



# WS-SecurityPolicy 1.2

## OASIS Standard

1 July 2007

### Specification URIs:

#### This Version:

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

#### Previous Version:

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

#### Latest Version:

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

### Artifact Type:

specification

### Technical Committee:

OASIS Web Services Secure Exchange TC

### Chair(s):

Kelvin Lawrence, IBM  
Chris Kaler, Microsoft

### Editor(s):

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

### Related work:

N/A

### Declared XML Namespace(s):

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

### Abstract:

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

### Status:

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

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

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

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

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

---

## Notices

Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

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

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

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

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

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

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

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

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

---

# Table of Contents

1	Introduction .....	7
1.1	Example .....	7
1.2	Namespaces .....	8
1.3	Schema Files .....	9
1.4	Terminology .....	9
1.4.1	Notational Conventions .....	9
1.5	Normative References .....	10
1.6	Non-Normative References .....	13
2	Security Policy Model .....	14
2.1	Security Assertion Model .....	14
2.2	Nested Policy Assertions .....	15
2.3	Security Binding Abstraction .....	15
3	Policy Considerations .....	17
3.1	Nested Policy .....	17
3.2	Policy Subjects .....	17
4	Protection Assertions .....	19
4.1	Integrity Assertions .....	19
4.1.1	SignedParts Assertion .....	19
4.1.2	SignedElements Assertion .....	20
4.2	Confidentiality Assertions .....	21
4.2.1	EncryptedParts Assertion .....	21
4.2.2	EncryptedElements Assertion .....	22
4.2.3	ContentEncryptedElements Assertion .....	22
4.3	Required Elements Assertion .....	23
4.3.1	RequiredElements Assertion .....	23
4.3.2	RequiredParts Assertion .....	24
5	Token Assertions .....	25
5.1	Token Inclusion .....	25
5.1.1	Token Inclusion Values .....	25
5.1.2	Token Inclusion and Token References .....	26
5.2	Token Issuer and Required Claims .....	26
5.2.1	Token Issuer .....	26
5.2.2	Token Issuer Name .....	26
5.2.3	Required Claims .....	26
5.2.4	Processing Rules and Token Matching .....	27
5.3	Token Properties .....	27
5.3.1	[Derived Keys] Property .....	27
5.3.2	[Explicit Derived Keys] Property .....	27
5.3.3	[Implied Derived Keys] Property .....	27
5.4	Token Assertion Types .....	27
5.4.1	UsernameToken Assertion .....	27

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

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

---

# 1 Introduction

WS-Policy defines a framework for allowing web services to express their constraints and requirements. Such constraints and requirements are expressed as policy assertions. This document defines a set of security policy assertions for use with the [WS-Policy] framework with respect to security features provided in WSS: SOAP Message Security [WSS10, WSS11], [WS-Trust] and [WS-SecureConversation]. The assertions defined within this specification have been designed to work independently of a specific version of WS-Policy. At the time of the publication of this specification the versions of WS-Policy known to correctly compose with this specification are WS-Policy 1.2 and 1.5. Within this specification the use of the namespace prefix `wsp` refers generically to the WS-Policy namespace, not a specific version. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

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

## 1.1 Example

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

Table 1: Example security policy.

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

```

44 (20) </sp:SignedParts>
45 (21) <sp:EncryptedParts>
46 (22)   <sp:Body/>
47 (23) </sp:EncryptedParts>
48 (24) </wsp:Policy>

```

49

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

## 62 1.2 Namespaces

63 The XML namespace URI that MUST be used by implementations of this specification is:

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

65

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

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

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



wsa	<a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a>	[ <a href="#">WS-Addressing</a> ]
sp	<a href="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702</a>	This specification

## 69 1.3 Schema Files

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

72 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2.xsd>

## 73 1.4 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

### 90 1.4.1 Notational Conventions

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

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

- 95 • The syntax appears as an XML instance, but values in italics indicate data types instead of literal  
96 values.
- 97 • Characters are appended to elements and attributes to indicate cardinality:
  - 98 ○ "?" (0 or 1)
  - 99 ○ "\*" (0 or more)
  - 100 ○ "+" (1 or more)
- 101 • The character "|" is used to indicate a choice between alternatives.
- 102 • The characters "(" and ")" are used to indicate that contained items are to be treated as a group  
103 with respect to cardinality or choice.
- 104 • The characters "[" and "]" are used to call out references and property names.
- 105 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be  
106 added at the indicated extension points but MUST NOT contradict the semantics of the parent

107 and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver  
108 SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated  
109 below.

- 110 • XML namespace prefixes (see Table 2) are used to indicate the namespace of the element being  
111 defined.

112

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

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

121 Extensibility points in the exemplar may not be described in the corresponding text.

122 In this document reference is made to the `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires`  
123 elements in a utility schema ([http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-  
124 1.0.xsd](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd)). The `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires` elements were added to the  
125 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp  
126 element could reference it (as is done here).

127

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

## 133 1.5 Normative References

- 134 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement  
135 Levels", RFC 2119, Harvard University, March 1997.  
136 <http://www.ietf.org/rfc/rfc2119.txt>
- 137
- 138 [SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.  
139 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 140
- 141 [SOAP12] W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24  
142 June 2003.  
143 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 144
- 145 [SOAPNorm] W3C Working Group Note, "SOAP Version 1.2 Message  
146 Normalization", 8 October 2003.  
147 <http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>
- 148
- 149 [URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers  
150 (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe  
151 Systems, January 2005.  
152 <http://www.ietf.org/rfc/rfc3986.txt>

153		
154	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
155		
156		<a href="http://www.ietf.org/rfc/rfc2068.txt">http://www.ietf.org/rfc/rfc2068.txt</a>
157		
158	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
159		<a href="http://www.ietf.org/rfc/rfc2246.txt">http://www.ietf.org/rfc/rfc2246.txt</a>
160		
161	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
162		<a href="http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211">http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211</a>
163		
164	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
165		
166		<a href="http://www.w3.org/TR/2006/REC-ws-addr-core-20060509">http://www.w3.org/TR/2006/REC-ws-addr-core-20060509</a>
167		
168	[WS-Policy]	W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
169		
170		<a href="http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/">http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/</a>
171		W3C Candidate Recommendation "Web Services Policy 1.5 – Framework", 28 February 2007
172		
173		<a href="http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/">http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/</a>
174		
175	[WS-PolicyAttachment]	W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
176		
177		<a href="http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/">http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/</a>
178		
179		W3C Candidate Recommendation "Web Services Policy 1.5 – Attachment", 28 February 2007
180		
181		<a href="http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/">http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/</a>
182		
183	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.3", September 2006
184		<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512</a>
185		
186	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.3", September 2006
187		
188		<a href="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512">http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512</a>
189		
190	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
191		
192		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf</a>
193		
194		
195	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
196		
197		<a href="http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf">http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf</a>
198		

199		
200	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile",
201		March 2004
202		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-</a>
203		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf">token-profile-1.0.pdf</a>
204		
205	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile
206		1.1", February 2006
207		<a href="http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-</a>
208		<a href="http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf">spec-os-UsernameTokenProfile.pdf</a>
209		
210	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token
211		Profile", March 2004
212		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-</a>
213		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf">profile-1.0.pdf</a>
214		
215	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token
216		Profile", February 2006
217		<a href="http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-</a>
218		<a href="http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf">spec-os-x509TokenProfile.pdf</a>
219		
220	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1",
221		February 2006
222		<a href="http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-</a>
223		<a href="http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf">spec-os-KerberosTokenProfile.pdf</a>
224		
225	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile",
226		December 2004
227		<a href="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf</a>
228		
229	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1",
230		February 2006
231		<a href="http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-</a>
232		<a href="http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf">spec-os-SAMLTokenProfile.pdf</a>
233		
234	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language
235		(REL) Token Profile", December 2004
236		<a href="http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf</a>
237		
238	[WSS:RELTTokenProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language
239		(REL) Token Profile 1.1", February 2006
240		<a href="http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf">http://www.oasis-open.org/committees/download.php/16687/oasis-</a>
241		<a href="http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf">wss-rel-token-profile-1.1.pdf</a>
242		
243	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with
244		Attachments (SwA) Profile 1.1", February 2006

245 <http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf>  
246  
247  
248 [XML-Encrypt] W3C Recommendation, "XML Encryption Syntax and Processing", 10  
249 December 2002.  
250 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>  
251  
252 [XML-Signature] W3C Recommendation, "XML-Signature Syntax and Processing", 12  
253 February 2002.  
254 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>  
255  
256 [XPath] W3C Recommendation "XML Path Language (XPath) Version 1.0", 16  
257 November 1999.  
258 <http://www.w3.org/TR/1999/REC-xpath-19991116>  
259  
260 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second  
261 Edition", 28 October 2004.  
262 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>  
263  
264 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second  
265 Edition", 28 October 2004.  
266 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>  
267

## 268 **1.6 Non-Normative References**

269 None.  
270

---

## 271 **2 Security Policy Model**

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

276  
277 The primary goal of this specification is to define an initial set of patterns or sets of assertions that  
278 represent common ways to describe how messages are secured on a communication path. The intent is  
279 to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging  
280 transport security, but to be specific enough to ensure interoperability based on assertion matching.

281  
282 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for  
283 selecting policy alternatives and the attachment mechanism for associating policy assertions with web  
284 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters  
285 or attributes. This enables first-level, QName based assertion matching without security domain-specific  
286 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed  
287 set of policy alternatives that are shared by the two parties attempting to establish a secure  
288 communication path.

289  
290 In general, assertions defined in this specification allow additional attributes, based on schemas, to be  
291 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not  
292 match based on these attributes. Attributes specified on the assertion element that are not defined in this  
293 specification or in WS-Policy are to be treated as informational properties.

### 294 **2.1 Security Assertion Model**

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

302  
303 To indicate the scope of protection, assertions identify message parts that are to be protected in a  
304 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

305  
306 The general aspects of security includes the relationships between or characteristics of the environment  
307 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality  
308 protection and which are supporting, the applicable algorithms to use, etc.

309  
310 The security binding assertion is a logical grouping which defines how the general aspects are used to  
311 protect the indicated parts. For example, that an asymmetric token is used with a digital signature to  
312 provide integrity protection, and that parts are encrypted with a symmetric key which is then encrypted

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

315

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

## 320 **2.2 Nested Policy Assertions**

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

## 327 **2.3 Security Binding Abstraction**

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

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

333

334 A binding defines the following security characteristics:

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

343

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

350

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

- 354 • TransportBinding (Section 7.3)
- 355 • SymmetricBinding (Section 7.4)

- AsymmetricBinding (Section 7.5)



---

## 357 3 Policy Considerations

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

### 360 3.1 Nested Policy

361 This specification makes extensive use of nested policy assertions as described in the [Policy Assertion](#)  
362 [Nesting](#) section of WS-Policy.  
363

### 364 3.2 Policy Subjects

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

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

#### 369 [Message Policy Subject]

370 This property identifies a Message Policy Subject [[WS-PolicyAttachment](#)]. WS-PolicyAttachment defines  
371 seven WSDL [WSDL 1.1] policy attachment points with Message Policy Subject:  
372

373 wsdl:message

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

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

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

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

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

#### 382 [Operation Policy Subject]

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

384 wsdl:portType/wsdl:operation

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

387 wsdl:binding/wsdl:operation

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

390

391

#### 392 [Endpoint Policy Subject]

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

395 wsdl:portType

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

398 wsdl:binding

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

401 wsdl:port

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

---

## 404 4 Protection Assertions

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

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

### 412 4.1 Integrity Assertions

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

#### 416 4.1.1 SignedParts Assertion

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

422

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

#### 427 Syntax

```
428 <sp:SignedParts xmlns:sp="..." ... >  
429   <sp:Body />?  
430   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
431   <sp:Attachments />?  
432   ...  
433 </sp:SignedParts>
```

434

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

436 /sp:SignedParts

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

440 /sp:SignedParts/sp:Body

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

443 /sp:SignedParts/sp:Header

444 Presence of this optional element indicates a specific SOAP header, its attributes and content (or  
445 set of such headers) needs to be protected. There may be multiple sp:Header elements within a

446 single sp:SignedParts element. If multiple SOAP headers with the same local name but different  
447 namespace names are to be integrity protected multiple sp:Header elements are needed, either  
448 as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts assertions.  
449 This element only applies to SOAP header elements targeted to the same actor/role as the  
450 Security header impacted by the policy. If it is necessary to specify a requirement to sign specific  
451 SOAP Header elements targeted to a different actor/role, that may be accomplished using the  
452 sp:SignedElements assertion.

453 /sp:SignedParts/sp:Header/@Name

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

457 /sp:SignedParts/sp:Header/@Namespace

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

459 /sp:SignedParts/sp:Attachments

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

#### 465 4.1.2 SignedElements Assertion

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

471  
472 There MAY be multiple SignedElements assertions present. Multiple SignedElements assertions present  
473 within a policy alternative are equivalent to a single SignedElements assertion containing the union of all  
474 specified XPath expressions.

#### 475 Syntax

```
476 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
477 <sp:XPath>xs:string</sp:XPath>+  
478 ...  
479 </sp:SignedElements>
```

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

481 /sp:SignedElements

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

483 /sp:SignedElements/@XPathVersion

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

486 /sp:SignedElements/sp:XPath

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

## 491 4.2 Confidentiality Assertions

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

### 495 4.2.1 EncryptedParts Assertion

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

501  
502 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present  
503 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all  
504 specified message parts. Note that this assertion does not require that a given part appear in a message,  
505 just that if such a part appears, it requires confidentiality protection.

#### 506 Syntax

```
507 <sp:EncryptedParts xmlns:sp="..." ... >  
508   <sp:Body/>?  
509   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
510   <sp:Attachments />?  
511   ...  
512 </sp:EncryptedParts>
```

513  
514 The following describes the attributes and elements listed in the schema outlined above:

#### 515 /sp:EncryptedParts

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

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

#### 519 /sp:EncryptedParts/sp:Body

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

#### 524 /sp:EncryptedParts/sp:Header

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

#### 533 /sp:EncryptedParts/sp:Header/@Name

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

537 /sp:EncryptedParts/sp:Header/@Namespace

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

540 /sp:EncryptedParts/sp:Attachments

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

## 546 4.2.2 EncryptedElements Assertion

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

552

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

### 556 Syntax

```
557 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
558 <sp:XPath>xs:string</sp:XPath>+  
559 ...  
560 </sp:EncryptedElements>
```

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

562 /sp:EncryptedElements

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

565 /sp:EncryptedElements/@XPathVersion

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

568 /sp:EncryptedElements/sp:XPath

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

## 573 4.2.3 ContentEncryptedElements Assertion

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

579

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

### 583 **Syntax**

```
584 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
585   <sp:XPath>xs:string</sp:XPath>+  
586   ...  
587 </sp:ContentEncryptedElements>
```

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

589 /sp:ContentEncryptedElements

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

592 /sp:ContentEncryptedElements/@XPathVersion

593 This optional attribute contains a URI which indicates the version of XPath to use.

594 /sp:ContentEncryptedElements/sp:XPath

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

## 599 **4.3 Required Elements Assertion**

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

602

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

### 606 **4.3.1 RequiredElements Assertion**

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

609

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

### 613 **Syntax**

```
614 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
615   <sp:XPath>xs:string</sp:XPath> +  
616   ...  
617 </sp:RequiredElements>
```

618

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

620 /sp:RequiredElements

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

622 /sp:RequiredElements/@XPathVersion

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

625 /sp:RequiredElements/sp:XPath

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

### 630 4.3.2 RequiredParts Assertion

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

634

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

#### 638 Syntax

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

642

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

644 /sp:RequiredParts/sp:Header

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

646 /sp:RequiredParts/sp:Header/@Name

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

649 /sp:RequiredParts/sp:Header/@Namespace

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



---

## 652 5 Token Assertions

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

### 657 5.1 Token Inclusion

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

#### 663 5.1.1 Token Inclusion Values

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

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

665  
666 Note: In examples, the namespace URI is replaced with "...". For example,  
667 `.../IncludeToken/Never` is actually `http://docs.oasis-open.org/ws-sx/ws-`  
668 `securitypolicy/200702/IncludeToken/Never`. Other token inclusion URI values MAY be defined but are out-  
669 of-scope of this specification.

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

## 672 **5.1.2 Token Inclusion and Token References**

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

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

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

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

## 689 **5.2 Token Issuer and Required Claims**

### 690 **5.2.1 Token Issuer**

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

### 695 **5.2.2 Token Issuer Name**

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

700  
701 It is out of scope of this specification how the relationship between the issuer's logical name and the  
702 physical manifestation of the issuer in the security token is defined.

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

### 705 **5.2.3 Required Claims**

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

709  
710 This element indicates the required claims that the security token must contain in order to satisfy the  
711 requirements of the token assertion.

712  
713 Individual token assertions may further limit what claims may be specified for that specific token assertion.

## 714 **5.2.4 Processing Rules and Token Matching**

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

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

- 722 1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and  
723 T2 is issued by issuer B and contains claims C3 and C4.
- 724 2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is  
725 also issued by issuer A and contains claim C2 and T3 is issued by issuer B and  
726 contains claims C3 and C4.
- 727 3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2,  
728 T2 is issued by issuer B and contains claim C3 and T3 is also issued by issuer B and  
729 contains claim C4.
- 730 4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is  
731 also issued by issuer A and contains claim C2, T3 is issued by issuer B and contains  
732 claim C3 and T4 is also issued by issuer B and contains claim C4.

## 734 **5.3 Token Properties**

### 735 **5.3.1 [Derived Keys] Property**

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

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

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

### 744 **5.3.2 [Explicit Derived Keys] Property**

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

### 748 **5.3.3 [Implied Derived Keys] Property**

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

## 752 **5.4 Token Assertion Types**

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

### 754 **5.4.1 UsernameToken Assertion**

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

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

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

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

764

## 765 Syntax

```
766 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
767   (  
768     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
769     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
770   ) ?  
771   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
772   <wsp:Policy xmlns:wsp="...">  
773     (  
774       <sp:NoPassword ... /> |  
775       <sp:HashPassword ... />  
776     ) ?  
777     (  
778       <sp:RequireDerivedKeys /> |  
779       <sp:RequireImpliedDerivedKeys ... /> |  
780       <sp:RequireExplicitDerivedKeys ... />  
781     ) ?  
782     (  
783       <sp:WssUsernameToken10 ... /> |  
784       <sp:WssUsernameToken11 ... />  
785     ) ?  
786     ...  
787   </wsp:Policy>  
788   ...  
789 </sp:UsernameToken>
```

790

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

792 /sp:UsernameToken

793 This identifies a UsernameToken assertion.

794 /sp:UsernameToken/@sp:IncludeToken

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

796 /sp:UsernameToken/sp:Issuer

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

799 /sp:UsernameToken/sp:IssuerName

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

802 /sp:UsernameToken/wst:Claims

803 This optional element identifies the required claims that a security token must contain in order to  
804 satisfy the token assertion requirements.

805 /sp:UsernameToken/wsp:Policy

806 This required element identifies additional requirements for use of the sp:UsernameToken  
807 assertion.

808 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

811 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

816 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

819 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

822 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

826 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

829 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

## 832 5.4.2 IssuedToken Assertion

833 This element represents a requirement for an issued token, which is one issued by some token issuer  
834 using the mechanisms defined in WS-Trust. This assertion is used in 3<sup>rd</sup> party scenarios. For example,  
835 the initiator may need to request a SAML token from a given token issuer in order to secure messages  
836 sent to the recipient.

