



# WS-SecurityPolicy 1.3

## OASIS Editor Draft

1 February 2008

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802/ws-securitypolicy-1.3-spec-ed-01.doc>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802/ws-securitypolicy-1.3-spec-ed-01.pdf>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802/ws-securitypolicy-1.3-spec-ed-01.html>

#### Previous Version:

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

#### Latest Version:

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

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

### Abstract:

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

### Status:

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

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

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

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

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

---

## Notices

Copyright © OASIS® 1993–2008. All Rights Reserved. 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
1.7	Conformance .....	<del>14</del> 13
2	Security Policy Model .....	15
2.1	Security Assertion Model .....	15
2.2	Nested Policy Assertions .....	16
2.3	Security Binding Abstraction .....	16
3	Policy Considerations .....	18
3.1	Nested Policy .....	18
3.2	Policy Subjects .....	18
4	Protection Assertions .....	20
4.1	Integrity Assertions .....	20
4.1.1	SignedParts Assertion .....	20
4.1.2	SignedElements Assertion .....	21
4.2	Confidentiality Assertions .....	22
4.2.1	EncryptedParts Assertion .....	22
4.2.2	EncryptedElements Assertion .....	23
4.2.3	ContentEncryptedElements Assertion .....	24
4.3	Required Elements Assertion .....	24
4.3.1	RequiredElements Assertion .....	24
4.3.2	RequiredParts Assertion .....	25
5	Token Assertions .....	26
5.1	Token Inclusion .....	26
5.1.1	Token Inclusion Values .....	26
5.1.2	Token Inclusion and Token References .....	27
5.2	Token Issuer and Required Claims .....	27
5.2.1	Token Issuer .....	27
5.2.2	Token Issuer Name .....	27
5.2.3	Required Claims .....	27
5.2.4	Processing Rules and Token Matching .....	28
5.3	Token Properties .....	28
5.3.1	[Derived Keys] Property .....	28
5.3.2	[Explicit Derived Keys] Property .....	28
5.3.3	[Implied Derived Keys] Property .....	28
5.4	Token Assertion Types .....	28

5.4.1	UsernameToken Assertion .....	28
5.4.2	ICreatessuedToken Assertion .....	30
5.4.3	X509Token Assertion .....	32
5.4.4	KerberosToken Assertion.....	34
5.4.5	SpnegoContextToken Assertion.....	36
5.4.6	SecurityContextToken Assertion .....	37
5.4.7	SecureConversationToken Assertion.....	38
5.4.8	SamlToken Assertion .....	41
5.4.9	RelToken Assertion.....	43
5.4.10	HttpsToken Assertion.....	44
5.4.11	KeyValueToken Assertion .....	45
6	Security Binding Properties .....	48
6.1	[Algorithm Suite] Property.....	48
6.2	[Timestamp] Property .....	50
6.3	[Protection Order] Property .....	50
6.4	[Signature Protection] Property.....	50
6.5	[Token Protection] Property.....	50
6.6	[Entire Header and Body Signatures] Property .....	51
6.7	[Security Header Layout] Property.....	51
6.7.1	Strict Layout Rules for WSS 1.0 .....	51
7	Security Binding Assertions .....	53
7.1	AlgorithmSuite Assertion .....	53
7.2	Layout Assertion .....	55
7.3	TransportBinding Assertion .....	56
7.4	SymmetricBinding Assertion .....	57
7.5	AsymmetricBinding Assertion .....	59
8	Supporting Tokens.....	62
8.1	SupportingTokens Assertion.....	63
8.2	SignedSupportingTokens Assertion .....	64
8.3	EndorsingSupportingTokens Assertion .....	66
8.4	SignedEndorsingSupportingTokens Assertion .....	68
8.5	SignedEncryptedSupportingTokens Assertion .....	70
8.6	EncryptedSupportingTokens Assertion .....	70
8.7	EndorsingEncryptedSupportingTokens Assertion.....	70
8.8	SignedEndorsingEncryptedSupportingTokens Assertion .....	70
8.9	Interaction between [Token Protection] property and supporting token assertions .....	70
8.10	Example .....	71
9	WSS: SOAP Message Security Options .....	72
9.1	Wss10 Assertion.....	73
9.2	Wss11 Assertion.....	74
10	WS-Trust Options.....	76
10.1	Trust13 Assertion .....	77
11	Guidance on creating new assertions and assertion extensibility.....	79

