systems
4110 Toshihiro Nishimura, Fujitsu Limited
4111 Rob Philpott, RSA Security
4112 Denis Pilipchuk, Oracle Corporation.
4113 Darren Platt, Ping Identity Corporation
4114 Martin Raepple, SAP AG
4115 Nick Ragouzis, Enosis Group LLC
4116 Prakash Reddy, CA
4117 Alain Regnier, Ricoh Company, Ltd.
4118 Irving Reid, Hewlett-Packard
4119 Bruce Rich, IBM
4120 Tom Rutt, Fujitsu Limited
4121 Maneesh Sahu, Actional Corporation
4122 Frank Siebenlist, Argonne National Laboratory
4123 Joe Smith, Apani Networks
4124 Davanum Srinivas, WSO2
4125 David Staggs, Veterans Health Administration
4126 Yakov Sverdlov, CA
4127 Gene Thurston, AmberPoint
4128 Victor Valle, IBM
4129 Asir Vedamuthu, Microsoft Corporation
4130 Greg Whitehead, Hewlett-Packard
4131 Ron Williams, IBM
4132 Corinna Witt, Oracle Corporation
4133 Kyle Young, Microsoft Corporation
4134
4135