### 837 Syntax

```
838 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
839 (
840 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
841 <sp:IssuerName>xs:anyURI</sp:IssuerName>
842 ) ?
```

```

843 <wst:Claims Dialect="..."> ... </wst:Claims> ?
844 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
845   ...
846 </sp:RequestSecurityTokenTemplate>
847 <wsp:Policy xmlns:wsp="...">
848   (
849     <sp:RequireDerivedKeys ... /> |
850     <sp:RequireImpliedDerivedKeys ... /> |
851     <sp:RequireExplicitDerivedKeys ... />
852   ) ?
853   <sp:RequireExternalReference ... /> ?
854   <sp:RequireInternalReference ... /> ?
855   ...
856 </wsp:Policy>
857   ...
858 </sp:IssuedToken>

```

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

860 /sp:IssuedToken

861       This identifies an IssuedToken assertion.

862 /sp:IssuedToken/@sp:IncludeToken

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

864 /sp:IssuedToken/sp:Issuer

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

867 /sp:IssuedToken/sp:IssuerName

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

869 /sp:IssuedToken/wst:Claims

870       This optional element identifies the required claims that a security token must contain in order to  
871 satisfy the token assertion requirements.

872 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

877 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

878       This optional attribute contains a WS-Trust specification namespace URI identifying the version of  
879 WS-Trust referenced by the contents of this element.

880 /sp:IssuedToken/wsp:Policy

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

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

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

885 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

888 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

892 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

893 This optional element is a policy assertion that indicates whether an internal reference is required  
894 when referencing this token.

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

896 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

897 This optional element is a policy assertion that indicates whether an external reference is required  
898 when referencing this token.

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

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

### 905 5.4.3 X509Token Assertion

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

#### 907 Syntax

```
908 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
909   (  
910     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
911     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
912   ) ?  
913   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
914   <wsp:Policy xmlns:wsp="...">  
915     (  
916       <sp:RequireDerivedKeys ... /> |  
917       <sp:RequireExplicitDerivedKeys ... /> |  
918       <sp:RequireImpliedDerivedKeys ... />  
919     ) ?  
920     <sp:RequireKeyIdentifierReference ... /> ?  
921     <sp:RequireIssuerSerialReference ... /> ?  
922     <sp:RequireEmbeddedTokenReference ... /> ?  
923     <sp:RequireThumbprintReference ... /> ?  
924     (  
925       <sp:WssX509V3Token10 ... /> |  
926       <sp:WssX509Pkcs7Token10 ... /> |  
927       <sp:WssX509PkiPathV1Token10 ... /> |  
928       <sp:WssX509V1Token11 ... /> |  
929       <sp:WssX509V3Token11 ... /> |  
930       <sp:WssX509Pkcs7Token11 ... /> |  
931       <sp:WssX509PkiPathV1Token11 ... />  
932     ) ?  
933     ...  
934   </wsp:Policy>  
935   ...  
936 </sp:X509Token>
```

937

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

939 /sp:X509Token

940 This identifies an X509Token assertion.

941 /sp:X509Token/@sp:IncludeToken  
 942 This optional attribute identifies the token inclusion value for this token assertion.

943 /sp:X509Token/sp:Issuer  
 944 This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of  
 945 the `sp:X509Token`.

946 /sp:X509Token/sp:IssuerName  
 947 This optional element, of type `xs:anyURI`, contains the logical name of the `sp:X509Token` issuer.

948 /sp:X509Token/wst:Claims  
 949 This optional element identifies the required claims that a security token must contain in order to  
 950 satisfy the token assertion requirements.

951 /sp:X509Token/wsp:Policy  
 952 This required element identifies additional requirements for use of the `sp:X509Token` assertion.

953 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys  
 954 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 955 and [Implied Derived Keys] properties for this token to 'true'.

956 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys  
 957 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]  
 958 properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.

959 /sp:X509Token/wsp:Policy/sp:RequireImpliedDerivedKeys  
 960 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 961 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 962 'false'.

963 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference  
 964 This optional element is a policy assertion that indicates that a key identifier reference is required  
 965 when referencing this token.

966 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference  
 967 This optional element is a policy assertion that indicates that an issuer serial reference is required  
 968 when referencing this token.

969 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference  
 970 This optional element is a policy assertion that indicates that an embedded token reference is  
 971 required when referencing this token.

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

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

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

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



- 984 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11
- 985 This optional element is a policy assertion that indicates that an X509 Version 1 token should be  
986 used as defined in [\[WSS:X509TokenProfile1.1\]](#).
- 987 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11
- 988 This optional element is a policy assertion that indicates that an X509 Version 3 token should be  
989 used as defined in [\[WSS:X509TokenProfile1.1\]](#).
- 990 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11
- 991 This optional element is a policy assertion that indicates that an X509 PKCS7 token should be  
992 used as defined in [\[WSS:X509TokenProfile1.1\]](#).
- 993 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11
- 994 This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token  
995 should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

#### 996 5.4.4 KerberosToken Assertion

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

#### 998 Syntax

```

999 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1000 (
1001   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1002   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1003 ) ?
1004 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1005 <wsp:Policy xmlns:wsp="...">
1006   (
1007     <sp:RequireDerivedKeys ... /> |
1008     <sp:RequireImpliedDerivedKeys ... /> |
1009     <sp:RequireExplicitDerivedKeys ... />
1010   ) ?
1011   <sp:RequireKeyIdentifierReference ... /> ?
1012   (
1013     <sp:WssKerberosV5ApReqToken11 ... /> |
1014     <sp:WssGssKerberosV5ApReqToken11 ... />
1015   ) ?
1016   ...
1017 </wsp:Policy>
1018   ...
1019 </sp:KerberosToken>

```

1021

1022 The following describes the attributes and elements listed in the schema outlined above:

1023 /sp:KerberosToken

1024 This identifies a KerberosV5ApReqToken assertion.

1025 /sp:KerberosToken/@sp:IncludeToken

1026 This optional attribute identifies the token inclusion value for this token assertion.

1027 /sp:KerberosToken/sp:Issuer

1028 This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of  
1029 the sp:KerberosToken.

1030 /sp:KerberosToken/sp:IssuerName

1031 This optional element, of type `xs:anyURI`, contains the logical name of the `sp:KerberosToken`  
 1032 issuer.

1033 `/sp:KerberosToken/wst:Claims`

1034 This optional element identifies the required claims that a security token must contain in order to  
 1035 satisfy the token assertion requirements.

1036 `/sp:KerberosToken/wsp:Policy`

1037 This required element identifies additional requirements for use of the `sp:KerberosToken`  
 1038 assertion.

1039 `/sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys`

1040 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1041 and [Implied Derived Keys] properties for this token to 'true'.

1042 `/sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys`

1043 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]  
 1044 properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.

1045 `/sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys`

1046 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1047 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1048 'false'.

1049 `/sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference`

1050 This optional element is a policy assertion that indicates that a key identifier reference is required  
 1051 when referencing this token.

1052 `/sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11`

1053 This optional element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ token  
 1054 should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

1055 `/sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11`

1056 This optional element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-REQ  
 1057 token should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

## 1058 5.4.5 SpnegoContextToken Assertion

1059 This element represents a requirement for a `SecurityContextToken` obtained by executing an n-leg  
 1060 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in `WS-Trust`.

### 1061 Syntax

```

1062 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1063   (
1064     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1065     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1066   ) ?
1067   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1068   <wsp:Policy xmlns:wsp="...">
1069     (
1070       <sp:RequireDerivedKeys ... /> |
1071       <sp:RequireImpliedDerivedKeys ... /> |
1072       <sp:RequireExplicitDerivedKeys ... />
1073     ) ?
1074     <sp:MustNotSendCancel ... /> ?
1075     <sp:MustNotSendAmend ... /> ?
  
```

1076  
1077  
1078  
1079  
1080

```
<sp:MustNotSendRenew ... /> ?  
...  
</wsp:Policy>  
...  
</sp:SpnegoContextToken>
```

1081

1082 The following describes the attributes and elements listed in the schema outlined above:

1083 /sp:SpnegoContextToken

1084 This identifies a SpnegoContextToken assertion.

1085 /sp:SpnegoContextToken/@sp:IncludeToken

1086 This optional attribute identifies the token inclusion value for this token assertion.

1087 /sp:SpnegoContextToken/sp:Issuer

1088 This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for  
1089 the Spnego Context Token.

1090 /sp:SpnegoContextToken/sp:IssuerName

1091 This optional element, of type xs:anyURI, contains the logical name of the  
1092 sp:SpnegoContextToken issuer.

1093 /sp:SpnegoContextToken/wst:Claims

1094 This optional element identifies the required claims that a security token must contain in order to  
1095 satisfy the token assertion requirements.

1096 /sp:SpnegoContextToken/wsp:Policy

1097 This required element identifies additional requirements for use of the sp:SpnegoContextToken  
1098 assertion.

1099 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1100 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1101 and [Implied Derived Keys] properties for this token to 'true'.

1102 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1105 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1109 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1110 This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token  
1111 does not support SCT/Cancel RST messages. If this assertion is missing it means that  
1112 SCT/Cancel RST messages are supported by the STS.

1113 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1114 This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token  
1115 does not support SCT/Amend RST messages. If this assertion is missing it means that  
1116 SCT/Amend RST messages are supported by the STS.

1117 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1118 This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token  
1119 does not support SCT/Renew RST messages. If this assertion is missing it means that  
1120 SCT/Renew RST messages are supported by the STS.

## 1121 5.4.6 SecurityContextToken Assertion

1122 This element represents a requirement for a SecurityContextToken token.

### 1123 Syntax

```
1124 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1125 (
1126   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1127   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1128 ) ?
1129 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1130 <wsp:Policy xmlns:wsp="...">
1131   (
1132     <sp:RequireDerivedKeys ... /> |
1133     <sp:RequireImpliedDerivedKeys ... /> |
1134     <sp:RequireExplicitDerivedKeys ... />
1135   ) ?
1136   <sp:RequireExternalUriReference ... /> ?
1137   <sp:SC13SecurityContextToken... /> ?
1138   ...
1139 </wsp:Policy>
1140 ...
1141 </sp:SecurityContextToken>
```

1142  
1143 The following describes the attributes and elements listed in the schema outlined above:

1144 /sp:SecurityContextToken

1145       This identifies a SecurityContextToken assertion.

1146 /sp:SecurityContextToken/@sp:IncludeToken

1147       This optional attribute identifies the token inclusion value for this token assertion.

1148 /sp:SecurityContextToken/sp:Issuer

1149       This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of  
1150 the sp:SecurityContextToken.

1151 /sp:SecurityContextToken/sp:IssuerName

1152       This optional element, of type xs:anyURI, contains the logical name of the  
1153 sp:SecurityContextToken issuer.

1154 /sp:SecurityContextToken/wst:Claims

1155       This optional element identifies the required claims that a security token must contain in order to  
1156 satisfy the token assertion requirements.

1157 /sp:SecurityContextToken/wsp:Policy

1158       This required element identifies additional requirements for use of the sp:SecurityContextToken  
1159 assertion.

1160 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1161       This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1162 and [Implied Derived Keys] properties for this token to 'true'.

1163 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1166 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1170 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1171 This optional element is a policy assertion that indicates that an external URI reference is  
1172 required when referencing this token.

1173 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1174 This optional element is a policy assertion that indicates that a Security Context Token should be  
1175 used as defined in [WS-SecureConversation].

1176

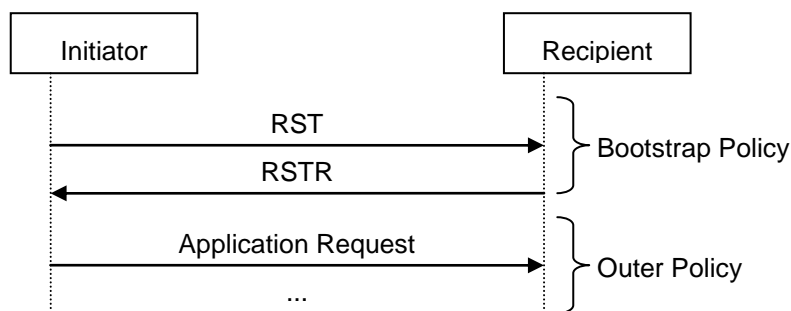
1177 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that  
1178 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If  
1179 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the  
1180 sp:SecureConversationToken or the sp:IssuedToken assertion should be used instead.

### 1181 5.4.7 SecureConversationToken Assertion

1182 This element represents a requirement for a Security Context Token retrieved from the indicated issuer  
1183 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the  
1184 service endpoint address.

1185

1186 Note: This assertion describes the token accepted by the target service. Because this token is issued by  
1187 the target service and may not have a separate port (with separate policy), this assertion SHOULD  
1188 contain a bootstrap policy indicating the security binding and policy that is used when requesting this  
1189 token from the target service. That is, the bootstrap policy is used to obtain the token and then the  
1190 current (outer) policy is used when making requests with the token. This is illustrated in the diagram  
1191 below.



1192

### 1193 Syntax

```
1194 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1195 (
1196   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1197   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1198 ) ?
1199 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1200 <wsp:Policy xmlns:wsp="...">
1201 (
1202   <sp:RequireDerivedKeys ... /> |
1203   <sp:RequireImpliedDerivedKeys ... /> |
1204   <sp:RequireExplicitDerivedKeys ... />
1205 ) ?
1206 <sp:RequireExternalUriReference ... /> ?
1207 <sp:SC13SecurityContextToken ... /> ?
```

```
1208 <sp:MustNotSendCancel ... /> ?
1209 <sp:MustNotSendAmend ... /> ?
1210 <sp:MustNotSendRenew ... /> ?
1211 <sp:BootstrapPolicy ... >
1212 <wsp:Policy> ... </wsp:Policy>
1213 </sp:BootstrapPolicy> ?
1214 </wsp:Policy>
1215 ...
1216 </sp:SecureConversationToken>
```

1217

1218 The following describes the attributes and elements listed in the schema outlined above:

1219 /sp:SecureConversationToken

1220 This identifies a SecureConversationToken assertion.

1221 /sp:SecureConversationToken/@sp:IncludeToken

1222 This optional attribute identifies the token inclusion value for this token assertion.

1223 /sp:SecureConversationToken/sp:Issuer

1224 This optional element, of type `wsa:EndpointReferenceType`, contains a reference to the issuer for  
1225 the Security Context Token.

1226 /sp:SecureConversationToken/sp:IssuerName

1227 This optional element, of type `xs:anyURI`, contains the logical name of the  
1228 `sp:SecureConversationToken` issuer.

1229 /sp:SpnegoContextToken/wst:Claims

1230 This optional element identifies the required claims that a security token must contain in order to  
1231 satisfy the token assertion requirements.

1232 /sp:SecureConversationToken/wsp:Policy

1233 This required element identifies additional requirements for use of the  
1234 `sp:SecureConversationToken` assertion.

1235 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1236 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1237 and [Implied Derived Keys] properties for this token to 'true'.

1238 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1239 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]  
1240 properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.

1241 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1242 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1243 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1244 'false'.

1245 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1246 This optional element is a policy assertion that indicates that an external URI reference is  
1247 required when referencing this token.

1248 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken

1249 This optional element is a policy assertion that indicates that a Security Context Token should be  
1250 used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1251 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

1252 This optional element is a policy assertion that indicates that the STS issuing the secure  
1253 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it  
1254 means that SCT/Cancel RST messages are supported by the STS.

1255 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

1256 This optional element is a policy assertion that indicates that the STS issuing the secure  
1257 conversation token does not support SCT/Amend RST messages. If this assertion is missing it  
1258 means that SCT/Amend RST messages are supported by the STS.

1259 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1260 This optional element is a policy assertion that indicates that the STS issuing the secure  
1261 conversation token does not support SCT/Renew RST messages. If this assertion is missing it  
1262 means that SCT/Renew RST messages are supported by the STS.

1263 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1264 This optional element is a policy assertion that contains the policy indicating the requirements for  
1265 obtaining the Security Context Token.

1266 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1267 This element contains the security binding requirements for obtaining the Security Context Token.  
1268 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with  
1269 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that  
1270 are to be protected.

#### 1271 Example

```
1272 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
1273 <sp:SymmetricBinding>  
1274 <wsp:Policy>  
1275 <sp:ProtectionToken>  
1276 <wsp:Policy>  
1277 <sp:SecureConversationToken>  
1278 <sp:Issuer>  
1279 <wsa:Address>http://example.org/sts</wsa:Address>  
1280 </sp:Issuer>  
1281 </wsp:Policy>
```

```

1282         <sp:SC10SecurityContextToken />
1283         <sp:BootstrapPolicy>
1284             <wsp:Policy>
1285                 <sp:AsymmetricBinding>
1286                     <wsp:Policy>
1287                         <sp:InitiatorToken>
1288                             ...
1289                         </sp:InitiatorToken>
1290                         <sp:RecipientToken>
1291                             ...
1292                         </sp:RecipientToken>
1293                     </wsp:Policy>
1294                 </sp:AsymmetricBinding>
1295                 <sp:SignedParts>
1296                     ...
1297                 </sp:SignedParts>
1298                 ...
1299             </wsp:Policy>
1300         </sp:BootstrapPolicy>
1301     </wsp:Policy>
1302 </sp:SecureConversationToken>
1303 </wsp:Policy>
1304 </sp:ProtectionToken>
1305     ...
1306 </wsp:Policy>
1307 </sp:SymmetricBinding>
1308 <sp:SignedParts>
1309     ...
1310 </sp:SignedParts>
1311     ...
1312 </wsp:Policy>

```

#### 1313 5.4.8 SamlToken Assertion

1314 This element represents a requirement for a SAML token.

#### 1315 Syntax

```

1316 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1317 (
1318     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1319     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1320 ) ?
1321 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1322 <wsp:Policy xmlns:wsp="...">
1323 (
1324     <sp:RequireDerivedKeys ... /> |
1325     <sp:RequireImpliedDerivedKeys ... /> |
1326     <sp:RequireExplicitDerivedKeys ... />
1327 ) ?
1328 <sp:RequireKeyIdentifierReference ... /> ?
1329 (
1330     <sp:WssSamlV11Token10 ... /> |
1331     <sp:WssSamlV11Token11 ... /> |
1332     <sp:WssSamlV20Token11 ... />
1333 ) ?
1334     ...
1335 </wsp:Policy>
1336     ...
1337 </sp:SamlToken>

```

1338

1339 The following describes the attributes and elements listed in the schema outlined above:



1340 /sp:SamIToken  
1341 This identifies a SamIToken assertion.

1342 /sp:SamIToken/@sp:IncludeToken  
1343 This optional attribute identifies the token inclusion value for this token assertion.

1344 /sp:SamIToken/sp:Issuer  
1345 This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of  
1346 the `sp:SamIToken`.

1347 /sp:SamIToken/sp:IssuerName  
1348 This optional element, of type `xs:anyURI`, contains the logical name of the `sp:SamIToken` issuer.

1349 /sp:SamIToken/wst:Claims  
1350 This optional element identifies the required claims that a security token must contain in order to  
1351 satisfy the token assertion requirements.

1352 /sp:SamIToken/wsp:Policy  
1353 This required element identifies additional requirements for use of the `sp:SamIToken` assertion.

1354 /sp:SamIToken/wsp:Policy/sp:RequireDerivedKeys  
1355 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1356 and [Implied Derived Keys] properties for this token to 'true'.

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

1360 /sp:SamIToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
1361 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1362 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1363 'false'.

1364 /sp:SamIToken/wsp:Policy/sp:RequireKeyIdentifierReference  
1365 This optional element is a policy assertion that indicates that a key identifier reference is required  
1366 when referencing this token.

1367 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10  
1368 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be  
1369 used as defined in [\[WSS:SAMLTokenProfile1.0\]](#).

1370 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11  
1371 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be  
1372 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1373 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11  
1374 This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be  
1375 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1376  
1377 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties  
1378 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
1379 of the mechanism for obtaining the SAML Token is desired in policy, the `sp:IssuedToken` assertion should  
1380 be used instead.