11.1	General Design Points .....	79
11.2	Detailed Design Guidance .....	79
12	Security Considerations.....	81
A.	Assertions and WS-PolicyAttachment .....	82
A.1	Endpoint Policy Subject Assertions .....	82
A.1.1	Security Binding Assertions .....	82
A.1.2	Token Assertions .....	82
A.1.3	WSS: SOAP Message Security 1.0 Assertions .....	82
A.1.4	WSS: SOAP Message Security 1.1 Assertions .....	82
A.1.5	Trust 1.0 Assertions .....	82
A.2	Operation Policy Subject Assertions .....	82
A.2.1	Security Binding Assertions .....	82
A.2.2	Supporting Token Assertions .....	82
A.3	Message Policy Subject Assertions .....	83
A.3.1	Supporting Token Assertions .....	83
A.3.2	Protection Assertions .....	83
A.4	Assertions With Undefined Policy Subject .....	83
A.4.1	General Assertions .....	83
A.4.2	Token Usage Assertions .....	83
A.4.3	Token Assertions .....	83
B.	Issued Token Policy .....	85
C.	Strict Security Header Layout Examples .....	87
C.1	Transport Binding .....	87
C.1.1	Policy .....	87
C.1.2	Initiator to Recipient Messages .....	88
C.1.3	Recipient to Initiator Messages .....	89
C.2	Symmetric Binding .....	90
C.2.1	Policy .....	91
C.2.2	Initiator to Recipient Messages .....	92
C.2.3	Recipient to Initiator Messages .....	96
C.3	Asymmetric Binding .....	99
C.3.1	Policy .....	99
C.3.2	Initiator to Recipient Messages .....	101
C.3.3	Recipient to Initiator Messages .....	105
D.	Signed and Encrypted Elements in the Security Header .....	109
D.1	Elements signed by the message signature .....	109
D.2	Elements signed by all endorsing signatures .....	109
D.3	Elements signed by a specific endorsing signature .....	109
D.4	Elements that are encrypted .....	109
E.	Acknowledgements .....	110
F.	Revision History .....	113

# 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 1.5 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 URIs that MUST be used by implementations of this specification are:

```

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

```

66

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

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

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



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>	[WS-Addressing]
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
<a href="#">sp13</a>	<a href="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802">http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802</a>	<a href="#">This specification</a>
<a href="#">wsp</a>	<a href="http://www.w3.org/ns/ws-policy">http://www.w3.org/ns/ws-policy</a>	<a href="#">[WS-Policy]</a>

## 70 1.3 Schema Files

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

73 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2.xsd>  
74 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.3.xsd>

## 75 1.4 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

### 92 1.4.1 Notational Conventions

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

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

- 97 • The syntax appears as an XML instance, but values in italics indicate data types instead of literal  
98 values.
- 99 • Characters are appended to elements and attributes to indicate cardinality:
  - 100 ○ "?" (0 or 1)
  - 101 ○ "\*" (0 or more)
  - 102 ○ "+" (1 or more)

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

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

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

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

124 In this document reference is made to the `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires`  
 125 elements in a utility schema (<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>). The `wsu:Id` attribute and the `wsu:Created` and `wsu:Expires` elements were added to the  
 126 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp  
 127 element could reference it (as is done here).  
 128

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

## 135 1.5 Normative References

- |     |            |   |
|-----|------------|---|
| 136 | [RFC2119]  | S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.       |
| 137 |            |   |
| 138 |            | <a href="http://www.ietf.org/rfc/rfc2119.txt">http://www.ietf.org/rfc/rfc2119.txt</a>                                   |
| 139 |            |   |
| 140 | [SOAP]     | W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.   |
| 141 |            | <a href="http://www.w3.org/TR/2000/NOTE-SOAP-20000508/">http://www.w3.org/TR/2000/NOTE-SOAP-20000508/</a>               |
| 142 |            |   |
| 143 | [SOAP12]   | W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003.   |
| 144 |            |   |
| 145 |            | <a href="http://www.w3.org/TR/2003/REC-soap12-part1-20030624/">http://www.w3.org/TR/2003/REC-soap12-part1-20030624/</a> |
| 146 |            |   |
| 147 | [SOAPNorm] | W3C Working Group Note, "SOAP Version 1.2 Message Normalization", 8 October 2003.                                       |
| 148 |            |   |

149 <http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>  
150  
151 [URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers  
152 (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe  
153 Systems, January 2005.  
154 <http://www.ietf.org/rfc/rfc3986.txt>  
155  
156 [RFC2068] IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January  
157 1997  
158 <http://www.ietf.org/rfc/rfc2068.txt>  
159  
160 [RFC2246] IETF Standard, "The TLS Protocol", January 1999.  
161 <http://www.ietf.org/rfc/rfc2246.txt>  
162  
163 [SwA] W3C Note, "SOAP Messages with Attachments", 11 December 2000  
164 <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211>  
165  
166 [WS-Addressing] W3C Recommendation, "Web Services Addressing (WS-Addressing)",  
167 9 May 2006.  
168 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>  
169  
170 [WS-Policy] ~~[W3C Recommendation, "Web Services Policy 1.5 - Framework", 04](http://www.w3.org/TR/2007/REC-ws-policy-20070904/)~~  
171 ~~[September 2007.](http://www.w3.org/TR/2007/REC-ws-policy-20070904/)~~  
172 ~~<http://www.w3.org/TR/2007/REC-ws-policy-20070904/>~~  
173 W3C Member Submission "Web Services Policy 1.2 - Framework", 25  
174 April 2006.  
175 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>  
176 ~~[W3C Candidate Recommendation "Web Services Policy 1.5—](http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/)~~  
177 ~~[Framework", 28 February 2007](http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/)~~  
178 ~~<http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/>~~  
179  
180 [WS-PolicyAttachment] ~~[W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04](http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/)~~  
181 ~~[September 2007.](http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/)~~  
182 ~~<http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/>~~  
183 W3C Member Submission "Web Services Policy 1.2 - Attachment", 25  
184 April 2006.  
185 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)  
186 [20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)  
187 ~~[W3C Candidate Recommendation "Web Services Policy 1.5—](http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/)~~  
188 ~~[Attachment", 28 February 2007](http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/)~~  
189 ~~<http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/>~~  
190  
191 [WS-Trust] ~~[OASIS Standard, "WS-Trust 1.4", 2008](http://docs.oasis-open.org/ws-sx/ws-trust/200802)~~  
192 ~~<http://docs.oasis-open.org/ws-sx/ws-trust/200802>~~  
193 OASIS ~~Committee Draft Standard~~, "WS-Trust 1.3", ~~September March~~  
194 ~~2007~~

195		<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512</a>
196		
197	[WS-SecureConversation]	OASIS <del>Committee Draft</del> Standard, "WS-SecureConversation 1.3",
198		September 2006
199		<a href="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512">http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512</a>
200		
201	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message
202		Security 1.0 (WS-Security 2004)", March 2004.
203		<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-</a>
204		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf">message-security-1.0.pdf</a>
205		
206	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message
207		Security 1.1 (WS-Security 2004)", February 2006.
208		<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-</a>
209		<a href="http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf">spec-os-SOAPMessageSecurity.pdf</a>
210		
211	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile",
212		March 2004
213		<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>
214		<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>
215		
216	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile
217		1.1", February 2006
218		<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>
219		<a href="http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf">spec-os-UsernameTokenProfile.pdf</a>
220		
221	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token
222		Profile", March 2004
223		<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>
224		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf">profile-1.0.pdf</a>
225		
226	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token
227		Profile", February 2006
228		<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>
229		<a href="http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf">spec-os-x509TokenProfile.pdf</a>
230		
231	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1",
232		February 2006
233		<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>
234		<a href="http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf">spec-os-KerberosTokenProfile.pdf</a>
235		
236	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile",
237		December 2004
238		<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>
239		
240	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1",
241		February 2006

242		<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-spec-os-SAMLTokenProfile.pdf</a>
243		
244		
245	[WSS:RELTOKENPROFILE1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
246		
247		<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>
248		
249	[WSS:RELTOKENPROFILE1.1]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile 1.1", February 2006
250		
251		<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-wss-rel-token-profile-1.1.pdf</a>
252		
253		
254	[WSS:SWAPROFILE1.1]	OASIS Standard, "Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1", February 2006
255		
256		<a href="http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwapProfile.pdf">http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwapProfile.pdf</a>
257		
258		
259	[XML-ENCRYPT]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
260		
261		<a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/</a>
262		
263	[XML-SIGNATURE]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002.
264		
265		<a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/</a>
266		
267	[XPATH]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999.
268		
269		<a href="http://www.w3.org/TR/1999/REC-xpath-19991116">http://www.w3.org/TR/1999/REC-xpath-19991116</a>
270		
271	[XPath 2.0 Filter]	W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November 2002.
272		
273		<a href="http://www.w3.org/TR/2002/REC-xmlsig-filter2-20021108/">http://www.w3.org/TR/2002/REC-xmlsig-filter2-20021108/</a>
274		
275	[XML-SCHEMA1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition", 28 October 2004.
276		
277		<a href="http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/">http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/</a>
278		
279	[XML-SCHEMA2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition", 28 October 2004.
280		
281		<a href="http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/">http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/</a>
282		

## 283 1.6 Non-Normative References

284 None.

285  
286  
287  
288  
289  
  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
  
301  
302  
303  
304  
305  
306  
  
307

## 1.7 Conformance

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

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

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

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

---

## 308 2 Security Policy Model

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

313  
314 The primary goal of this specification is to define an initial set of patterns or sets of assertions that  
315 represent common ways to describe how messages are secured on a communication path. The intent is  
316 to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging  
317 transport security, but to be specific enough to ensure interoperability based on assertion matching.