## 1381 5.4.9 RelToken Assertion

1382 This element represents a requirement for a REL token.

### 1383 Syntax

```
1384 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1385 (
1386   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1387   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1388 ) ?
1389 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1390 <wsp:Policy xmlns:wsp="...">
1391 (
1392   <sp:RequireDerivedKeys ... /> |
1393   <sp:RequireImpliedDerivedKeys ... /> |
1394   <sp:RequireExplicitDerivedKeys ... />
1395 ) ?
1396 <sp:RequireKeyIdentifierReference ... /> ?
1397 (
1398   <sp:WssRelV10Token10 ... /> |
1399   <sp:WssRelV20Token10 ... /> |
1400   <sp:WssRelV10Token11 ... /> |
1401   <sp:WssRelV20Token11 ... />
1402 ) ?
1403 ...
1404 </wsp:Policy>
1405 ...
1406 </sp:RelToken>
```

1407

1408 The following describes the attributes and elements listed in the schema outlined above:

1409 /sp:RelToken

1410 This identifies a RelToken assertion.

1411 /sp:RelToken/@sp:IncludeToken

1412 This optional attribute identifies the token inclusion value for this token assertion.

1413 /sp:RelToken/sp:Issuer

1414 This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of  
1415 the sp:RelToken.

1416 /sp:RelToken/sp:IssuerName

1417 This optional element, of type xs:anyURI, contains the logical name of the sp:RelToken issuer.

1418 /sp:RelToken/wst:Claims

1419 This optional element identifies the required claims that a security token must contain in order to  
1420 satisfy the token assertion requirements.

1421 /sp:RelToken/wsp:Policy

1422 This required element identifies additional requirements for use of the sp:RelToken assertion.

1423 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1424 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1425 and [Implied Derived Keys] property for this token to 'true'.

1426 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1427 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]  
1428 properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.

1429 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1430 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1431 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1432 'false'.

1433 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference  
 1434 This optional element is a policy assertion that indicates that a key identifier reference is required  
 1435 when referencing this token.

1436 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10  
 1437 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be  
 1438 used as defined in [WSS:RELTokenProfile1.0].

1439 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10  
 1440 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be  
 1441 used as defined in [WSS:RELTokenProfile1.0].

1442 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11  
 1443 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be  
 1444 used as defined in [WSS:RELTokenProfile1.1].

1445 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11  
 1446 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be  
 1447 used as defined in [WSS:RELTokenProfile1.1].

1448

1449 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties  
 1450 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
 1451 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion should  
 1452 be used instead.

## 1453 5.4.10 HttpsToken Assertion

1454 This element represents a requirement for a transport binding to support the use of HTTPS.

### 1455 Syntax

```

1456 <sp:HttpsToken xmlns:sp="..." ... >
1457 (
1458   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1459   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1460 ) ?
1461 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1462 <wsp:Policy xmlns:wsp="...">
1463 (
1464   <sp:HttpBasicAuthentication /> |
1465   <sp:HttpDigestAuthentication /> |
1466   <sp:RequireClientCertificate /> |
1467   ...
1468 ) ?
1469 ...
1470 </wsp:Policy>
1471 ...
1472 </sp:HttpsToken>
  
```

1473 The following describes the attributes and elements listed in the schema outlined above:

1474 /sp:HttpsToken

1475 This identifies an Https assertion stating that use of the HTTPS protocol specification is  
 1476 supported.

- 1477 /sp:HttpsToken/sp:Issuer
- 1478        This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of
- 1479        the `sp:HttpsToken`.
- 1480 /sp:HttpsToken/sp:IssuerName
- 1481        This optional element, of type `xs:anyURI`, contains the logical name of the `sp:HttpsToken` issuer.
- 1482 /sp:HttpsToken/wst:Claims
- 1483        This optional element identifies the required claims that a security token must contain in order to
- 1484        satisfy the token assertion requirements.
- 1485 /sp:HttpsToken/wsp:Policy
- 1486        This required element identifies additional requirements for use of the `sp:HttpsToken` assertion.
- 1487 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication
- 1488        This optional element is a policy assertion that indicates that the client **MUST** use HTTP Basic
- 1489        Authentication [RFC2068] to authenticate to the service.
- 1490 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication
- 1491        This optional element is a policy assertion that indicates that the client **MUST** use HTTP Digest
- 1492        Authentication [RFC2068] to authenticate to the service.
- 1493 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate
- 1494        This optional element is a policy assertion that indicates that the client **MUST** provide a certificate
- 1495        when negotiating the HTTPS session.

### 1496 **5.4.11 KeyValueToken Assertion**

1497 This element represents a requirement for a KeyValue token. The next section defines the KeyValue  
 1498 security token abstraction for purposes of this token assertion.

1499 This document defines requirements for KeyValue token when used in combination with RSA  
 1500 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by  
 1501 introducing new nested assertions besides `sp:RsaKeyValue`.

#### 1503 **Syntax**

```

1504 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1505   <wsp:Policy xmlns:wsp="...">
1506     <sp:RsaKeyValue ... /> ?
1507     ...
1508   </wsp:Policy>
1509   ...
1510 </sp:KeyValueToken>
```

1511 The following describes the attributes listed in the schema outlined above:

- 1512 /sp:KeyValueToken
- 1513        This identifies a RsaToken assertion.
- 1514 /sp:KeyValueToken/@sp:IncludeToken
- 1515        This optional attribute identifies the token inclusion value for this token assertion.
- 1516 /sp:KeyValueToken/wsp:Policy
- 1517        This required element identifies additional requirements for use of the `sp:KeyValueToken`
- 1518        assertion.
- 1519 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1520 This optional element is a policy assertion that indicates that the `ds:RSAKeyValue` element must  
1521 be present in the `KeyValue` token. This indicates that an RSA key pair must be used.

#### 1522 5.4.11.1 KeyValue Token

1523 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key  
1524 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this  
1525 section is to define the `KeyValue` token abstraction that represents such key pair referencing mechanism.

1526 Although the `ds:KeyValue` element as defined in the XML Signature specification is generic enough to be  
1527 used with any asymmetric cryptographic algorithm this document only profiles the usage of `ds:KeyValue`  
1528 element in combination with RSA cryptographic algorithm.

1530 The RSA key pair is represented by the `ds:KeyInfo` element containing the `ds:KeyValue` element with the  
1531 RSA public key value in `ds:RSAKeyValue` as defined in the XML Signature specification:

```
1532 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1533   <ds:KeyValue>  
1534     <ds:RSAKeyValue>  
1535       <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1536       <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1537     </ds:RSAKeyValue>  
1538   </ds:KeyValue>  
1539 </ds:KeyInfo>
```

1541 When the `KeyValue` token is used the corresponding public key value appears directly in the signature or  
1542 encrypted data `ds:KeyInfo` element like in the following example. There is no `KeyValue` token  
1543 manifestation outside the `ds:KeyInfo` element.

```
1544 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1545   <SignedInfo>  
1546     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1547 c14n#" />  
1548     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1549     <Reference URI="#_1">  
1550       <Transforms>  
1551         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1552       </Transforms>  
1553       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
1554       <DigestValue>...</DigestValue>  
1555     </Reference>  
1556   </SignedInfo>  
1557   <SignatureValue>...</SignatureValue>  
1558   <KeyInfo>  
1559     <KeyValue>  
1560       <RSAKeyValue>  
1561         <Modulus>...</Modulus>  
1562         <Exponent>...</Exponent>  
1563       </RSAKeyValue>  
1564     </KeyValue>  
1565   </KeyInfo>  
1566 </Signature>
```

1568 Since there is no representation of the `KeyValue` token outside the `ds:KeyInfo` element and thus no  
1569 identifier can be associated with the token, the `KeyValue` token cannot be referenced by using  
1570 `wsse:SecurityTokenReference` element. However the `ds:KeyInfo` element representing the `KeyValue`  
1571 token can be used whenever a security token can be used as illustrated on the following example:

```
1572 <t:RequestSecurityToken xmlns:t="...">  
1573   <t:RequestType>...</t:RequestType>  
1574   ...  
1575   <t:UseKey>  
1576     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
1578     <KeyValue>
1579         <RSAKeyValue>
1580             <Modulus>...</Modulus>
1581             <Exponent>...</Exponent>
1582         </RSAKeyValue>
1583     </KeyValue>
1584 </KeyInfo>
1585 </t:UseKey>
1586 </t:RequestSecurityToken>
```

1587

## 6 Security Binding Properties

1588 This section defines the various properties or conditions of a security binding, their semantics, values and  
1589 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are  
1590 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that  
1591 populates a value of a property appears in a policy, that property is set to the value indicated by the  
1592 assertion. The security binding then uses the value of the property to control its behavior. The properties  
1593 listed here are common to the various security bindings described in Section 7. Assertions that define  
1594 values for these properties are defined in Section 7. The following properties are used by the security  
1595 binding assertions.

### 1596 6.1 [Algorithm Suite] Property

1597 This property specifies the algorithm suite required for performing cryptographic operations with  
1598 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and  
1599 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This  
1600 property defines the set of available algorithms. The value of this property is typically referenced by a  
1601 security binding and is used to specify the algorithms used for all message level cryptographic operations  
1602 performed under the security binding.

1603 Note: In some cases, this property MAY be referenced under a context other than a security binding and  
1604 used to control the algorithms used under that context. For example, supporting token assertions define  
1605 such a context. In such contexts, the specified algorithms still apply to message level cryptographic  
1606 operations.

1607 An algorithm suite defines values for each of the following operations and properties:

- 1608 • [Sym Sig] Symmetric Key Signature
- 1609 • [Asym Sig] Signature with an asymmetric key
- 1610 • [Dig] Digest
- 1611 • [Enc] Encryption
- 1612 • [Sym KW] Symmetric Key Wrap
- 1613 • [Asym KW] Asymmetric Key Wrap
- 1614 • [Comp Key] Computed key
- 1615 • [Enc KD] Encryption key derivation
- 1616 • [Sig KD] Signature key derivation
- 1617 • [Min SKL] Minimum symmetric key length
- 1618 • [Max SKL] Maximum symmetric key length
- 1619 • [Min AKL] Minimum asymmetric key length
- 1620 • [Max AKL] Maximum asymmetric key length

1621

1622 The following table provides abbreviations for the algorithm URI used in the table below:

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

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](http://www.w3.org/2001/04/xmlenc#rsa-1_5)  
 PSha1 [http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p\\_sha1](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](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](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](http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1)  
 XPath <http://www.w3.org/TR/1999/REC-xpath-19991116>  
 XPath20 <http://www.w3.org/2002/06/xmldsig-filter2>  
 C14n <http://www.w3.org/2001/10/xml-c14n#>  
 ExC14n <http://www.w3.org/2001/10/xml-exc-c14n#>  
 SNT <http://www.w3.org/TR/soap12-n11n>  
 STRT10 <http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-soap-message-security-1.0#STR-Transform>  
 AbsXPath <http://docs.oasis-open.org/...TBD.../AbsXPath>

1623

1624 The tables below show all the base algorithm suites defined by this specification. This table defines  
 1625 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

1626 This table defines additional properties whose values can be specified along with the default value for that  
 1627 property.

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

1628 This table defines values for the remaining components for each algorithm suite.

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



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

## 1629 6.2 [Timestamp] Property

1630 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`  
1631 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected  
1632 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be  
1633 present. The default value for this property is 'false'.

## 1634 6.3 [Protection Order] Property

1635 This property indicates the order in which integrity and confidentiality are applied to the message, in  
1636 cases where both integrity and confidentiality are required:

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

1637 The default value for this property is 'SignBeforeEncrypting'.

## 1638 6.4 [Signature Protection] Property

1639 This boolean property specifies whether the signature must be encrypted. If the value is 'true', the primary  
1640 signature MUST be encrypted and any signature confirmation elements MUST also be encrypted. The  
1641 primary signature element is not required to be encrypted if the value is 'true' when there is nothing else  
1642 in the message that is encrypted. If the value is 'false', the primary signature MUST NOT be encrypted  
1643 and any signature confirmation elements MUST NOT be encrypted. The default value for this property is  
1644 'false'.

## 1645 6.5 [Token Protection] Property

1646 This boolean property specifies whether signatures must cover the token used to generate that signature.  
1647 If the value is 'true', then each token used to generate a signature MUST be covered by that signature. If  
1648 the value is 'false', then the token MUST NOT be covered by the signature. Note that in cases where  
1649 derived keys are used the 'main' token, and NOT the derived key token, is covered by the signature. It is  
1650 recommended that assertions that define values for this property apply to [Endpoint Policy Subject]. The  
1651 default value for this property is 'false'.

## 1652 **6.6 [Entire Header and Body Signatures] Property**

1653 This boolean property specifies whether signature digests over the SOAP body and SOAP headers must  
1654 only cover the entire body and entire header elements. If the value is 'true', then each digest over the  
1655 SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In  
1656 addition each digest over a SOAP header MUST be over an actual header element and not a descendant  
1657 of a header element. This restriction does not specifically apply to the wsse:Security header. However  
1658 signature digests over child elements of the wsse:Security header MUST be over the entire child element  
1659 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a  
1660 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to  
1661 'true' mitigates against some possible re-writing attacks. It is recommended that assertions that define  
1662 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

## 1663 **6.7 [Security Header Layout] Property**

1664 This property indicates which layout rules to apply when adding items to the security header. The  
1665 following table shows which rules are defined by this specification.

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

1666

### 1667 **6.7.1 Strict Layout Rules for WSS 1.0**

- 1668 1. Tokens that are included in the message MUST be declared before use. For example:
- 1669 a. A local signing token MUST occur before the signature that uses it.
- 1670 b. A local token serving as the source token for a derived key token MUST occur before that
- 1671 derived key token.
- 1672 c. A local encryption token MUST occur before the reference list that points to
- 1673 xenc:EncryptedData elements that use it.
- 1674 d. If the same token is used for both signing and encryption, then it should appear before
- 1675 the ds:Signature and xenc:ReferenceList elements in the security header that are
- 1676 generated using the token.
- 1677 2. Signed elements inside the security header MUST occur before the signature that signs them.
- 1678 For example:
- 1679 a. A timestamp MUST occur before the signature that signs it.

- 1680                   b. A Username token (usually in encrypted form) MUST occur before the signature that  
1681                   signs it.
- 1682                   c. A primary signature MUST occur before the supporting token signature that signs the  
1683                   primary signature's signature value element.
- 1684           3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1685           has the same order requirements as the source plain text element, unless requirement 4  
1686           indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1687           supporting token signature per 2.c above and an encrypted token has the same ordering  
1688           requirements as the unencrypted token.

1689   If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top  
1690   level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the  
1691   security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any  
1692   xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict  
1693   Layout Rules for WSS 1.1

- 1694           1. Tokens that are included in the message MUST be declared before use. For example:
- 1695                   a. A local signing token MUST occur before the signature that uses it.
- 1696                   b. A local token serving as the source token for a derived key token MUST occur before that  
1697                   derived key token.
- 1698                   c. A local encryption token MUST occur before the reference list that points to  
1699                   xenc:EncryptedData elements that use it.
- 1700                   d. If the same token is used for both signing and encryption, then it should appear before  
1701                   the ds:Signature and xenc:ReferenceList elements in the security header that are  
1702                   generated using the token.
- 1703           2. Signed elements inside the security header MUST occur before the signature that signs them.  
1704           For example:
- 1705                   a. A timestamp MUST occur before the signature that signs it.
- 1706                   b. A Username token (usually in encrypted form) MUST occur before the signature that  
1707                   signs it.
- 1708                   c. A primary signature MUST occur before the supporting token signature that signs the  
1709                   primary signature's signature value element.
- 1710                   d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1711           3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1712           has the same order requirements as the source plain text element, unless requirement 4  
1713           indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1714           supporting token signature per 2.c above and an encrypted token has the same ordering  
1715           requirements as the unencrypted token.
- 1716           4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element  
1717           MUST be present in the security header. The xenc:ReferenceList MUST occur before any  
1718           xenc:EncryptedData elements in the security header that are referenced from the reference list.  
1719           However, the xenc:ReferenceList is not required to appear before independently encrypted  
1720           tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1721           5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)  
1722           1.1] MUST obey rule 1 above.

---

## 1723 7 Security Binding Assertions

1724 The appropriate representation of the different facets of security mechanisms requires distilling the  
1725 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy  
1726 scope of assertions defined in this section is the policy scope of their containing element.

### 1727 7.1 AlgorithmSuite Assertion

1728 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]  
1729 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

#### 1730 Syntax

```
1731 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1732   <wsp:Policy xmlns:wsp="...">  
1733     (<sp:Basic256 ... /> |  
1734     <sp:Basic192 ... /> |  
1735     <sp:Basic128 ... /> |  
1736     <sp:TripleDes ... /> |  
1737     <sp:Basic256Rsa15 ... /> |  
1738     <sp:Basic192Rsa15 ... /> |  
1739     <sp:Basic128Rsa15 ... /> |  
1740     <sp:TripleDesRsa15 ... /> |  
1741     <sp:Basic256Sha256 ... /> |  
1742     <sp:Basic192Sha256 ... /> |  
1743     <sp:Basic128Sha256 ... /> |  
1744     <sp:TripleDesSha256 ... /> |  
1745     <sp:Basic256Sha256Rsa15 ... /> |  
1746     <sp:Basic192Sha256Rsa15 ... /> |  
1747     <sp:Basic128Sha256Rsa15 ... /> |  
1748     <sp:TripleDesSha256Rsa15 ... /> |  
1749     ...)  
1750     <sp:InclusiveC14N ... /> ?  
1751     <sp:SOAPNormalization10 ... /> ?  
1752     <sp:STRTransform10 ... /> ?  
1753     (<sp:XPath10 ... /> |  
1754     <sp:XPathFilter20 ... /> |  
1755     <sp:AbsXPath ... /> |  
1756     ...)?  
1757     ...  
1758   </wsp:Policy>  
1759   ...  
1760 </sp:AlgorithmSuite>
```

1761  
1762 The following describes the attributes and elements listed in the schema outlined above:

1763 /sp:AlgorithmSuite

1764       This identifies an AlgorithmSuite assertion.

1765 /sp:AlgorithmSuite/wsp:Policy

1766       This required element contains one or more policy assertions that indicate the specific algorithm  
1767       suite to use.

1768 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1769       This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1770       to 'Basic256'.

1771 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

1772            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1773            to 'Basic192'.

1774    /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1775            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1776            to 'Basic128'.

1777    /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1778            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1779            to 'TripleDes'.

1780    /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1781            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1782            to 'Basic256Rsa15'.

1783    /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1784            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1785            to 'Basic192Rsa15'.

1786    /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1787            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1788            to 'Basic128Rsa15'.

1789    /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1790            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1791            to 'TripleDesRsa15'.

1792    /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1793            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1794            to 'Basic256Sha256'.

1795    /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1796            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1797            to 'Basic192Sha256'.

1798    /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1799            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1800            to 'Basic128Sha256'.

1801    /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1802            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1803            to 'TripleDesSha256'.

1804    /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1805            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1806            to 'Basic256Sha256Rsa15'.

1807    /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1808            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1809            to 'Basic192Sha256Rsa15'.

1810    /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1811            This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1812            to 'Basic128Sha256Rsa15'.

1813    /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1814 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set  
1815 to 'TripleDesSha256Rsa15'.

1816 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1817 This optional element is a policy assertion that indicates that the [C14N] property of an algorithm  
1818 suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N] property is  
1819 'ExcC14N'.

1820 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1821 This optional element is a policy assertion that indicates that the [SOAP Norm] property is set to  
1822 'SNT'.

1823 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1824 This optional element is a policy assertion that indicates that the [STR Transform] property is set  
1825 to 'STRT10'.

1826 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1827 This optional element is a policy assertion that indicates that the [XPath] property is set to 'XPath'.

1828 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1829 This optional element is a policy assertion that indicates that the [XPath] property is set to  
1830 'XPath20'.