318  
319 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for  
320 selecting policy alternatives and the attachment mechanism for associating policy assertions with web  
321 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters  
322 or attributes. This enables first-level, QName based assertion matching without security domain-specific  
323 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed  
324 set of policy alternatives that are shared by the two parties attempting to establish a secure  
325 communication path. Parameters defined by this specification represent additional information for  
326 engaging behaviors that do not need to participate in matching. When multiple security policy assertions  
327 of the same type with parameters present occur in the same policy alternative the parameters should be  
328 treated as a union. Note that a service may choose to accept messages that do not match its policy.

329  
330 In general, assertions defined in this specification allow additional attributes, based on schemas, to be  
331 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not  
332 match based on these attributes. Attributes specified on the assertion element that are not defined in this  
333 specification or in WS-Policy are to be treated as informational properties.

### 334 2.1 Security Assertion Model

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

342  
343 To indicate the scope of protection, assertions identify message parts that are to be protected in a  
344 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

345  
346 The general aspects of security includes the relationships between or characteristics of the environment  
347 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality  
348 protection and which are supporting, the applicable algorithms to use, etc.

349

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

355

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

360

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

## 363 2.2 Nested Policy Assertions

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

## 370 2.3 Security Binding Abstraction

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

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

376

377 A binding defines the following security characteristics:

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

386

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

393



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

- 397 • TransportBinding (Section 7.3)
- 398 • SymmetricBinding (Section 7.4)
- 399 • AsymmetricBinding (Section 7.5)

---

## 400 3 Policy Considerations

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

### 403 3.1 Nested Policy

404 This specification makes extensive use of nested policy assertions as described in the [Policy Assertion](#)  
405 [Nesting](#) section of WS-Policy.  
406

### 407 3.2 Policy Subjects

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

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

#### 412 [Message Policy Subject]

413 This property identifies a Message Policy Subject [[WS-PolicyAttachment](#)]. WS-PolicyAttachment defines  
414 seven WSDL [WSDL 1.1] policy attachment points with Message Policy Subject:  
415

416 wsdl:message

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

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

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

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

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

#### 425 [Operation Policy Subject]

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

427 wsdl:portType/wsdl:operation

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

430 wsdl:binding/wsdl:operation

431 A policy expression containing one or more token assertions MUST be attached to a  
432 wsdl:binding/wsdl:operation.  
433  
434

#### 435 [Endpoint Policy Subject]

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

438 wsdl:portType

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

441    wsdl:binding

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

444    wsdl:port

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

---

## 447 4 Protection Assertions

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

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

### 455 4.1 Integrity Assertions

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

#### 459 4.1.1 SignedParts Assertion

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

465  
466 There MAY be multiple SignedParts assertions present. Multiple SignedParts assertions present within a  
467 policy alternative are equivalent to a single SignedParts assertion containing the union of all specified  
468 message parts. Note that this assertion does not require that a given part appear in a message, just that if  
469 such a part appears, it requires integrity protection.

#### 470 Syntax

```
471 <sp:SignedParts xmlns:sp="..." ... >  
472   <sp:Body />?  
473   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
474   <sp:Attachments />?  
475   ...  
476 </sp:SignedParts>
```

477  
478 The following describes the attributes and elements listed in the schema outlined above:

479 /sp:SignedParts

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

483 /sp:SignedParts/sp:Body

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

486 /sp:SignedParts/sp:Header

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

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

497 /sp:SignedParts/sp:Header/@Name

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

501 /sp:SignedParts/sp:Header/@Namespace

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

504 /sp:SignedParts/sp:Attachments

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

## 510 4.1.2 SignedElements Assertion

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

516

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

### 520 Syntax

```
521 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
522 <sp:XPath>xs:string</sp:XPath>+  
523 <sp13:XPath2 Filter="xs:string">xs:string</sp13:XPath2>+  
524 ...  
525 </sp:SignedElements>
```

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

527 /sp:SignedElements

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

529 /sp:SignedElements/@XPathVersion

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

532 /sp:SignedElements/sp:XPath

533 This element contains a string specifying an XPath expression that identifies the nodes to be  
534 integrity protected. The XPath expression is evaluated against the S:Envelope element node of  
535 the message. Multiple instances of this element MAY appear within this assertion and SHOULD  
536 be treated as separate references in a signature when message security is used.