1831 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1832 This optional element is a policy assertion that indicates that the [XPath] property is set to  
1833 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1834

## 1835 7.2 Layout Assertion

1836 This assertion indicates a requirement for a particular security header layout as defined under the  
1837 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its  
1838 containing assertion.

### 1839 Syntax

```
1840 <sp:Layout xmlns:sp="..." ... >  
1841   <wsp:Policy xmlns:wsp="...">  
1842     <sp:Strict ... /> |  
1843     <sp:Lax ... /> |  
1844     <sp:LaxTsFirst ... /> |  
1845     <sp:LaxTsLast ... /> |  
1846     ...  
1847   </wsp:Policy>  
1848   ...  
1849 </sp:Layout>
```

1850

1851 The following describes the attributes and elements listed in the schema outlined above:

1852 /sp:Layout

1853 This identifies a Layout assertion.

1854 /sp:Layout/wsp:Policy

1855 This required element contains one or more policy assertions that indicate the specific security  
1856 header layout to use.

1857 /sp:Layout/wsp:Policy/sp:Strict

1858 This optional element is a policy assertion that indicates that the [Security Header Layout]  
1859 property is set to 'Strict'.

1860 /sp:Layout/wsp:Policy/sp:Lax

1861 This optional element is a policy assertion that indicates that the [Security Header Layout]  
1862 property is set to 'Lax'.

1863 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1864 This optional element is a policy assertion that indicates that the [Security Header Layout]  
1865 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to  
1866 'true' by the presence of an sp:IncludeTimestamp assertion.

1867 /sp:Layout/wsp:Policy/sp:LaxTsLast

1868 This optional element is a policy assertion that indicates that the [Security Header Layout]  
1869 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to  
1870 'true' by the presence of an sp:IncludeTimestamp assertion.

### 1871 7.3 TransportBinding Assertion

1872 The TransportBinding assertion is used in scenarios in which message protection and security correlation  
1873 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like  
1874 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by  
1875 the transport. This binding has one binding specific token property; [Transport Token]. This assertion  
1876 MUST apply to [Endpoint Policy Subject].

#### 1877 Syntax

```
1878 <sp:TransportBinding xmlns:sp="..." ... >  
1879   <wsp:Policy xmlns:wsp="...">  
1880     <sp:TransportToken ... >  
1881       <wsp:Policy> ... </wsp:Policy>  
1882       ...  
1883     </sp:TransportToken>  
1884     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1885     <sp:Layout ... > ... </sp:Layout> ?  
1886     <sp:IncludeTimestamp ... /> ?  
1887     ...  
1888   </wsp:Policy>  
1889   ...  
1890 </sp:TransportBinding>
```

1891  
1892 The following describes the attributes and elements listed in the schema outlined above:

1893 /sp:TransportBinding

1894 This identifies a TransportBinding assertion.

1895 /sp:TransportBinding/wsp:Policy

1896 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding  
1897 assertion.

1898 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1899 This required element is a policy assertion that indicates a requirement for a Transport Token.  
1900 The specified token populates the [Transport Token] property and indicates how the transport is  
1901 secured.

1902 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1903 This indicates a nested policy that identifies the type of Transport Token to use.

- 1904 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite
- 1905 This required element is a policy assertion that indicates a value that populates the [Algorithm
- 1906 Suite] property. See Section 6.1 for more details.
- 1907 /sp:TransportBinding/wsp:Policy/sp:Layout
- 1908 This optional element is a policy assertion that indicates a value that populates the [Security
- 1909 Header Layout] property. See Section 6.7 for more details.
- 1910 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp
- 1911 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
- 1912 'true'.

## 1913 7.4 SymmetricBinding Assertion

1914 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means

1915 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;

1916 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this

1917 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to

1918 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to

1919 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token

1920 properties and is used as the basis for both encryption and signature in both directions. This assertion

1921 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

### 1922 Syntax

```

1923 <sp:SymmetricBinding xmlns:sp="..." ... >
1924   <wsp:Policy xmlns:wsp="...">
1925     (
1926       <sp:EncryptionToken ... >
1927         <wsp:Policy> ... </wsp:Policy>
1928       </sp:EncryptionToken>
1929       <sp:SignatureToken ... >
1930         <wsp:Policy> ... </wsp:Policy>
1931       </sp:SignatureToken>
1932     ) | (
1933       <sp:ProtectionToken ... >
1934         <wsp:Policy> ... </wsp:Policy>
1935       </sp:ProtectionToken>
1936     )
1937     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1938     <sp:Layout ... > ... </sp:Layout> ?
1939     <sp:IncludeTimestamp ... /> ?
1940     <sp:EncryptBeforeSigning ... /> ?
1941     <sp:EncryptSignature ... /> ?
1942     <sp:ProtectTokens ... /> ?
1943     <sp:OnlySignEntireHeadersAndBody ... /> ?
1944     ...
1945   </wsp:Policy>
1946   ...
1947 </sp:SymmetricBinding>

```

1948

1949 The following describes the attributes and elements listed in the schema outlined above:

- 1950 /sp:SymmetricBinding
- 1951 This identifies a SymmetricBinding assertion.
- 1952 /sp:SymmetricBinding/wsp:Policy
- 1953 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
- 1954 assertion.



- 1955 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
- 1956 This optional element is a policy assertion that indicates a requirement for an Encryption Token.  
1957 The specified token populates the [Encryption Token] property and is used for encryption. It is an  
1958 error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.
- 1959 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
- 1960 The policy contained here MUST identify exactly one token to use for encryption.
- 1961 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
- 1962 This optional element is a policy assertion that indicates a requirement for a Signature Token.  
1963 The specified token populates the [Signature Token] property and is used for the message  
1964 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be  
1965 specified.
- 1966 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
- 1967 The policy contained here MUST identify exactly one token to use for signatures.
- 1968 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
- 1969 This optional element is a policy assertion that indicates a requirement for a Protection Token.  
1970 The specified token populates the [Encryption Token] and [Signature Token properties] and is  
1971 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken  
1972 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be  
1973 specified.
- 1974 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
- 1975 The policy contained here MUST identify exactly one token to use for protection.
- 1976 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
- 1977 This required element is a policy assertion that indicates a value that populates the [Algorithm  
1978 Suite] property. See Section 6.1 for more details.
- 1979 /sp:SymmetricBinding/wsp:Policy/sp:Layout
- 1980 This optional element is a policy assertion that indicates a value that populates the [Security  
1981 Header Layout] property. See Section 6.7 for more details.
- 1982 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
- 1983 This optional element is a policy assertion that indicates that the [Timestamp] property is set to  
1984 'true'.
- 1985 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
- 1986 This optional element is a policy assertion that indicates that the [Protection Order] property is set  
1987 to 'EncryptBeforeSigning'.
- 1988 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
- 1989 This optional element is a policy assertion that indicates that the [Signature Protection] property is  
1990 set to 'true'.
- 1991 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
- 1992 This optional element is a policy assertion that indicates that the [Token Protection] property is  
1993 set to 'true'.
- 1994 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
- 1995 This optional element is a policy assertion that indicates that the [Entire Header And Body  
1996 Signatures] property is set to 'true'.

## 1997 **7.5 AsymmetricBinding Assertion**

1998 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means  
1999 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly  
2000 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and  
2001 signature. However it is also common practice to use distinct keys for encryption and signature, because  
2002 of their different lifecycles.

2003  
2004 This binding enables either of these practices by means of four binding specific token properties: [Initiator  
2005 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption  
2006 Token].

2007  
2008 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator  
2009 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient  
2010 Encryption Token] will both refer to the same token.

2011  
2012 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator  
2013 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient  
2014 Encryption Token] will refer to different tokens.

2015  
2016 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the  
2017 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response  
2018 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the  
2019 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for  
2020 the message encryption from initiator to the recipient. Note that in each case, the token is associated with  
2021 the party (initiator or recipient) who knows the secret.

2022 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy  
2023 Subject].

### 2024 **Syntax**

```
2025 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2026   <wsp:Policy xmlns:wsp="...">  
2027     (  
2028       <sp:InitiatorToken>  
2029         <wsp:Policy> ... </wsp:Policy>  
2030       </sp:InitiatorToken>  
2031     ) | (  
2032       <sp:InitiatorSignatureToken>  
2033         <wsp:Policy> ... </wsp:Policy>  
2034       </sp:InitiatorSignatureToken>  
2035       <sp:InitiatorEncryptionToken>  
2036         <wsp:Policy> ... </wsp:Policy>  
2037       </sp:InitiatorEncryptionToken>  
2038     )  
2039     (  
2040       <sp:RecipientToken>  
2041         <wsp:Policy> ... </wsp:Policy>  
2042       </sp:RecipientToken>  
2043     ) | (  
2044       <sp:RecipientSignatureToken>  
2045         <wsp:Policy> ... </wsp:Policy>  
2046       </sp:RecipientSignatureToken>  
2047       <sp:RecipientEncryptionToken>  
2048         <wsp:Policy> ... </wsp:Policy>
```

```

2049     </sp:RecipientEncryptionToken>
2050   )
2051   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2052   <sp:Layout ... > ... </sp:Layout> ?
2053   <sp:IncludeTimestamp ... /> ?
2054   <sp:EncryptBeforeSigning ... /> ?
2055   <sp:EncryptSignature ... /> ?
2056   <sp:ProtectTokens ... /> ?
2057   <sp:OnlySignEntireHeadersAndBody ... /> ?
2058   ...
2059 </wsp:Policy>
2060   ...
2061 </sp:AsymmetricBinding>

```

2062

2063 The following describes the attributes and elements listed in the schema outlined above:

2064 /sp:AsymmetricBinding

2065 This identifies a AsymmetricBinding assertion.

2066 /sp:AsymmetricBinding/wsp:Policy

2067 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding  
 2068 assertion.

2069 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2070 This optional element is a policy assertion that indicates a requirement for an Initiator Token. The  
 2071 specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]  
 2072 properties and is used for the message signature from initiator to recipient, and encryption from  
 2073 recipient to initiator.

2074 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2075 The policy contained here MUST identify one or more token assertions.

2076 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2077 This optional element is a policy assertion that indicates a requirement for an Initiator Signature  
 2078 Token. The specified token populates the [Initiator Signature Token] property and is used for the  
 2079 message signature from initiator to recipient.

2080 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2081 The policy contained here MUST identify one or more token assertions.

2082 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2083 This optional element is a policy assertion that indicates a requirement for an Initiator Encryption  
 2084 Token. The specified token populates the [Initiator Encryption Token] property and is used for the  
 2085 message encryption from recipient to initiator.

2086 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2087 The policy contained here MUST identify one or more token assertions.

2088 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2089 This optional element is a policy assertion that indicates a requirement for a Recipient Token. The  
 2090 specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]  
 2091 property and is used for encryption from initiator to recipient, and for the message signature from  
 2092 recipient to initiator.

2093 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2094 The policy contained here MUST identify one or more token assertions.

2095 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2096 This optional element is a policy assertion that indicates a requirement for a Recipient Signature  
2097 Token. The specified token populates the [Recipient Signature Token] property and is used for  
2098 the message signature from Recipient to recipient.

2099 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy  
2100 The policy contained here MUST identify one or more token assertions.

2101 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken  
2102 This optional element is a policy assertion that indicates a requirement for a Recipient Encryption  
2103 Token. The specified token populates the [Recipient Encryption Token] property and is used for  
2104 the message encryption from recipient to Recipient.

2105 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy  
2106 The policy contained here MUST identify one or more token assertions.

2107 /sp:AsymmetricBinding/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:AsymmetricBinding/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:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
2114 This optional element is a policy assertion that indicates that the [Timestamp] property is set to  
2115 'true'.

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

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

2122 /sp:AsymmetricBinding/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:AsymmetricBinding/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

## 8 Supporting Tokens

2129 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to  
2130 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore  
2131 be referred to as the “message signature”. In case of Transport Binding the message is signed outside of  
2132 the message XML by the underlying transport protocol and the signature itself is not part of the message.  
2133 Additional tokens may be specified to augment the claims provided by the token associated with the  
2134 “message signature” provided by the Security Binding. This section defines seven properties related to  
2135 supporting token requirements which may be referenced by a Security Binding: [Supporting Tokens],  
2136 [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens],  
2137 [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing  
2138 Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties:  
2139 SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,  
2140 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,  
2141 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These  
2142 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy  
2143 Subject] or [Operation Policy Subject].

2144

2145 Supporting tokens may be specified at a different scope than the binding assertion which provides  
2146 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while  
2147 the supporting tokens might be defined at the scope of a message. When assertions that populate this  
2148 property are defined in overlapping scopes, the sender should merge the requirements by including all  
2149 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

2150

2151 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the  
2152 tokens should sign and encrypt the various message parts. In such cases ordering of elements (tokens,  
2153 signatures, reference lists etc.) in the security header would be used to determine which order signature  
2154 and encryptions occurred in.

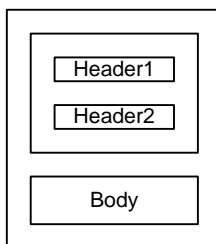
2155

2156 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional  
2157 constraints defined by the supporting token assertion. For example, if the supporting token assertion  
2158 specifies message parts that need to be encrypted, the specified tokens need to be capable of  
2159 encryption.

2160

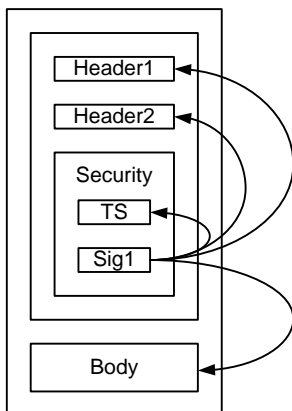
2161 To illustrate the different ways that supporting tokens may be bound to the message, let’s consider a  
2162 message with three components: Header1, Header2, and Body.

2163

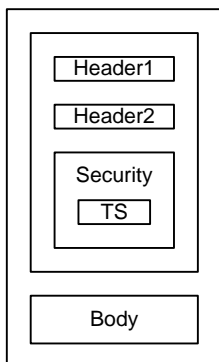


2164

2165 Even before any supporting tokens are added, each binding requires that the message is signed using a  
 2166 token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts  
 2167 of the message including the message timestamp (TS) facilitate replay detection. The signature is then  
 2168 included as part of the Security header as illustrated below:  
 2169



2170  
 2171 Note: if required, the initiator may also include in the Security header the token used as the basis for the  
 2172 message signature (Sig1), not shown in the diagram.  
 2173 If transport security is used, only the message timestamp (TS) is included in the Security header as  
 2174 illustrated below. The “message signature” is provided by the underlying transport protocol and is not part  
 2175 of the message XML.



2176  
 2177 **8.1 SupportingTokens Assertion**

2178 Supporting tokens are included in the security header and may optionally include additional message  
 2179 parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and do not  
 2180 require any protection (signature or encryption) to be applied to the message before they are added.  
 2181 More specifically there is no requirement on “message signature” being present before the supporting  
 2182 tokens are added. However it is RECOMMENDED to employ underlying protection mechanism to ensure  
 2183 that the supporting tokens are cryptographically bound to the message during the transmission.

2184 **Syntax**

```
2185 <sp:SupportingTokens xmlns:sp="..." ... >
2186   <wsp:Policy xmlns:wsp="...">
2187     [Token Assertion]+
2188     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2189     (
2190       <sp:SignedParts ... > ... </sp:SignedParts> |
2191       <sp:SignedElements ... > ... </sp:SignedElements> |
```

2192  
2193  
2194  
2195  
2196  
2197  
2198

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

2199

2200 The following describes the attributes and elements listed in the schema outlined above:

2201 /sp:SupportingTokens

2202 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting  
2203 Tokens] property.

2204 /sp:SupportingTokens/wsp:Policy

2205 This describes additional requirements for satisfying the SupportingTokens assertion.

2206 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2207 The policy MUST identify one or more token assertions.

2208 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2209 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and  
2210 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2211 by this policy assertion.

2212 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2213 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and  
2214 describes additional message parts that MUST be included in the signature generated with the  
2215 token identified by this policy assertion.

2216 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2217 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and  
2218 describes additional message elements that MUST be included in the signature generated with  
2219 the token identified by this policy assertion.

2220 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2221 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and  
2222 describes additional message parts that MUST be encrypted using the token identified by this  
2223 policy assertion.

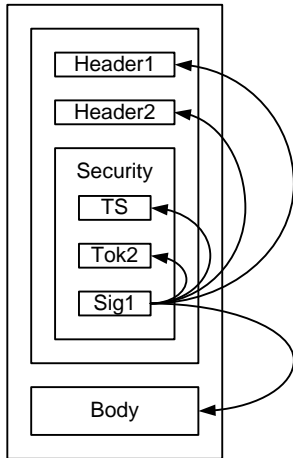
2224 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2225 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and  
2226 describes additional message elements that MUST be encrypted using the token identified by this  
2227 policy assertion.

## 2228 **8.2 SignedSupportingTokens Assertion**

2229 Signed tokens are included in the “message signature” as defined above and may optionally include  
2230 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token  
2231 (Tok2) is signed by the message signature (Sig1):

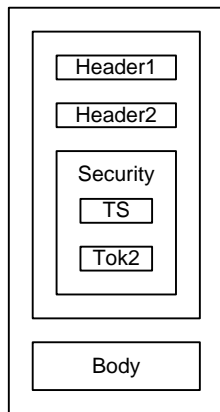
2232



2233

2234 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2235



2236

2237 **Syntax**

```

2238 <sp:SignedSupportingTokens xmlns:sp="..." ... >
2239   <wsp:Policy xmlns:wsp="...">
2240     [Token Assertion]+
2241     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2242     (
2243       <sp:SignedParts ... > ... </sp:SignedParts> |
2244       <sp:SignedElements ... > ... </sp:SignedElements> |
2245       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2246       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2247     ) *
2248     ...
2249   </wsp:Policy>
2250   ...
2251 </sp:SignedSupportingTokens>

```

2252

2253 The following describes the attributes and elements listed in the schema outlined above:

2254 /sp:SignedSupportingTokens

2255 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed  
2256 Supporting Tokens] property.

2257 /sp:SignedSupportingTokens/wsp:Policy

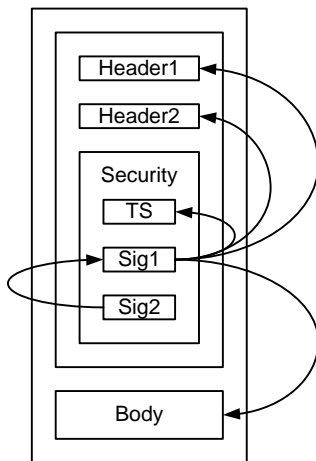
2258 This describes additional requirements for satisfying the SignedSupportingTokens assertion.



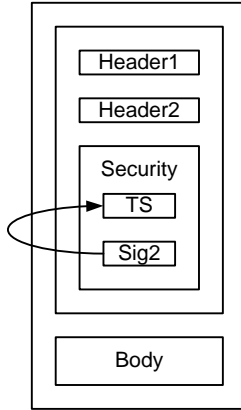
- 2259 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]  
 2260 The policy MUST identify one or more token assertions.
- 2261 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite  
 2262 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and  
 2263 describes the algorithms to use for cryptographic operations performed with the tokens identified  
 2264 by this policy assertion.
- 2265 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts  
 2266 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and  
 2267 describes additional message parts that MUST be included in the signature generated with the  
 2268 token identified by this policy assertion.
- 2269 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements  
 2270 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and  
 2271 describes additional message elements that MUST be included in the signature generated with  
 2272 the token identified by this policy assertion.
- 2273 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts  
 2274 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and  
 2275 describes additional message parts that MUST be encrypted using the token identified by this  
 2276 policy assertion.
- 2277 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements  
 2278 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and  
 2279 describes additional message elements that MUST be encrypted using the token identified by this  
 2280 policy assertion.