537 [/sp:SignedElements/sp:XPath2](#)

538 [This element contains a string specifying an XPath 2 expression that identifies the nodes to be](#)  
539 [integrity protected. The XPath expression is evaluated against the S:Envelope element node of](#)  
540 [the message. Multiple instances of this element MAY appear within this assertion and SHOULD](#)  
541 [be treated as separate references in a signature when message security is used.](#)

542 [/sp:SignedElements/sp:XPath2@Filter](#)

543 [This REQUIRED attribute contains a string to specify an \[XPath Filter 2.0\] transform to apply.](#)

## 544 4.2 Confidentiality Assertions

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

### 548 4.2.1 EncryptedParts Assertion

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

554  
555 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present  
556 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all  
557 specified message parts. Note that this assertion does not require that a given part appear in a message,  
558 just that if such a part appears, it requires confidentiality protection.

#### 559 Syntax

```
560 <sp:EncryptedParts xmlns:sp="..." ... >  
561 <sp:Body/>?  
562 <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />?  
563 <sp:Attachments />?  
564 ...  
565 </sp:EncryptedParts>
```

566  
567 The following describes the attributes and elements listed in the schema outlined above:

568 [/sp:EncryptedParts](#)

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

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

572 [/sp:EncryptedParts/sp:Body](#)

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

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

587 /sp:EncryptedParts/sp:Header/@Name

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

591 /sp:EncryptedParts/sp:Header/@Namespace

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

594 /sp:EncryptedParts/sp:Attachments

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

## 600 4.2.2 EncryptedElements Assertion

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

606

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

### 610 Syntax

```
611 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
612   <sp:XPath>xs:string</sp:XPath>+  
613   ...  
614 </sp:EncryptedElements>
```

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

616 /sp:EncryptedElements

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

619 /sp:EncryptedElements/@XPathVersion

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

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

### 627 **4.2.3 ContentEncryptedElements Assertion**

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

633  
634 There MAY be multiple ContentEncryptedElements assertions present. Multiple  
635 ContentEncryptedElements assertions present within a policy alternative are equivalent to a single  
636 ContentEncryptedElements assertion containing the union of all specified XPath expressions.

#### 637 **Syntax**

```
638 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
639 <sp:XPath>xs:string</sp:XPath>+  
640 ...  
641 </sp:ContentEncryptedElements>
```

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

643 /sp:ContentEncryptedElements

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

646 /sp:ContentEncryptedElements/@XPathVersion

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

649 /sp:ContentEncryptedElements/sp:XPath

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

### 654 **4.3 Required Elements Assertion**

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

657

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

#### 661 **4.3.1 RequiredElements Assertion**

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

664



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

#### 668 **Syntax**

```
669 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
670 <sp:XPath>xs:string</sp:XPath> +  
671 ...  
672 </sp:RequiredElements>
```

673

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

675 /sp:RequiredElements

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

677 /sp:RequiredElements/@XPathVersion

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

680 /sp:RequiredElements/sp:XPath

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

### 685 **4.3.2 RequiredParts Assertion**

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

689

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

#### 693 **Syntax**

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

697

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

699 /sp:RequiredParts/sp:Header

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

701 /sp:RequiredParts/sp:Header/@Name

702 This REQUIRED attribute indicates the local name of the SOAPHeader that needs to be present  
703 in the message.

704 /sp:RequiredParts/sp:Header/@Namespace

705 This REQUIRED attribute indicates the namespace of the SOAP header that needs to be present  
706 in the message.

## 707 5 Token Assertions

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

### 712 5.1 Token Inclusion

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

#### 718 5.1.1 Token Inclusion Values

719 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.

720

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

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

## 727 **5.1.2 Token Inclusion and Token References**

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

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

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

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

## 744 **5.2 Token Issuer and Required Claims**

### 745 **5.2.1 Token Issuer**

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

### 750 **5.2.2 Token Issuer Name**

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

755  
756 It is out of scope of this specification how the relationship between the issuer's logical name and the  
757 physical manifestation of the issuer in the security token is defined.

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

### 760 **5.2.3 Required Claims**

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

764  
765 This element indicates the REQUIRED claims that the security token must contain in order to satisfy the  
766 requirements of the token assertion.

767  
768 Individual token assertions MAY further limit what claims MAY be specified for that specific token  
769 assertion.

## 770 **5.2.4 Processing Rules and Token Matching**

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

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

- 778 1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and  
779 T2 is issued by issuer B and contains claims C3 and C4.
- 780 2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is  
781 also issued by issuer A and contains claim C2 and T3 is issued by issuer B and  
782 contains claims C3 and C4.
- 783 3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2,  
784 T2 is issued by issuer B and contains claim C3 and T3 is also issued by issuer B and  
785 contains claim C4.
- 786 4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is  
787 also issued by issuer A and contains claim C2, T3 is issued by issuer B and contains  
788 claim C3 and T4 is also issued by issuer B and contains claim C4.

## 790 **5.3 Token Properties**

### 791 **5.3.1 [Derived Keys] Property**

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

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

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

### 800 **5.3.2 [Explicit Derived Keys] Property**

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

### 804 **5.3.3 [Implied Derived Keys] Property**

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

## 808 **5.4 Token Assertion Types**

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

### 810 **5.4.1 UsernameToken Assertion**

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

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

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

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

820

## 821 Syntax

```
822 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
823 (
824   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
825   <sp:IssuerName>xs:anyURI</sp:IssuerName>
826 ) ?
827 <wst:Claims Dialect="..."> ... </wst:Claims> ?
828 <wsp:Policy xmlns:wsp="...">
829   (
830     <sp:NoPassword ... /> |
831     <sp:HashPassword ... />
832   ) |
833   (
834     <sp13:Created .../> ?
835     <sp13:Nonce .../> ?
836   ) ?
837   (
838     <sp:RequireDerivedKeys /> |
839     <sp:RequireImpliedDerivedKeys ... /> |
840     <sp:RequireExplicitDerivedKeys ... />
841   ) ?
842   (
843     <sp:WssUsernameToken10 ... /> |
844     <sp:WssUsernameToken11 ... />
845   ) ?
846   ...
847 </wsp:Policy>
848   ...
849 </sp:UsernameToken>
```

850

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

852 /sp:UsernameToken

853 This identifies a UsernameToken assertion.

854 /sp:UsernameToken/@sp:IncludeToken

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

856 /sp:UsernameToken/sp:Issuer

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

859 /sp:UsernameToken/sp:IssuerName

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

862 /sp:UsernameToken/wst:Claims

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

865 /sp:UsernameToken/wsp:Policy

866 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken  
867 assertion.

868 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

871 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

876 [/sp13:UsernameToken/wsp:Policy/sp13:Created](#)

877 [This OPTIONAL element is a policy assertion that MUST only be used with the default clear text](#)  
878 [password case, and, if present, indicates that the wsse:Created element MUST be present in the](#)  
879 [Username token.](#)

880 [/sp13:UsernameToken/wsp:Policy/sp13:Nonce](#)

881 [This OPTIONAL element is a policy assertion that MUST only be used with the default clear text](#)  
882 [password case, and, if present, that indicates that the wsse:Nonce element MUST be present in](#)  
883 [the Username token.](#)

884 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

887 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

891 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

895 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

898 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

## 901 **5.4.2 IssuedToken Assertion**

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

### 906 **Syntax**

```

907 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
908 (
909 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
910 <sp:IssuerName>xs:anyURI</sp:IssuerName>
911 ) ?
912 <wst:Claims Dialect="..."> ... </wst:Claims> ?
913 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
914 ...
915 </sp:RequestSecurityTokenTemplate>
916 <wsp:Policy xmlns:wsp="...">
917 (
918 <sp:RequireDerivedKeys ... /> |
919 <sp:RequireImpliedDerivedKeys ... /> |
920 <sp:RequireExplicitDerivedKeys ... />
921 ) ?
922 <sp:RequireExternalReference ... /> ?
923 <sp:RequireInternalReference ... /> ?
924 ...
925 </wsp:Policy>
926 ...
927 </sp:IssuedToken>

```

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

929 /sp:IssuedToken

930 This identifies an IssuedToken assertion.

931 /sp:IssuedToken/@sp:IncludeToken

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

933 /sp:IssuedToken/sp:Issuer

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

936 /sp:IssuedToken/sp:IssuerName

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

939 /sp:IssuedToken/wst:Claims

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

942 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

947 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

948 This OPTIONAL attribute contains a WS-Trust specification namespace URI identifying the  
949 version of WS-Trust referenced by the contents of this element. [For example, when using Trust](http://docs.oasis-open.org/ws-sx/ws-trust/200512)  
950 [1.3 the URI http://docs.oasis-open.org/ws-sx/ws-trust/200512 should be used and when using](http://docs.oasis-open.org/ws-sx/ws-trust/200512)  
951 [Trust 1.4 the URI http://docs.oasis-open.org/ws-sx/ws-trust/200802 should be used.](http://docs.oasis-open.org/ws-sx/ws-trust/200802)