### 2281 8.3 EndorsingSupportingTokens Assertion

2282 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element  
 2283 produced from the message signature and may optionally include additional message parts to sign and/or  
 2284 encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message signature  
 2285 (Sig1):  
 2286



2287  
 2288 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated  
 2289 below:  
 2290



2291

2292 **Syntax**

2293  
2294  
2295  
2296  
2297  
2298  
2299  
2300  
2301  
2302  
2303  
2304  
2305  
2306

```

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

```

2307

2308 The following describes the attributes and elements listed in the schema outlined above:

2309 /sp:EndorsingSupportingTokens

2310 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the  
2311 [Endorsing Supporting Tokens] property.

2312 /sp:EndorsingSupportingTokens/wsp:Policy

2313 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2314 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2315 The policy MUST identify one or more token assertions.

2316 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2317 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and  
2318 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2319 by this policy assertion.

2320 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2321 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and  
2322 describes additional message parts that MUST be included in the signature generated with the  
2323 token identified by this policy assertion.

2324 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2325 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and  
2326 describes additional message elements that MUST be included in the signature generated with  
2327 the token identified by this policy assertion.

2328 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2329 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and  
2330 describes additional message parts that MUST be encrypted using the token identified by this  
2331 policy assertion.

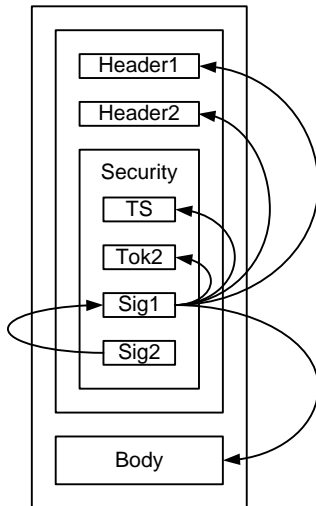
2332 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2333 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and  
2334 describes additional message elements that MUST be encrypted using the token identified by this  
2335 policy assertion.

## 2336 8.4 SignedEndorsingSupportingTokens Assertion

2337 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature  
2338 and are themselves signed by that message signature, that is both tokens (the token used for the  
2339 message signature and the signed endorsing token) sign each other. This assertion may optionally  
2340 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed  
2341 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the  
2342 message signature (Sig1):

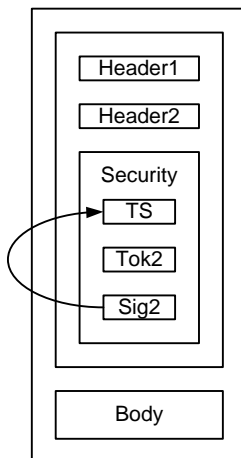
2343



2344

2345 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)  
2346 should cover the message timestamp as illustrated below:

2347



2348

2349 **Syntax**

```
2350 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2351   <wsp:Policy xmlns:wsp="...">
2352     [Token Assertion]+
2353     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2354     (
2355       <sp:SignedParts ... > ... </sp:SignedParts> |
2356       <sp:SignedElements ... > ... </sp:SignedElements> |
2357       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2358       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2359     ) *
2360     ...
2361   </wsp:Policy>
2362   ...
2363 </sp:SignedEndorsingSupportingTokens>
```

2364

2365 The following describes the attributes and elements listed in the schema outlined above:

2366 /sp:SignedEndorsingSupportingTokens

2367 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the  
2368 [Signed Endorsing Supporting Tokens] property.

2369 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2370 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2371 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2372 The policy MUST identify one or more token assertions.

2373 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2374 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and  
2375 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2376 by this policy assertion.

2377 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2378 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and  
2379 describes additional message parts that MUST be included in the signature generated with the  
2380 token identified by this policy assertion.

2381 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2382 This optional element follows the schema outlined in Section 4.1.2 and describes additional  
2383 message elements that MUST be included in the signature generated with the token identified by  
2384 this policy assertion.

2385 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2386 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and  
2387 describes additional message parts that MUST be encrypted using the token identified by this  
2388 policy assertion.

2389 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2390 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and  
2391 describes additional message elements that MUST be encrypted using the token identified by this  
2392 policy assertion.

## 2393 **8.5 SignedEncryptedSupportingTokens Assertion**

2394 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also  
2395 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2396 encrypting the supporting tokens.

2397 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of  
2398 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2399 sp:SignedSupportingTokens assertion.

## 2400 **8.6 EncryptedSupportingTokens Assertion**

2401 Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in  
2402 the security header and MUST be encrypted when they appear in the security header.  
2403 Element encryption SHOULD be used for encrypting these tokens. The encrypted supporting  
2404 tokens can be added to any SOAP message and do not require the "message signature"  
2405 being present before the encrypted supporting tokens are added.

2406 The syntax for the sp:EncryptedSupportingTokens differs from the syntax of  
2407 sp:SupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2408 sp:SupportingTokens assertion.

2409 The encrypted supporting tokens SHOULD be used only when the sender cannot provide the  
2410 "message signature" and it is RECOMMENDED that the receiver employs some security  
2411 mechanisms external to the message to prevent the spoofing attacks. In all other cases it is  
2412 RECOMMENDED to use signed encrypted supporting tokens instead to ensure that the  
2413 encrypted tokens are cryptographically bound to the message (See section 8.5).

## 2414 **8.7 EndorsingEncryptedSupportingTokens Assertion**

2415 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also  
2416 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2417 encrypting the supporting tokens.

2418 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of  
2419 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2420 sp:EndorsingSupportingTokens assertion.

## 2421 **8.8 SignedEndorsingEncryptedSupportingTokens Assertion**

2422 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section  
2423 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD  
2424 be used for encrypting the supporting tokens.

2425 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of  
2426 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per  
2427 the sp:SignedEndorsingSupportingTokens assertion.

## 2428 **8.9 Interaction between [Token Protection] property and supporting 2429 token assertions**

2430 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that  
2431 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2432 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or  
2433 the [Signature Token] in the Symmetric binding case, covers that token.
- 2434 • Endorsing signatures cover the main signature and the endorsing token.

- 2435       • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the  
2436       message signature and once by the endorsing signature.

2437 In addition, signed supporting tokens are covered by the message signature, although this is independent  
2438 of [Token Protection].

## 2439 8.10 Example

2440 Example policy containing supporting token assertions:

```
2441 <!-- Example Endpoint Policy -->
2442 <wsp:Policy xmlns:wsp="...">
2443   <sp:SymmetricBinding xmlns:sp="...">
2444     <wsp:Policy>
2445       <sp:ProtectionToken>
2446         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2447           <sp:Issuer>...</sp:Issuer>
2448           <sp:RequestSecurityTokenTemplate>
2449             ...
2450           </sp:RequestSecurityTokenTemplate>
2451         </sp:IssuedToken>
2452       </sp:ProtectionToken>
2453     <sp:AlgorithmSuite>
2454       <wsp:Policy>
2455         <sp:Basic256 />
2456       </wsp:Policy>
2457     </sp:AlgorithmSuite>
2458     ...
2459   </wsp:Policy>
2460 </sp:SymmetricBinding>
2461 ...
2462 <sp:SignedSupportingTokens>
2463   <wsp:Policy>
2464     <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2465   </wsp:Policy>
2466 </sp:SignedSupportingTokens>
2467 <sp:SignedEndorsingSupportingTokens>
2468   <wsp:Policy>
2469     <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2470       <wsp:Policy>
2471         <sp:WssX509v3Token10 />
2472       </wsp:Policy>
2473     </sp:X509Token>
2474   </wsp:Policy>
2475 </sp:SignedEndorsingSupportingTokens>
2476   ...
2477 </wsp:Policy>
```

2478 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be  
2479 included in the security header and covered by the message signature. The  
2480 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the  
2481 security header and covered by the message signature. In addition, a signature over the message  
2482 signature based on the key material associated with the X509 certificate must be included in the security  
2483 header.

2484

## 9 WSS: SOAP Message Security Options

2485 There are several optional aspects to the WSS: SOAP Message Security specification that are  
2486 independent of the trust and token taxonomies. This section describes another class of properties and  
2487 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The  
2488 assertions defined here MUST apply to [Endpoint Policy Subject].

2489 The properties and assertions dealing with token references defined in this section indicate whether the  
2490 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and  
2491 recipient MAY send a fault if such references are encountered.

2492

2493 Note: This approach is chosen because:

2494 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used  
2495 in a single reference.

2496 B) In a multi-message exchange, a token may be referenced using different mechanisms depending  
2497 on which of a series of messages is being secured.

2498

2499 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a  
2500 `wsse:InvalidSecurity` fault.

2501

### 2502 **WSS: SOAP Message Security 1.0 Properties**

#### 2503 **[Direct References]**

2504 This property indicates whether the initiator and recipient MUST be able to process direct token  
2505 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations  
2506 MUST be able to process such references.

2507

#### 2508 **[Key Identifier References]**

2509 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific  
2510 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2511 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST  
2512 NOT generate such references and that the initiator and recipient MAY send a fault if such references are  
2513 encountered. This property has a default value of 'false'.

2514

#### 2515 **[Issuer Serial References]**

2516 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2517 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST  
2518 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT  
2519 generate such references and that the initiator and recipient MAY send a fault if such references are  
2520 encountered. This property has a default value of 'false'.

2521

#### 2522 **[External URI References]**

2523 This boolean property indicates whether the initiator and recipient MUST be able to process references to  
2524 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST  
2525 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2526 generate such references and that the initiator and recipient MAY send a fault if such references are  
2527 encountered. This property has a default value of 'false'.

### 2528 **[Embedded Token References]**

2529 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2530 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to  
2531 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2532 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2533 This property has a default value of 'false'.

2534

## 2535 **WSS: SOAP Message Security 1.1 Properties**

### 2536 **[Thumbprint References]**

2537 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2538 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to  
2539 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2540 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2541 This property has a default value of 'false'.

2542

### 2543 **[EncryptedKey References]**

2544 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2545 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2546 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2547 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2548 This property has a default value of 'false'.

2549

### 2550 **[Signature Confirmation]**

2551 This boolean property specifies whether `wss11:SignatureConfirmation` elements should be used  
2552 as defined in WSS: Soap Message Security 1.1. If the value is 'true',  
2553 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If  
2554 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property  
2555 applies to all signatures that are included in the security header. This property has a default value of  
2556 'false'.

## 2557 **9.1 Wss10 Assertion**

2558 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are  
2559 supported.

### 2560 **Syntax**

```
2561 <sp:Wss10 xmlns:sp="..." ... >  
2562   <wsp:Policy xmlns:wsp="...">  
2563     <sp:MustSupportRefKeyIdentifier ... /> ?  
2564     <sp:MustSupportRefIssuerSerial ... /> ?  
2565     <sp:MustSupportRefExternalURI ... /> ?  
2566     <sp:MustSupportRefEmbeddedToken ... /> ?  
2567     ...  
2568   </wsp:Policy>  
2569   ...  
2570 </sp:Wss10>
```

2571

2572 The following describes the attributes and elements listed in the schema outlined above:



- 2573 /sp:Wss10  
 2574 This identifies a WSS10 assertion.
- 2575 /sp:Wss10/wsp:Policy  
 2576 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.
- 2577 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2578 This optional element is a policy assertion indicates that the [Key Identifier References] property  
 2579 is set to 'true'.
- 2580 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2581 This optional element is a policy assertion indicates that the [Issuer Serial References] property is  
 2582 set to 'true'.
- 2583 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI  
 2584 This optional element is a policy assertion indicates that the [External URI References] property is  
 2585 set to 'true'.
- 2586 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken  
 2587 This optional element is a policy assertion indicates that the [Embedded Token References]  
 2588 property is set to 'true'.

## 2589 9.2 Wss11 Assertion

2590 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are  
 2591 supported.

### 2592 Syntax

```

2593 <sp:Wss11 xmlns:sp="..." ... >
2594   <wsp:Policy xmlns:wsp="...">
2595     <sp:MustSupportRefKeyIdentifier ... /> ?
2596     <sp:MustSupportRefIssuerSerial ... /> ?
2597     <sp:MustSupportRefExternalURI ... /> ?
2598     <sp:MustSupportRefEmbeddedToken ... /> ?
2599     <sp:MustSupportRefThumbprint ... /> ?
2600     <sp:MustSupportRefEncryptedKey ... /> ?
2601     <sp:RequireSignatureConfirmation ... /> ?
2602     ...
2603   </wsp:Policy>
2604 </sp:Wss11>
  
```

2605  
 2606 The following describes the attributes and elements listed in the schema outlined above:

- 2607 /sp:Wss11  
 2608 This identifies an WSS11 assertion.
- 2609 /sp:Wss11/wsp:Policy  
 2610 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.
- 2611 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2612 This optional element is a policy assertion indicates that the [Key Identifier References] property  
 2613 is set to 'true'.
- 2614 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2615 This optional element is a policy assertion indicates that the [Issuer Serial References] property is  
 2616 set to 'true'.

- 2617 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI  
2618 This optional element is a policy assertion indicates that the [External URI References] property is  
2619 set to 'true'.
- 2620 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken  
2621 This optional element is a policy assertion indicates that the [Embedded Token References]  
2622 property is set to 'true'.
- 2623 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint  
2624 This optional element is a policy assertion indicates that the [Thumbprint References] property is  
2625 set to 'true'.
- 2626 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey  
2627 This optional element is a policy assertion indicates that the [EncryptedKey References] property  
2628 is set to 'true'.
- 2629 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation  
2630 This optional element is a policy assertion indicates that the [Signature Confirmation] property is  
2631 set to 'true'.

---

## 2632 10 WS-Trust Options

2633 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically  
2634 with client and server challenges and entropy behaviors. These assertions relate to interactions with a  
2635 Security Token Service and may augment the behaviors defined by the Binding Property Assertions  
2636 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2637

### 2638 **WS-Trust 1.3 Properties**

#### 2639 **[Client Challenge]**

2640 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a  
2641 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of  
2642 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of  
2643 messages exchanged by the client and service in satisfying the RST. This property has a default value of  
2644 'false'.

2645

#### 2646 **[Server Challenge]**

2647 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a  
2648 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of  
2649 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server may  
2650 increase the number of messages exchanged by the client and service in order to accommodate the  
2651 `wst:SignChallengeResponse` element sent by the client to the server in response to the  
2652 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the  
2653 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2654

#### 2655 **[Client Entropy]**

2656 This boolean property indicates whether client entropy is required to be used as key material for a  
2657 requested proof token. A value of 'true' indicates that client entropy is required. A value of 'false' indicates  
2658 that client entropy is not required. This property has a default value of 'false'.

2659

#### 2660 **[Server Entropy]**

2661 This boolean property indicates whether server entropy is required to be used as key material for a  
2662 requested proof token. A value of 'true' indicates that server entropy is required. A value of 'false'  
2663 indicates that server entropy is not required. This property has a default value of 'false'.

2664 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy  
2665 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm  
2666 Suite] property.

2667

#### 2668 **[Issued Tokens]**

2669 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in  
2670 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'  
2671 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of  
2672 'false'.

#### 2673 **[Collection]**

2674 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A  
2675 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and  
2676 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that  
2677 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default  
2678 value of 'false'.  
2679

## 2680 10.1 Trust13 Assertion

2681 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.

### 2682 Syntax

```
2683 <sp:Trust13 xmlns:sp="..." ... >  
2684   <wsp:Policy xmlns:wsp="...">  
2685     <sp:MustSupportClientChallenge ... />?  
2686     <sp:MustSupportServerChallenge ... />?  
2687     <sp:RequireClientEntropy ... />?  
2688     <sp:RequireServerEntropy ... />?  
2689     <sp:MustSupportIssuedTokens ... />?  
2690     <sp:RequireRequestSecurityTokenCollection />?  
2691     <sp:RequireAppliesTo />?  
2692     ...  
2693   </wsp:Policy>  
2694   ...  
2695 </sp:Trust13 ... >
```

2696  
2697 The following describes the attributes and elements listed in the schema outlined above:

2698 /sp:Trust13

2699       This identifies a Trust13 assertion.

2700 /sp:Trust13/wsp:Policy

2701       This indicates a policy that controls WS-Trust 1.3 options.

2702 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge

2703       This optional element is a policy assertion indicates that the [Client Challenge] property is set to  
2704 'true'.

2705 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge

2706       This optional element is a policy assertion indicates that the [Server Challenge] property is set to  
2707 'true'.

2708 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy

2709       This optional element is a policy assertion indicates that the [Client Entropy] property is set to  
2710 'true'.

2711 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy

2712       This optional element is a policy assertion indicates that the [Server Entropy] property is set to  
2713 'true'.

2714 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens

2715       This optional element is a policy assertion indicates that the [Issued Tokens] property is set to  
2716 'true'.

2717 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2718       This optional element is a policy assertion that indicates that the [Collection] property is set to  
2719 'true'.

2720 /sp:Trust10/wsp:Policy/sp:RequireAppliesTo

2721           This optional element is a policy assertion that indicates that the STS requires the requestor to  
2722           specify the scope for the issued token using wsp:AppliesTo in the RST.

---

## 2723 11 Guidance on creating new assertions and assertion 2724 extensibility

2725 This non-normative appendix provides guidance for designers of new assertions intended for use with this  
2726 specification.

### 2727 11.1 General Design Points

- 2728 • Prefer Distinct QNames
- 2729 • Parameterize using nested policy where possible.
- 2730 • Parameterize using attributes and/or child elements where necessary.

### 2731 11.2 Detailed Design Guidance

2732 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per  
2733 WS-Policy is performed by matching element QNames. Matching does not take into account attributes  
2734 that are present on the assertion element. Nor does it take into account child elements except for  
2735 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions  
2736 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2737  
2738 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken  
2739 into account. In general, multiple assertions with distinct QNames are preferably to a single assertion that  
2740 uses attributes and/or content to distinguish different cases. For example, given two possible assertion  
2741 designs;

```
2742  
2743 Design 1  
2744  
2745 <A1/>  
2746 <A2/>  
2747 <A3/>  
2748  
2749 Design 2.  
2750  
2751 <A Parameter='1' />  
2752 <A Parameter='2' />  
2753 <A Parameter='3' />  
2754
```

2755 then design 1. would generally be preferred because it allows the policy matching logic to provide more  
2756 accurate matches between policies.

2757  
2758 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct  
2759 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These  
2760 distinct token assertions make policy matching much more useful as less false positives are generated  
2761 when performing policy matching.

2762  
2763 There are cases where using attributes or child elements as parameters in assertion design is  
2764 reasonable. Examples include cases when implementations are expected to understand all the values for  
2765 a given parameter and when encoding the parameter information into the assertion QName would result  
2766 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2767 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken  
2768 attribute and implementations are expected to understand the meaning of all 5 values. If this information  
2769 was encoded into the assertion QNames, each existing token assertion would require five variants, one  
2770 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2771

2772 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For  
2773 example, the token version assertions defined in Section 5 use such an approach. The overall token type  
2774 assertion is parameterized by the nested token version assertions. Policy matching can use these  
2775 parameters to find matches between policies where the broad token type is support by both parties but  
2776 they might not support the same specific versions.

2777

2778 Note, when designing assertions for new token types such assertions SHOULD allow the  
2779 sp:IncludeToken attribute and SHOULD allow nested policy.

2780

---

## 2781 **12 Security Considerations**

2782 It is strongly recommended that policies and assertions be signed to prevent tampering.

2783 It is recommended that policies should not be accepted unless they are signed and have an associated  
2784 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely  
2785 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that  
2786 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2787  
2788 It should be noted that the mechanisms described in this document could be secured as part of a SOAP  
2789 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using  
2790 object-specific security mechanisms.

2791  
2792 It is recommended that policies not specify two (or more) SignedSupportingTokens or  
2793 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are  
2794 subject to modification which may be undetectable.

2795  
2796 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest  
2797 of the policy in order to combat certain XML substitution attacks.



---

## 2798 **A. Assertions and WS-PolicyAttachment**

2799 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per  
2800 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical  
2801 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any  
2802 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy  
2803 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

### 2804 **A.1 Endpoint Policy Subject Assertions**

#### 2805 **A.1.1 Security Binding Assertions**

2806 [TransportBinding Assertion](#) (Section 7.3)

2807 [SymmetricBinding Assertion](#) (Section 7.4)

2808 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2809 **A.1.2 Token Assertions**

2810 [SupportingTokens Assertion](#) (Section 8.1)

2811 [SignedSupportingTokens Assertion](#) (Section 8.2)

2812 [EndorsingSupportingTokens Assertion](#) (Section 8.3)

2813 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)

2814 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)

2815 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)

2816 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

#### 2817 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

2818 [Wss10 Assertion](#) (Section 9.1)

#### 2819 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

2820 [Wss11 Assertion](#) (Section 9.2)

#### 2821 **A.1.5 Trust 1.0 Assertions**

2822 [Trust13 Assertion](#) (Section 10.1)

### 2823 **A.2 Operation Policy Subject Assertions**

#### 2824 **A.2.1 Security Binding Assertions**

2825 [SymmetricBinding Assertion](#) (Section 7.4)

2826 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2827 **A.2.2 Supporting Token Assertions**

2828 [SupportingTokens Assertion](#) (Section 8.1)

2829 [SignedSupportingTokens Assertion](#) (Section 8.2)

2830	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2831	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2832	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2833	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2834	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

## 2835 **A.3 Message Policy Subject Assertions**

### 2836 **A.3.1 Supporting Token Assertions**

2837	<a href="#">SupportingTokens Assertion</a>	(Section 8.1)
2838	<a href="#">SignedSupportingTokens Assertion</a>	(Section 8.2)
2839	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2840	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2841	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2842	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2843	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

### 2844 **A.3.2 Protection Assertions**

2845	<a href="#">SignedParts Assertion</a>	(Section 4.1.1)
2846	<a href="#">SignedElements Assertion</a>	(Section 4.1.2)
2847	<a href="#">EncryptedParts Assertion</a>	(Section 4.2.1)
2848	<a href="#">EncryptedElements Assertion</a>	(Section 4.2.2)
2849	<a href="#">ContentEncryptedElements Assertion</a>	(Section 4.2.3)
2850	<a href="#">RequiredElements Assertion</a>	(Section 4.3.1)
2851	<a href="#">RequiredParts Assertion</a>	(Section 4.3.2)

## 2852 **A.4 Assertions With Undefined Policy Subject**

2853 The assertions listed in this section do not have a defined policy subject because they appear nested  
 2854 inside some other assertion which does have a defined policy subject. This list is derived from nested  
 2855 assertions in the specification that have independent sections. It is not a complete list of nested  
 2856 assertions. Many of the assertions previously listed in this appendix as well as the ones below have  
 2857 additional nested assertions.

### 2858 **A.4.1 General Assertions**

2859	<a href="#">AlgorithmSuite Assertion</a>	(Section 7.1)
2860	<a href="#">Layout Assertion</a>	(Section 7.2)

### 2861 **A.4.2 Token Usage Assertions**

2862 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)  
 2863 assertions.

### 2864 **A.4.3 Token Assertions**

2865	<a href="#">UsernameToken Assertion</a>	(Section 5.3.1)
------	---	-----------------

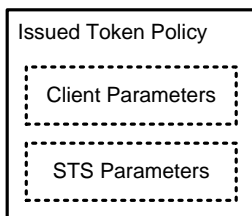
2866	<a href="#">IssuedToken Assertion</a>	(Section 5.3.2)
2867	<a href="#">X509Token Assertion</a>	(Section 5.3.3)
2868	<a href="#">KerberosToken Assertion</a>	(Section 5.3.4)
2869	<a href="#">SpnegoContextToken Assertion</a>	(Section 5.3.5)
2870	<a href="#">SecurityContextToken Assertion</a>	(Section 5.3.6)
2871	<a href="#">SecureConversationToken Assertion</a>	(Section 5.3.7)
2872	<a href="#">SamlToken Assertion</a>	(Section 5.3.8)
2873	<a href="#">RelToken Assertion</a>	(Section 5.3.9)
2874	<a href="#">HttpsToken Assertion</a>	(Section 5.3.10)

2875 **B. Issued Token Policy**

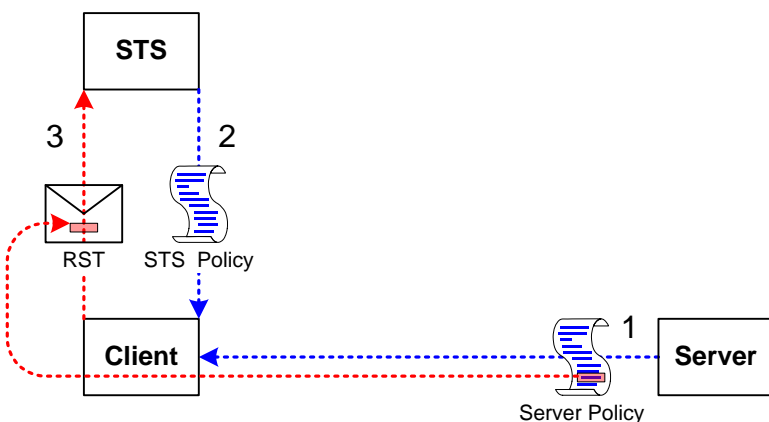
2876 The section provides further detail about behavior associated with the IssuedToken assertion in section  
2877 5.3.2.

2878  
2879 The issued token security model involves a three-party setup. There's a target Server, a Client, and a  
2880 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from  
2881 STS to Client. Policy may be embedded inside an Issued Token assertion, or acquired out-of-band. There  
2882 may be an explicit trust relationship between the Server and the STS. There must be a trust relationship  
2883 between the Client and the STS.

2884  
2885 The Issued Token policy assertion includes two parts: 1) client-specific parameters that must be  
2886 understood and processed by the client and 2) STS specific parameters which are to be processed by the  
2887 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



2888  
2889 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server  
2890 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are  
2891 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the  
2892 RST request sent by the Client to the STS as illustrated in the figure below.



2894  
2895 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to  
2896 formulate the RST request and will include any security-specific requirements of the STS.

2897  
2898 The Client may augment or replace the contents of the RST made to the STS based on the Client-specific  
2899 parameters received from the Issued Token policy assertion contained in the Server policy, from policy it  
2900 received for the STS, or any other local parameters.

2901

2902 The Issued Token Policy Assertion contains elements which must be understood by the Client. The  
2903 assertion contains one element which contains a list of arbitrary elements which should be sent along to  
2904 the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST  
2905 request sent by the Client to the STS following the protocol defined in WS-Trust.

2906

2907 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [[WS-](#)  
2908 [Trust](#)]. All items are optional, since the Server and STS may already have a pre-arranged relationship  
2909 which specifies some or all of the conditions and constraints for issued tokens.

2910

## C. Strict Security Header Layout Examples

2911 The following sections describe the security header layout for specific bindings when applying the 'Strict'  
2912 layout rules defined in Section 6.7.

### 2913 C.1 Transport Binding

2914 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

#### 2915 C.1.1 Policy

2916 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport  
2917 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username  
2918 token attached to the message, and finally an X509 token attached to the message and endorsing the  
2919 message signature. No message protection requirements are described since the transport covers all  
2920 message parts.

```
2921 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2922   <sp:TransportBinding>
2923     <wsp:Policy>
2924       <sp:TransportToken>
2925         <wsp:Policy>
2926           <sp:HttpsToken />
2927         </wsp:Policy>
2928       </sp:TransportToken>
2929       <sp:AlgorithmSuite>
2930         <wsp:Policy>
2931           <sp:Basic256 />
2932         </wsp:Policy>
2933       </sp:AlgorithmSuite>
2934       <sp:Layout>
2935         <wsp:Policy>
2936           <sp:Strict />
2937         </wsp:Policy>
2938       </sp:Layout>
2939       <sp:IncludeTimestamp />
2940     </wsp:Policy>
2941   </sp:TransportBinding>
2942   <sp:SignedSupportingTokens>
2943     <wsp:Policy>
2944       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2945     </wsp:Policy>
2946   </sp:SignedSupportingTokens>
2947   <sp:SignedEndorsingSupportingTokens>
2948     <wsp:Policy>
2949       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
2950         <wsp:Policy>
2951           <sp:WssX509v3Token10 />
2952         </wsp:Policy>
2953       </sp:X509Token>
2954     </wsp:Policy>
2955   </sp:SignedEndorsingSupportingTokens>
2956   <sp:Wss11>
2957     <sp:RequireSignatureConfirmation />
2958   </sp:Wss11>
2959 </wsp:Policy>
```

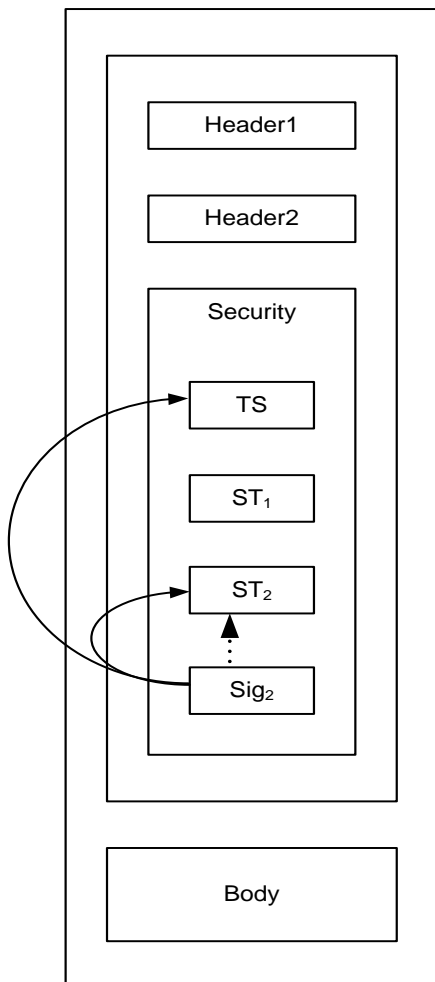
2960 This policy is used as the basis for the examples shown in the subsequent section describing the security  
2961 header layout for this binding.

2962 **C.1.2 Initiator to Recipient Messages**

2963 Messages sent from initiator to recipient have the following layout for the security header:

- 2964 1. A `wsu:Timestamp` element.
- 2965 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 2966 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the  
2967 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1  
2968 above and **SHOULD** cover any other unique identifier for the message in order to prevent  
2969 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If  
2970 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a  
2971 Derived Key Token, based on the supporting token, appears between the supporting token and  
2972 the signature.
- 2973 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each  
2974 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least  
2975 some other unique identifier for the message in order to prevent replays. If [Token Protection] is  
2976 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the  
2977 supporting token is associated with a symmetric key, then a Derived Key Token, based on the  
2978 supporting token, appears before the signature.

2979 The following diagram illustrates the security header layout for the initiator to recipient message:



2980

2981 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The  
 2982 arrows on the left from the box labeled Sig<sub>2</sub> indicate the parts signed by the supporting token labeled ST<sub>2</sub>,  
 2983 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST<sub>2</sub>.  
 2984 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering  
 2985 of the items in the security header follows the most optimal layout for a receiver to process its contents.

2986 *Example:*

2987 Initiator to recipient message

```

2988 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsp="..." xmlns:ds="...">
2989   <S:Header>
2990     ...
2991     <wsse:Security>
2992       <wsu:Timestamp wsu:Id="timestamp">
2993         <wsu:Created>[datetime]</wsu:Created>
2994         <wsu:Expires>[datetime]</wsu:Expires>
2995       </wsu:Timestamp>
2996       <wsse:UsernameToken wsu:Id='SomeSignedToken' >
2997         ...
2998       </wsse:UsernameToken>
2999       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
3000         ...
3001       </wsse:BinarySecurityToken>
3002       <ds:Signature>
3003         <ds:SignedInfo>
3004           <ds:References>
3005             <ds:Reference URI="#timestamp" />
3006             <ds:Reference URI="#SomeSignedEndorsingToken" />
3007           </ds:References>
3008         </ds:SignedInfo>
3009         <ds:SignatureValue>...</ds:SignatureValue>
3010         <ds:KeyInfo>
3011           <wsse:SecurityTokenReference>
3012             <wsse:Reference URI="#SomeSignedEndorsingToken" />
3013           </wsse:SecurityTokenReference>
3014         </ds:KeyInfo>
3015       </ds:Signature>
3016     ...
3017   </wsse:Security>
3018   ...
3019 </S:Header>
3020 <S:Body>
3021   ...
3022 </S:Body>
3023 </S:Envelope>
  
```

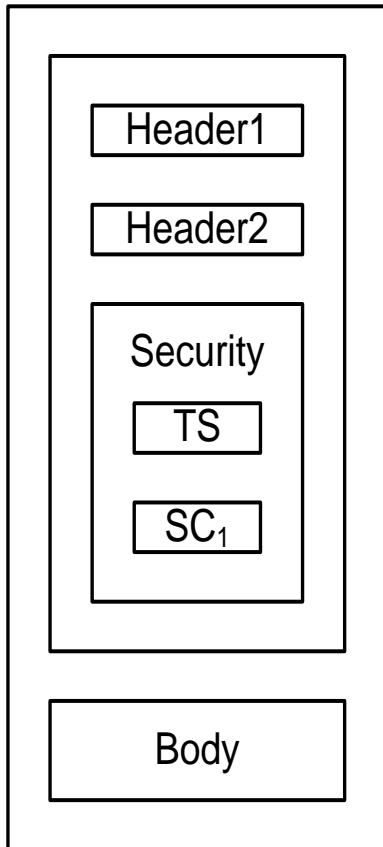
### 3024 C.1.3 Recipient to Initiator Messages

3025 Messages sent from recipient to initiator have the following layout for the security header:

- 3026 1. A `wsu:Timestamp` element.
- 3027 2. If the [Signature Confirmation] property has a value of 'true', then a  
 3028 `wsse11:SignatureConfirmation` element for each signature in the corresponding message  
 3029 sent from initiator to recipient. If there are no signatures in the corresponding message from the  
 3030 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no `Value`  
 3031 attribute.

3032 The following diagram illustrates the security header layout for the recipient to initiator message:





3033

3034 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One  
 3035 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial  
 3036 message illustrated previously is included. In general, the ordering of the items in the security header  
 3037 follows the most optimal layout for a receiver to process its contents.

3038 *Example:*

3039 Recipient to initiator message

```

3040 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3041   <S:Header>
3042     ...
3043     <wsse:Security>
3044       <wsu:Timestamp wsu:Id="timestamp">
3045         <wsu:Created>[datetime]</wsu:Created>
3046         <wsu:Expires>[datetime]</wsu:Expires>
3047       </wsu:Timestamp>
3048       <wsse11:SignatureConfirmation Value="..." />
3049       ...
3050     </wsse:Security>
3051     ...
3052   </S:Header>
3053   <S:Body>
3054     ...
3055   </S:Body>
3056 </S:Envelope>
  
```

## 3057 C.2 Symmetric Binding

3058 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

## 3059 C.2.1 Policy

3060 The following example shows a policy indicating a Symmetric Binding, a symmetric key based  
3061 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message  
3062 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in  
3063 the message signature and the supporting signatures, a username token attached to the message, and  
3064 finally an X509 token attached to the message and endorsing the message signature. Minimum message  
3065 protection requirements are described as well.

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

```

3115
3116 <!-- Example Message Policy -->
3117 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3118   <sp:SignedParts>
3119     <sp:Header Name="Header1" Namespace="..." />
3120     <sp:Header Name="Header2" Namespace="..." />
3121     <sp:Body/>
3122   </sp:SignedParts>
3123   <sp:EncryptedParts>
3124     <sp:Header Name="Header2" Namespace="..." />
3125     <sp:Body/>
3126   </sp:EncryptedParts>
3127 </wsp:Policy>

```

3128 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3129 header layout for this binding.

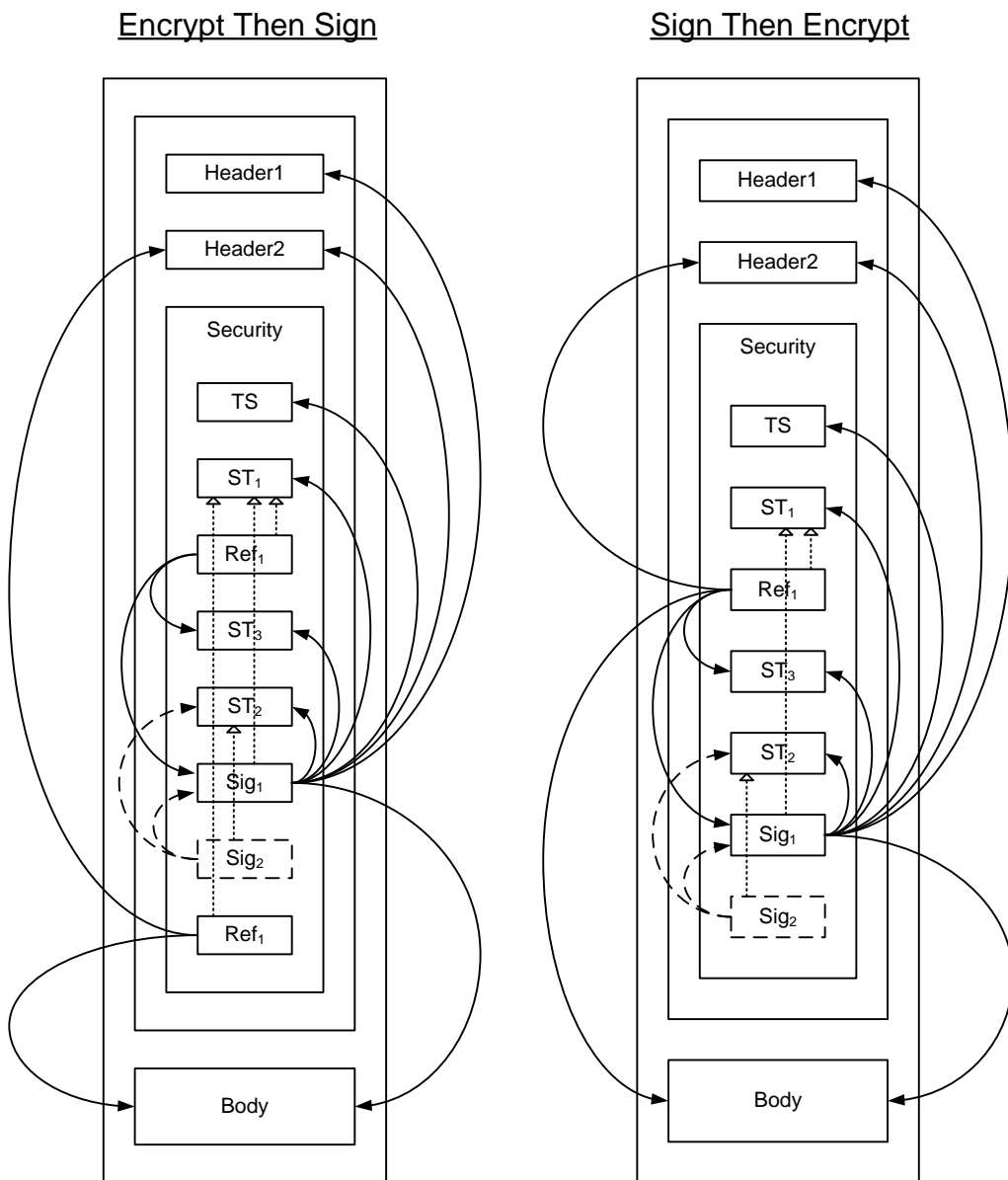
## 3130 C.2.2 Initiator to Recipient Messages

3131 Messages sent from initiator to recipient have the following layout for the security header:

- 3132 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3133 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or  
3134 `.../IncludeToken/Always`, then the [Encryption Token].
- 3135 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3136 Derived Key Token is used for encryption.
- 3137 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3138 reference list MUST include a reference to the message signature. If [Protection Order] is  
3139 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3140 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3141 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3142 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]  
3143 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or  
3144 `.../IncludeToken/Always`.
- 3145 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3146 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the  
3147 [Signature Token].
- 3148 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This  
3149 Derived Key Token is used for signature.
- 3150 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of  
3151 whether they are included in the message, and any message parts specified in SignedParts  
3152 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature  
3153 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in  
3154 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3155 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing  
3156 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]  
3157 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the  
3158 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the  
3159 endorsing token, appears before the signature.
- 3160 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message  
3161 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
3162 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3163 above.