952 /sp:IssuedToken/wsp:Policy

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

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

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

958 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

962 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

966 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

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

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

970 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

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

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

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

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

### 979 5.4.3 X509Token Assertion

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

#### 981 Syntax

```
982 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
983   (  
984     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
985     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
986   ) ?  
987   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```



```

988 <wsp:Policy xmlns:wsp="...">
989   (
990     <sp:RequireDerivedKeys ... /> |
991     <sp:RequireExplicitDerivedKeys ... /> |
992     <sp:RequireImpliedDerivedKeys ... />
993   ) ?
994   <sp:RequireKeyIdentifierReference ... /> ?
995   <sp:RequireIssuerSerialReference ... /> ?
996   <sp:RequireEmbeddedTokenReference ... /> ?
997   <sp:RequireThumbprintReference ... /> ?
998   (
999     <sp:WssX509V3Token10 ... /> |
1000    <sp:WssX509Pkcs7Token10 ... /> |
1001    <sp:WssX509PkiPathV1Token10 ... /> |
1002    <sp:WssX509V1Token11 ... /> |
1003    <sp:WssX509V3Token11 ... /> |
1004    <sp:WssX509Pkcs7Token11 ... /> |
1005    <sp:WssX509PkiPathV1Token11 ... />
1006  ) ?
1007   ...
1008 </wsp:Policy>
1009 ...
1010 </sp:X509Token>

```

1011

1012 The following describes the attributes and elements listed in the schema outlined above:

1013 /sp:X509Token

1014 This identifies an X509Token assertion.

1015 /sp:X509Token/@sp:IncludeToken

1016 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1017 /sp:X509Token/sp:Issuer

1018 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1019 of the sp:X509Token.

1020 /sp:X509Token/sp:IssuerName

1021 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:X509Token  
1022 issuer.

1023 /sp:X509Token/wst:Claims

1024 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1025 order to satisfy the token assertion requirements.

1026 /sp:X509Token/wsp:Policy

1027 This REQUIRED element identifies additional requirements for use of the sp:X509Token  
1028 assertion.

1029 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

1030 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1031 and [Implied Derived Keys] properties for this token to 'true'.

1032 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

1033 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1034 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1035 'false'.

1036 /sp:X509Token/wsp:Policy/sp:RequireImpliedDerivedKeys

1037 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1038 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1039 'false'.

1040 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference  
 1041 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
 1042 REQUIRED when referencing this token.

1043 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference  
 1044 This OPTIONAL element is a policy assertion that indicates that an issuer serial reference is  
 1045 REQUIRED when referencing this token.

1046 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference  
 1047 This OPTIONAL element is a policy assertion that indicates that an embedded token reference is  
 1048 REQUIRED when referencing this token.

1049 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference  
 1050 This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is  
 1051 REQUIRED when referencing this token.

1052 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10  
 1053 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should  
 1054 be used as defined in [WSS:X509TokenProfile1.0].

1055 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10  
 1056 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be  
 1057 used as defined in [WSS:X509TokenProfile1.0].

1058 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10  
 1059 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1  
 1060 token should be used as defined in [WSS:X509TokenProfile1.0].

1061 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11  
 1062 This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should  
 1063 be used as defined in [WSS:X509TokenProfile1.1].

1064 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11  
 1065 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should  
 1066 be used as defined in [WSS:X509TokenProfile1.1].

1067 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11  
 1068 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be  
 1069 used as defined in [WSS:X509TokenProfile1.1].

1070 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11  
 1071 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1  
 1072 token should be used as defined in [WSS:X509TokenProfile1.1].

#### 1073 5.4.4 KerberosToken Assertion

1074 This element represents a requirement for a Kerberos token [WSS:KerberosToken1.1].

#### 1075 Syntax

```
1076 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1077 (
1078   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1079   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1080 ) ?
```