3164

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



3166

3167 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
3168 The dashed arrows on the left from the box labeled Sig<sub>2</sub> indicate the parts signed by the supporting token  
3169 labeled ST<sub>2</sub>, namely the message signature labeled Sig<sub>1</sub> and the token used as the basis for the  
3170 signature labeled ST<sub>2</sub>. The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts  
3171 encrypted using a key based on the Shared Secret Token labeled ST<sub>1</sub>. The dotted arrows inside the box  
3172 labeled Security indicate the token that was used as the basis for each cryptographic operation. In  
3173 general, the ordering of the items in the security header follows the most optimal layout for a receiver to  
3174 process its contents.

3175 *Example:*

3176 Initiator to recipient message using EncryptBeforeSigning:

```
3177 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3178   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3179   xmlns:xenc="..." xmlns:ds="...">
3180 <S:Header>
3181   <x:Header1 wsu:Id="Header1" >
3182     ...
3183   </x:Header1>
3184
```

```

3185 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3186   <!-- Plaintext Header2
3187   <x:Header2 wsu:Id="Header2" >
3188     ...
3189   </x:Header2>
3190   -->
3191   ...
3192 </wsse1:EncryptedHeader>
3193 ...
3194 <wsse:Security>
3195   <wsu:Timestamp wsu:Id="Timestamp">
3196     <wsu:Created>...</wsu:Created>
3197     <wsu:Expires>...</wsu:Expires>
3198   </wsu:Timestamp>
3199   <saml:Assertion AssertionId="_SharedSecretToken" ...>
3200     ...
3201   </saml:Assertion>
3202   <xenc:ReferenceList>
3203     <xenc:DataReference URI="#enc_Signature" />
3204     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3205     ...
3206   </xenc:ReferenceList>
3207   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3208     <!-- Plaintext UsernameToken
3209     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3210       ...
3211     </wsse:UsernameToken>
3212     -->
3213     ...
3214     <ds:KeyInfo>
3215       <wsse:SecurityTokenReference>
3216         <wsse:Reference URI="#_SharedSecretToken" />
3217       </wsse:SecurityTokenReference>
3218     </ds:KeyInfo>
3219   </xenc:EncryptedData>
3220   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3221     ...
3222   </wsse:BinarySecurityToken>
3223   <xenc:EncryptedData ID="enc_Signature">
3224     <!-- Plaintext Signature
3225     <ds:Signature Id="Signature">
3226       <ds:SignedInfo>
3227         <ds:References>
3228           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3229           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3230           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3231           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3232           <ds:Reference URI="#Header1" >...</ds:Reference>
3233           <ds:Reference URI="#Header2" >...</ds:Reference>
3234           <ds:Reference URI="#Body" >...</ds:Reference>
3235         </ds:References>
3236       </ds:SignedInfo>
3237     </ds:SignatureValue>...</ds:SignatureValue>
3238     <ds:KeyInfo>
3239       <wsse:SecurityTokenReference>
3240         <wsse:Reference URI="#_SharedSecretToken" />
3241       </wsse:SecurityTokenReference>
3242     </ds:KeyInfo>
3243   </xenc:EncryptedData>
3244   -->
3245   ...
3246   <ds:KeyInfo>
3247     <wsse:SecurityTokenReference>
3248       <wsse:Reference URI="#_SharedSecretToken" />

```

```

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

```

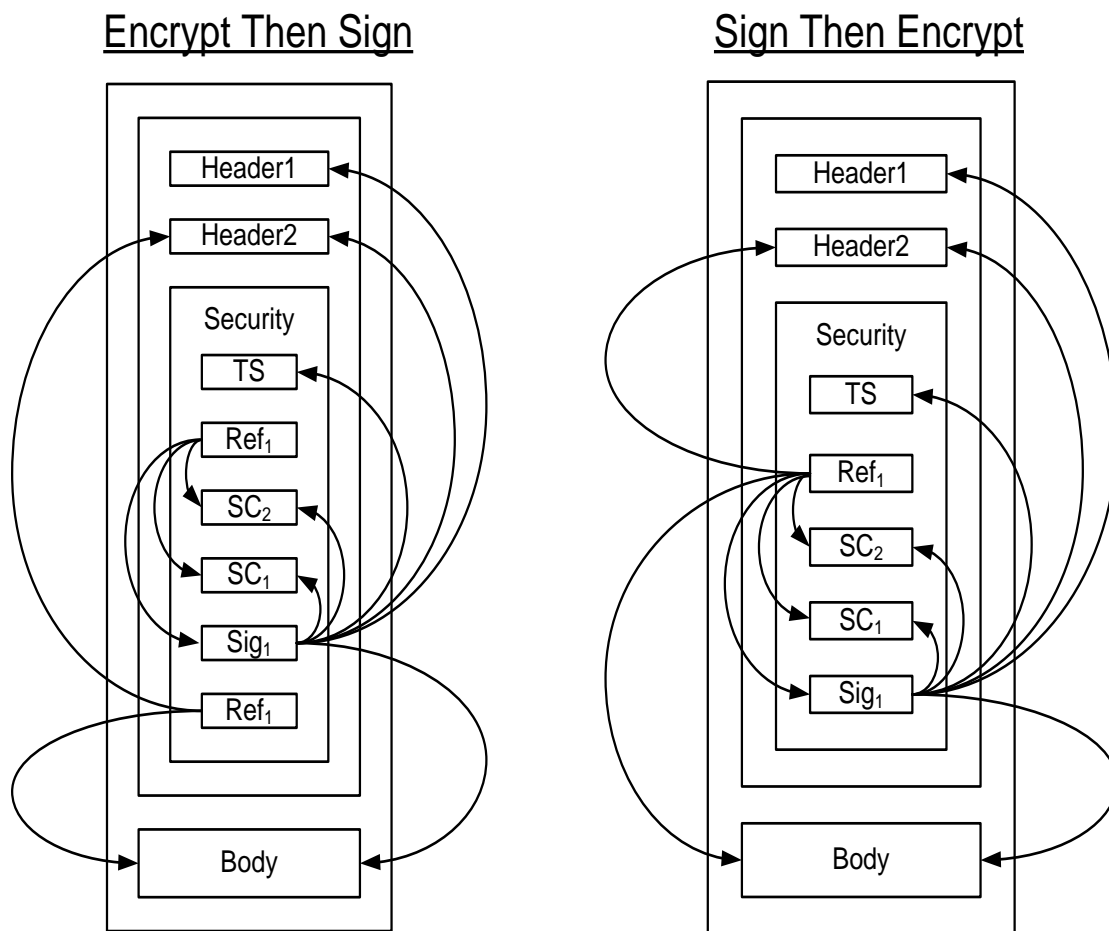
### 3284 C.2.3 Recipient to Initiator Messages

3285 Messages send from recipient to initiator have the following layout for the security header:

- 3286 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3287 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the  
3288 [Encryption Token].
- 3289 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3290 Derived Key Token is used for encryption.
- 3291 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3292 reference list MUST include a reference to the message signature from 6 below, and the  
3293 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is  
3294 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3295 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3296 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3297 above.
- 3298 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each  
3299 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3300 in the corresponding message from the initiator to the recipient, then a  
3301 `wss11:SignatureConfirmation` element with no Value attribute.
- 3302 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3303 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].

- 3304 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This  
 3305 Derived Key Token is used for signature.
- 3306 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation  
 3307 elements from 5 above, and all the message parts specified in SignedParts assertions in the  
 3308 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]  
 3309 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token  
 3310 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 3311 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message  
 3312 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
 3313 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption  
 3314 Token].

3315 The following diagram illustrates the security header layout for the recipient to initiator message:



3316

3317 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3318 The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts encrypted using a key based  
 3319 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two  
 3320 wssell:SignatureConfirmation elements labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures  
 3321 in the initial message illustrated previously is included. In general, the ordering of the items in the security  
 3322 header follows the most optimal layout for a receiver to process its contents. The rules used to determine  
 3323 this ordering are described in Appendix C.

3324 *Example:*



3325 Recipient to initiator message using EncryptBeforeSigning:

```
3326 <S:Envelope>
3327   <S:Header>
3328     <x:Header1 wsu:Id="Header1" >
3329       ...
3330     </x:Header1>
3331     <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3332       <!-- Plaintext Header2
3333       <x:Header2 wsu:Id="Header2" >
3334         ...
3335       </x:Header2>
3336       -->
3337       ...
3338     </wsse11:EncryptedHeader>
3339     ...
3340   <wsse:Security>
3341     <wsu:Timestamp wsu:Id="Timestamp">
3342       <wsu:Created>...</wsu:Created>
3343       <wsu:Expires>...</wsu:Expires>
3344     </wsu:Timestamp>
3345     <xenc:ReferenceList>
3346       <xenc:DataReference URI="#enc_Signature" />
3347       <xenc:DataReference URI="#enc_SigConf1" />
3348       <xenc:DataReference URI="#enc_SigConf2" />
3349       ...
3350     </xenc:ReferenceList>
3351     <xenc:EncryptedData ID="enc_SigConf1" >
3352       <!-- Plaintext SignatureConfirmation
3353       <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3354         ...
3355       </wsse11:SignatureConfirmation>
3356       -->
3357       ...
3358     </xenc:EncryptedData>
3359     <xenc:EncryptedData ID="enc_SigConf2" >
3360       <!-- Plaintext SignatureConfirmation
3361       <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3362         ...
3363       </wsse11:SignatureConfirmation>
3364       -->
3365       ...
3366     </xenc:EncryptedData>
```

```

3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
<xenc:EncryptedData Id="enc_Signature">
  <!-- Plaintext Signature
  <ds:Signature Id="Signature">
    <ds:SignedInfo>
      <ds:References>
        <ds:Reference URI="#Timestamp" >...</ds:Reference>
        <ds:Reference URI="#SigConf1" >...</ds:Reference>
        <ds:Reference URI="#SigConf2" >...</ds:Reference>
        <ds:Reference URI="#Header1" >...</ds:Reference>
        <ds:Reference URI="#Header2" >...</ds:Reference>
        <ds:Reference URI="#Body" >...</ds:Reference>
      </ds:References>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
-->
</xenc:EncryptedData>
...
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#_SomeIssuedToken" />
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:EncryptedData>
<xenc:ReferenceList>
  <xenc:DataReference URI="#enc_Body" />
  <xenc:DataReference URI="#enc_Header2" />
  ...
</xenc:ReferenceList>
</xenc:EncryptedData>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="Body">
  <xenc:EncryptedData Id="enc_Body">
    ...
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </xenc:EncryptedData>
</S:Body>
</S:Envelope>

```

## 3416 C.3 Asymmetric Binding

3417 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

### 3418 C.3.1 Policy

3419 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator  
3420 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the  
3421 message parts before signing, a requirement to encrypt the message signature, a requirement to include  
3422 tokens in the message signature and the supporting signatures, a requirement to include  
3423 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3424 an X509 token attached to the message and endorsing the message signature. Minimum message  
3425 protection requirements are described as well.

```
3426 <!-- Example Endpoint Policy -->  
3427 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
3428   <sp:AsymmetricBinding>  
3429     <wsp:Policy>  
3430       <sp:RecipientToken>  
3431         <wsp:Policy>  
3432           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3433         </wsp:Policy>  
3434       </sp:RecipientToken>  
3435       <sp:InitiatorToken>  
3436         <wsp:Policy>  
3437           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3438         </wsp:Policy>  
3439       </sp:InitiatorToken>  
3440       <sp:AlgorithmSuite>  
3441         <wsp:Policy>  
3442           <sp:Basic256 />  
3443         </wsp:Policy>  
3444       </sp:AlgorithmSuite>  
3445       <sp:Layout>  
3446         <wsp:Policy>  
3447           <sp:Strict />  
3448         </wsp:Policy>  
3449       </sp:Layout>  
3450       <sp:IncludeTimestamp />  
3451       <sp:EncryptBeforeSigning />  
3452       <sp:EncryptSignature />  
3453       <sp:ProtectTokens />  
3454     </wsp:Policy>  
3455   </sp:AsymmetricBinding>  
3456   <sp:SignedEncryptedSupportingTokens>  
3457     <wsp:Policy>  
3458       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />  
3459     </wsp:Policy>  
3460   </sp:SignedEncryptedSupportingTokens>  
3461   <sp:SignedEndorsingSupportingTokens>  
3462     <wsp:Policy>  
3463       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">  
3464         <wsp:Policy>  
3465           <sp:WssX509v3Token10 />  
3466         </wsp:Policy>  
3467       </sp:X509Token>  
3468     </wsp:Policy>  
3469   </sp:SignedEndorsingSupportingTokens>  
3470   <sp:Wss11>  
3471     <wsp:Policy>  
3472       <sp:RequireSignatureConfirmation />  
3473     </wsp:Policy>  
3474   </sp:Wss11>  
3475 </wsp:Policy>  
3476
```

3477

```
3478 <!-- Example Message Policy -->
3479 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3480   <sp:SignedParts>
3481     <sp:Header Name="Header1" Namespace="..." />
3482     <sp:Header Name="Header2" Namespace="..." />
3483     <sp:Body/>
3484   </sp:SignedParts>
3485   <sp:EncryptedParts>
3486     <sp:Header Name="Header2" Namespace="..." />
3487     <sp:Body/>
3488   </sp:EncryptedParts>
3489 </wsp:All>
```

3490  
3491 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3492 header layout for this binding.

### 3493 **C.3.2 Initiator to Recipient Messages**

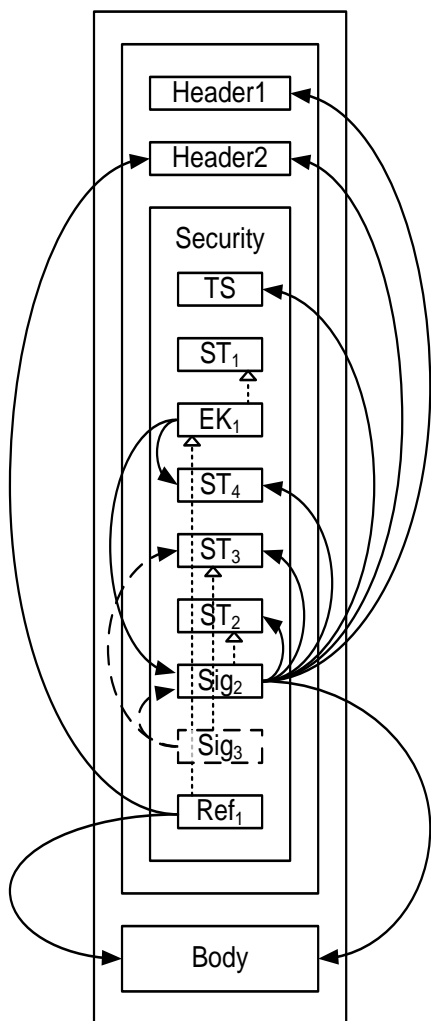
3494 Messages sent from initiator to recipient have the following layout:

- 3495 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3496 2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3497 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
- 3498 3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3499 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3500 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3501 reference to all the message parts specified in EncryptedParts assertions in the policy. If  
3502 [Signature Protection] is 'true' then the reference list MUST contain a reference to the message  
3503 signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message  
3504 signature.
- 3505 4. Any tokens from the supporting tokens properties (as defined in section 8) whose  
3506 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3507 5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3508 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
- 3509 6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from  
3510 1 above, any tokens from 4 above regardless of whether they are included in the message, and  
3511 any message parts specified in SignedParts assertions in the policy. If [Token Protection] is 'true',  
3512 the signature MUST also cover the [Initiator Token] regardless of whether it is included in the  
3513 message.
- 3514 7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting  
3515 Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a  
3516 symmetric key, then a Derived Key Token, based on the supporting token, appears before the  
3517 signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token  
3518 regardless of whether it is included in the message.
- 3519 8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
3520 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
3521 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
3522 reference list includes a reference to all the message parts specified in EncryptedParts assertions  
3523 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
3524 element from 3 above.

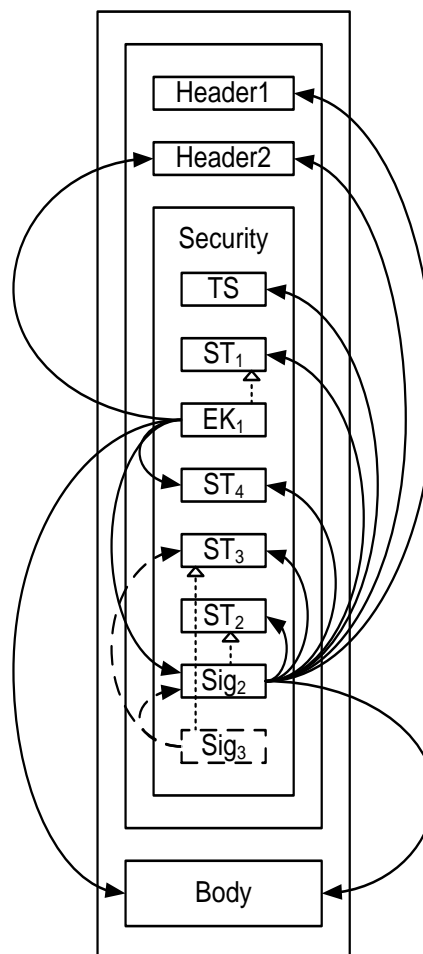
3525

3526 The following diagram illustrates the security header layout for the initiator to recipient messages:

### Encrypt Then Sign



### Sign Then Encrypt



3527

3528 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3529 using the [Initiator Token] labeled ST<sub>2</sub>. The dashed arrows on the left from the box labeled Sig<sub>3</sub> indicate  
 3530 the parts signed by the supporting token ST<sub>3</sub>, namely the message signature Sig<sub>2</sub> and the token used as  
 3531 the basis for the signature labeled ST<sub>3</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate references  
 3532 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on the left  
 3533 from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained in the  
 3534 encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token used as  
 3535 the basis for each cryptographic operation. In general, the ordering of the items in the security header  
 3536 follows the most optimal layout for a receiver to process its contents. The rules used to determine this  
 3537 ordering are described in Appendix C.

3538

3539 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted  
 3540 key contains an external reference to the token containing the encryption key. The diagram illustrates  
 3541 how one might attach a security token related to the encrypted key for completeness. One possible use-

3542 case for this approach might be a stack which does not support the STR Dereferencing Transform, but  
3543 wishes to include the encryption token in the message signature.

3544 Initiator to recipient message *Example*

3545 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3546   xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3547   <S:Header>
3548     <x:Header1 wsu:Id="Header1" >
3549     ...
3550   </x:Header1>
3551   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3552     <!-- Plaintext Header2
3553     <x:Header2 wsu:Id="Header2" >
3554     ...
3555     </x:Header2>
3556     -->
3557     ...
3558   </wssell1:EncryptedHeader>
3559   ...
3560   <wsse:Security>
3561     <wsu:Timestamp wsu:Id="Timestamp">
3562       <wsu:Created>...</wsu:Created>
3563       <wsu:Expires>...</wsu:Expires>
3564     </wsu:Timestamp>
3565     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3566     ...
3567   </wsse:BinarySecurityToken>
3568   <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3569     ...
3570     <xenc:ReferenceList>
3571       <xenc:DataReference URI="#enc_Signature" />
3572       <xenc:DataReference URI="#enc_SomeUsernameToken" />
3573       ...
3574     </xenc:ReferenceList>
3575   </xenc:EncryptedKey>
3576   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3577     <!-- Plaintext UsernameToken
3578     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3579     ...
3580     </wsse:UsernameToken>
3581     -->
3582     ...
3583   </xenc:EncryptedData>
3584   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3585   ...
3586   </wsse:BinarySecurityToken>
3587   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3588   ...
3589   </wsse:BinarySecurityToken>
3590   <xenc:EncryptedData ID="enc_Signature">
3591     <!-- Plaintext Signature
3592     <ds:Signature Id="Signature">
3593       <ds:SignedInfo>
3594         <ds:References>
3595           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3596           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3597           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3598           <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3599           <ds:Reference URI="#Header1" >...</ds:Reference>
3600           <ds:Reference URI="#Header2" >...</ds:Reference>
3601           <ds:Reference URI="#Body" >...</ds:Reference>
3602         </ds:References>
3603       </ds:SignedInfo>
3604     <ds:SignatureValue>...</ds:SignatureValue>
3605     <ds:KeyInfo>
3606       <wsse:SecurityTokenReference>
3607         <wsse:Reference URI="#InitiatorToken" />
3608       </wsse:SecurityTokenReference>
3609     </ds:KeyInfo>

```

```

3610     </ds:Signature>
3611     -->
3612     ...
3613 </xenc:EncryptedData>
3614 <ds:Signature>
3615   <ds:SignedInfo>
3616     <ds:References>
3617       <ds:Reference URI="#Signature" >...</ds:Reference>
3618       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3619     </ds:References>
3620   </ds:SignedInfo>
3621   <ds:SignatureValue>...</ds:SignatureValue>
3622   <ds:KeyInfo>
3623     <wsse:SecurityTokenReference>
3624       <wsse:Reference URI="#SomeSupportingToken" />
3625     </wsse:SecurityTokenReference>
3626   </ds:KeyInfo>
3627 </ds:Signature>
3628 <xenc:ReferenceList>
3629   <xenc:DataReference URI="#enc_Body" />
3630   <xenc:DataReference URI="#enc_Header2" />
3631   ...
3632 </xenc:ReferenceList>
3633 </wsse:Security>
3634 </S:Header>
3635 <S:Body wsu:Id="Body">
3636   <xenc:EncryptedData Id="enc_Body">
3637     ...
3638     <ds:KeyInfo>
3639       <wsse:SecurityTokenReference>
3640         <wsse:Reference URI="#RecipientEncryptedKey" />
3641       </wsse:SecurityTokenReference>
3642     </ds:KeyInfo>
3643   </xenc:EncryptedData>
3644 </S:Body>
3645 </S:Envelope>

```

### 3646 C.3.3 Recipient to Initiator Messages

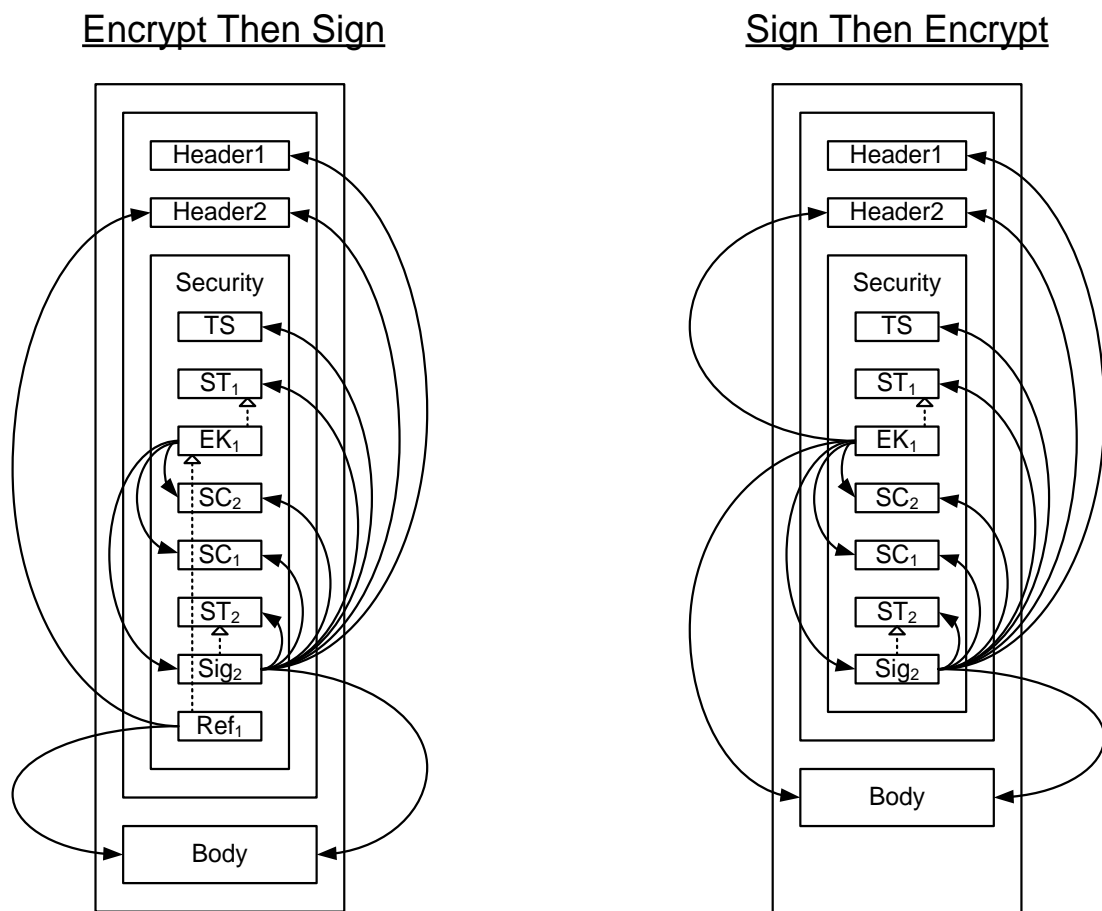
3647 Messages sent from recipient to initiator have the following layout:

- 3648 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3649 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3650 `.../IncludeToken/Always`, then the [Initiator Token].
- 3651 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3652 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3653 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3654 reference to all the message parts specified in EncryptedParts assertions in the policy. If  
3655 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the  
3656 message signature from 6 below, if any and references to the  
3657 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3658 4. If [Signature Confirmation] is 'true', then a `wss11:SignatureConfirmation` element for each  
3659 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3660 in the corresponding message from the initiator to the recipient, then a  
3661 `wss11:SignatureConfirmation` element with no Value attribute.
- 3662 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3663 `.../IncludeToken/Always`, then the [Recipient Token].



- 3664 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],  
 3665 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements  
 3666 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token  
 3667 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3668 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
 3669 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
 3670 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
 3671 reference list includes a reference to all the message parts specified in EncryptedParts assertions  
 3672 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
 3673 element from 3 above.

3674  
 3675 The following diagram illustrates the security header layout for the recipient to initiator messages:



3676  
 3677 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3678 using the [Recipient Token] labeled ST<sub>2</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate  
 3679 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on  
 3680 the left from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained  
 3681 in the encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token  
 3682 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements  
 3683 labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures in the initial message illustrated previously is  
 3684 included. In general, the ordering of the items in the security header follows the most optimal layout for a  
 3685 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.  
 3686 Recipient to initiator message *Example*:

```

3687 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3688   xmlns:wssell="..." xmlns:wsse="..."
3689   xmlns:xenc="..." xmlns:ds="...">
3690 <S:Header>
3691   <x:Header1 wsu:Id="Header1" >
3692     ...
3693   </x:Header1>
3694   <wssell:EncryptedHeader wsu:Id="enc_Header2">
3695     <!-- Plaintext Header2
3696     <x:Header2 wsu:Id="Header2" >
3697       ...
3698     </x:Header2>
3699     -->
3700     ...
3701   </wssell:EncryptedHeader>
3702   ...
3703   <wsse:Security>
3704     <wsu:Timestamp wsu:Id="Timestamp">
3705       <wsu:Created>...</wsu:Created>
3706       <wsu:Expires>...</wsu:Expires>
3707     </wsu:Timestamp>
3708     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3709       ...
3710     </wsse:BinarySecurityToken>
3711     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3712       ...
3713       <xenc:ReferenceList>
3714         <xenc:DataReference URI="#enc_Signature" />
3715         <xenc:DataReference URI="#enc_SigConf1" />
3716         <xenc:DataReference URI="#enc_SigConf2" />
3717         ...
3718       </xenc:ReferenceList>
3719     </xenc:EncryptedKey>
3720     <xenc:EncryptedData ID="enc_SigConf2" >
3721       <!-- Plaintext SignatureConfirmation
3722       <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3723         ...
3724       </wssell:SignatureConfirmation>
3725       -->
3726       ...
3727     </xenc:EncryptedData>
3728     <xenc:EncryptedData ID="enc_SigConf1" >
3729       <!-- Plaintext SignatureConfirmation
3730       <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3731         ...
3732       </wssell:SignatureConfirmation>
3733       -->
3734       ...
3735     </xenc:EncryptedData>
3736     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3737       ...
3738     </wsse:BinarySecurityToken>
3739

```

```

3740 <xenc:EncryptedData ID="enc_Signature">
3741   <!-- Plaintext Signature
3742   <ds:Signature Id="Signature">
3743     <ds:SignedInfo>
3744       <ds:References>
3745         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3746         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3747         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3748         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3749         <ds:Reference URI="#Header1" >...</ds:Reference>
3750         <ds:Reference URI="#Header2" >...</ds:Reference>
3751         <ds:Reference URI="#Body" >...</ds:Reference>
3752       </ds:References>
3753     </ds:SignedInfo>
3754     <ds:SignatureValue>...</ds:SignatureValue>
3755     <ds:KeyInfo>
3756       <wsse:SecurityTokenReference>
3757         <wsse:Reference URI="#RecipientToken" />
3758       </wsse:SecurityTokenReference>
3759     </ds:KeyInfo>
3760   </ds:Signature>
3761   -->
3762   ...
3763 </xenc:EncryptedData>
3764 <xenc:ReferenceList>
3765   <xenc:DataReference URI="#enc_Body" />
3766   <xenc:DataReference URI="#enc_Header2" />
3767   ...
3768 </xenc:ReferenceList>
3769 </wsse:Security>
3770 </S:Header>
3771 <S:Body wsu:Id="Body">
3772   <xenc:EncryptedData Id="enc_Body">
3773     ...
3774     <ds:KeyInfo>
3775       <wsse:SecurityTokenReference>
3776         <wsse:Reference URI="#InitiatorEncryptedKey" />
3777       </wsse:SecurityTokenReference>
3778     </ds:KeyInfo>
3779   </xenc:EncryptedData>
3780 </S:Body>
3781 </S:Envelope>

```

---

3782 **D. Signed and Encrypted Elements in the Security**  
3783 **Header**

3784 This section lists the criteria for when various child elements of the Security header are signed and/or  
3785 encrypted at the message level including whether they are signed by the message signature or a  
3786 supporting signature. It assumes that there are no `sp:SignedElements` and no  
3787 `sp:EncryptedElements` assertions in the policy. If such assertions are present in the policy then  
3788 additional child elements of the security header might be signed and/or encrypted.

3789 **D.1 Elements signed by the message signature**

- 3790 1. The `wsu:Timestamp` element (Section 6.2).  
3791 2. All `wssell:SignatureConfirmation` elements (Section 9).  
3792 3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token],  
3793 [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption  
3794 Token] when [Token Protection] has a value of 'true' (Section 6.5).  
3795 4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed  
3796 Endorsing Supporting Tokens] (Section 8.5).

3797 **D.2 Elements signed by all endorsing signatures**

- 3798 1. The `ds:Signature` element that forms the message signature (Section 8.3).  
3799 2. The `wsu:Timestamp` element in the case of a transport binding (Section 8.3).

3800 **D.3 Elements signed by a specific endorsing signature**

- 3801 1. Security Tokens corresponding to [Endorsing Supporting Tokens] or [Signed Endorsing  
3802 Supporting Tokens] when [Token Protection] has a value of 'true' (Section 8.8).

3803 **D.4 Elements that are encrypted**

- 3804 1. The `ds:Signature` element that forms the message signature when [Signature Protection]  
3805 has a value of 'true' (Section 6.4).  
3806 2. All `wssell:SignatureConfirmation` elements when [Signature Protection] has a value  
3807 of 'true' (Section 6.4).  
3808 3. A `wsse:UsernameToken` may be encrypted when a transport binding is not being used  
3809 (Section 5.3.1).  
3810

---

3811 **E. Acknowledgements**

3812 The following individuals have participated in the creation of this specification and are gratefully  
3813 acknowledged:

3814 **Original Authors of the initial contribution:**

3815 Giovanni Della-Libera, Microsoft  
3816 Martin Gudgin, Microsoft  
3817 Phillip Hallam-Baker, VeriSign  
3818 Maryann Hondo, IBM  
3819 Hans Granqvist, Verisign  
3820 Chris Kaler, Microsoft (editor)  
3821 Hiroshi Maruyama, IBM  
3822 Michael McIntosh, IBM  
3823 Anthony Nadalin, IBM (editor)  
3824 Nataraj Nagaratnam, IBM  
3825 Rob Philpott, RSA Security  
3826 Hemma Prafullchandra, VeriSign  
3827 John Shewchuk, Microsoft  
3828 Doug Walter, Microsoft  
3829 Riaz Zolfonoon, RSA Security

3830  
3831 **Original Acknowledgements of the initial contribution:**

3832 Vaithialingam B. Balayoghan, Microsoft  
3833 Francisco Curbera, IBM  
3834 Christopher Ferris, IBM  
3835 Cédric Fournet, Microsoft  
3836 Andy Gordon, Microsoft  
3837 Tomasz Janczuk, Microsoft  
3838 David Melgar, IBM  
3839 Mike Perks, IBM  
3840 Bruce Rich, IBM  
3841 Jeffrey Schlimmer, Microsoft  
3842 Chris Sharp, IBM  
3843 Kent Tamura, IBM  
3844 T.R. Vishwanath, Microsoft  
3845 Elliot Waingold, Microsoft

3846  
3847 **TC Members during the development of this specification:**

3848 Don Adams, Tibco Software Inc.  
3849 Jan Alexander, Microsoft Corporation  
3850 Steve Anderson, BMC Software  
3851 Donal Arundel, IONA Technologies  
3852 Howard Bae, Oracle Corporation  
3853 Abbie Barbir, Nortel Networks Limited  
3854 Charlton Barreto, Adobe Systems  
3855 Mighael Botha, Software AG, Inc.  
3856 Toufic Boubez, Layer 7 Technologies Inc.  
3857 Norman Brickman, Mitre Corporation  
3858 Melissa Brumfield, Booz Allen Hamilton

3859 Lloyd Burch, Novell  
3860 Scott Cantor, Internet2  
3861 Greg Carpenter, Microsoft Corporation  
3862 Steve Carter, Novell  
3863 Symon Chang, BEA Systems, Inc.  
3864 Ching-Yun (C.Y.) Chao, IBM  
3865 Martin Chapman, Oracle Corporation  
3866 Kate Cherry, Lockheed Martin  
3867 Henry (Hyenvui) Chung, IBM  
3868 Luc Clement, Systinet Corp.  
3869 Paul Cotton, Microsoft Corporation  
3870 Glen Daniels, Sonic Software Corp.  
3871 Peter Davis, Neustar, Inc.  
3872 Martijn de Boer, SAP AG  
3873 Werner Dittmann, Siemens AG  
3874 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
3875 Fred Dushin, IONA Technologies  
3876 Petr Dvorak, Systinet Corp.  
3877 Colleen Evans, Microsoft Corporation  
3878 Ruchith Fernando, WSO2  
3879 Mark Fussell, Microsoft Corporation  
3880 Vijay Gajjala, Microsoft Corporation  
3881 Marc Goodner, Microsoft Corporation  
3882 Hans Granqvist, VeriSign  
3883 Martin Gudgin, Microsoft Corporation  
3884 Tony Gullotta, SOA Software Inc.  
3885 Jiandong Guo, Sun Microsystems  
3886 Phillip Hallam-Baker, VeriSign  
3887 Patrick Harding, Ping Identity Corporation  
3888 Heather Hinton, IBM  
3889 Frederick Hirsch, Nokia Corporation  
3890 Jeff Hodges, Neustar, Inc.  
3891 Will Hopkins, BEA Systems, Inc.  
3892 Alex Hristov, Otecia Incorporated  
3893 John Hughes, PA Consulting  
3894 Diane Jordan, IBM  
3895 Venugopal K, Sun Microsystems  
3896 Chris Kaler, Microsoft Corporation  
3897 Dana Kaufman, Forum Systems, Inc.  
3898 Paul Knight, Nortel Networks Limited  
3899 Ramanathan Krishnamurthy, IONA Technologies  
3900 Christopher Kurt, Microsoft Corporation  
3901 Kelvin Lawrence, IBM  
3902 Hubert Le Van Gong, Sun Microsystems  
3903 Jong Lee, BEA Systems, Inc.  
3904 Rich Levinson, Oracle Corporation  
3905 Tommy Lindberg, Dajeil Ltd.  
3906 Mark Little, JBoss Inc.  
3907 Hal Lockhart, BEA Systems, Inc.  
3908 Mike Lyons, Layer 7 Technologies Inc.  
3909 Eve Maler, Sun Microsystems  
3910 Ashok Malhotra, Oracle Corporation  
3911 Anand Mani, CrimsonLogic Pte Ltd  
3912 Jonathan Marsh, Microsoft Corporation  
3913 Robin Martherus, Oracle Corporation  
3914 Miko Matsumura, Infravio, Inc.  
3915 Gary McAfee, IBM

3916 Michael McIntosh, IBM  
3917 John Merrells, Sxip Networks SRL  
3918 Jeff Mischkinsky, Oracle Corporation  
3919 Prateek Mishra, Oracle Corporation  
3920 Bob Morgan, Internet2  
3921 Vamsi Motukuru, Oracle Corporation  
3922 Raajmohan Na, EDS  
3923 Anthony Nadalin, IBM  
3924 Andrew Nash, Reactivity, Inc.  
3925 Eric Newcomer, IONA Technologies  
3926 Duane Nickull, Adobe Systems  
3927 Toshihiro Nishimura, Fujitsu Limited  
3928 Rob Philpott, RSA Security  
3929 Denis Pilipchuk, BEA Systems, Inc.  
3930 Darren Platt, Ping Identity Corporation  
3931 Martin Raepfle, SAP AG  
3932 Nick Ragouzis, Enosis Group LLC  
3933 Prakash Reddy, CA  
3934 Alain Regnier, Ricoh Company, Ltd.  
3935 Irving Reid, Hewlett-Packard  
3936 Bruce Rich, IBM  
3937 Tom Rutt, Fujitsu Limited  
3938 Maneesh Sahu, Actional Corporation  
3939 Frank Siebenlist, Argonne National Laboratory  
3940 Joe Smith, Apani Networks  
3941 Davanum Srinivas, WSO2  
3942 Yakov Sverdlov, CA  
3943 Gene Thurston, AmberPoint  
3944 Victor Valle, IBM  
3945 Asir Vedamuthu, Microsoft Corporation  
3946 Greg Whitehead, Hewlett-Packard  
3947 Ron Williams, IBM  
3948 Corinna Witt, BEA Systems, Inc.  
3949 Kyle Young, Microsoft Corporation