```

1081 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1082 <wsp:Policy xmlns:wsp="...">
1083   (
1084     <sp:RequireDerivedKeys ... /> |
1085     <sp:RequireImpliedDerivedKeys ... /> |
1086     <sp:RequireExplicitDerivedKeys ... />
1087   ) ?
1088   <sp:RequireKeyIdentifierReference ... /> ?
1089   (
1090     <sp:WssKerberosV5ApReqToken11 ... /> |
1091     <sp:WssGssKerberosV5ApReqToken11 ... />
1092   ) ?
1093   ...
1094 </wsp:Policy>
1095 ...
1096 ...
1097 </sp:KerberosToken>

```

1098  
1099 The following describes the attributes and elements listed in the schema outlined above:

1100 /sp:KerberosToken

1101 This identifies a KerberosV5ApReqToken assertion.

1102 /sp:KerberosToken/@sp:IncludeToken

1103 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1104 /sp:KerberosToken/sp:Issuer

1105 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1106 of the sp:KerberosToken.

1107 /sp:KerberosToken/sp:IssuerName

1108 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken  
1109 issuer.

1110 /sp:KerberosToken/wst:Claims

1111 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1112 order to satisfy the token assertion requirements.

1113 /sp:KerberosToken/wsp:Policy

1114 This REQUIRED element identifies additional requirements for use of the sp:KerberosToken  
1115 assertion.

1116 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1117 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1118 and [Implied Derived Keys] properties for this token to 'true'.

1119 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1120 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1121 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1122 'false'.

1123 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1124 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1125 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1126 'false'.

1127 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1128 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1129 REQUIRED when referencing this token.

1130 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken1

1131 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ  
1132 token should be used as defined in [WSS:KerberosTokenProfile1.1].

1133 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken1

1134 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-  
1135 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

## 1136 5.4.5 SpnegoContextToken Assertion

1137 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg  
1138 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

### 1139 Syntax

```
1140 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1141 (   
1142 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1143 <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1144 ) ?  
1145 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1146 <wsp:Policy xmlns:wsp="...">  
1147 (   
1148 <sp:RequireDerivedKeys ... /> |  
1149 <sp:RequireImpliedDerivedKeys ... /> |  
1150 <sp:RequireExplicitDerivedKeys ... />  
1151 ) ?  
1152 <sp:MustNotSendCancel ... /> ?  
1153 <sp:MustNotSendAmend ... /> ?  
1154 <sp:MustNotSendRenew ... /> ?  
1155 ...  
1156 </wsp:Policy>  
1157 ...  
1158 </sp:SpnegoContextToken>
```

1159  
1160 The following describes the attributes and elements listed in the schema outlined above:

1161 /sp:SpnegoContextToken

1162 This identifies a SpnegoContextToken assertion.

1163 /sp:SpnegoContextToken/@sp:IncludeToken

1164 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1165 /sp:SpnegoContextToken/sp:Issuer

1166 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
1167 for the Spnego Context Token.

1168 /sp:SpnegoContextToken/sp:IssuerName

1169 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1170 sp:SpnegoContextToken issuer.

1171 /sp:SpnegoContextToken/wst:Claims

1172 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1173 order to satisfy the token assertion requirements.

1174 /sp:SpnegoContextToken/wsp:Policy

1175 This REQUIRED element identifies additional requirements for use of the  
 1176 sp:SpnegoContextToken assertion.

1177 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1178 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1179 and [Implied Derived Keys] properties for this token to 'true'.

1180 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1181 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1182 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1183 'false'.

1184 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1185 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1186 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1187 'false'.

1188 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1189 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
 1190 token does not support SCT/Cancel RST messages. If this assertion is missing it means that  
 1191 SCT/Cancel RST messages are supported by the STS.

1192 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

1193 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
 1194 token does not support SCT/Amend RST messages. If this assertion is missing it means that  
 1195 SCT/Amend RST messages are supported by the STS.

1196 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1197 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
 1198 token does not support SCT/Renew RST messages. If this assertion is missing it means that  
 1199 SCT/Renew RST messages are supported by the STS.

## 1200 5.4.6 SecurityContextToken Assertion

1201 This element represents a requirement for a SecurityContextToken token.

### 1202 Syntax

```

1203 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1204 (
1205   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1206   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1207 ) ?
1208 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1209 <wsp:Policy xmlns:wsp="...">
1210   (
1211     <sp:RequireDerivedKeys ... /> |
1212     <sp:RequireImpliedDerivedKeys ... /> |
1213     <sp:RequireExplicitDerivedKeys ... />
1214   ) ?
1215   <sp:RequireExternalUriReference ... /> ?
1216   <sp:SC13SecurityContextToken... /> ?
1217   ...
1218 </wsp:Policy>
1219 ...
1220 </sp:SecurityContextToken>

```

1221

1222 The following describes the attributes and elements listed in the schema outlined above:

1223 /sp:SecurityContextToken

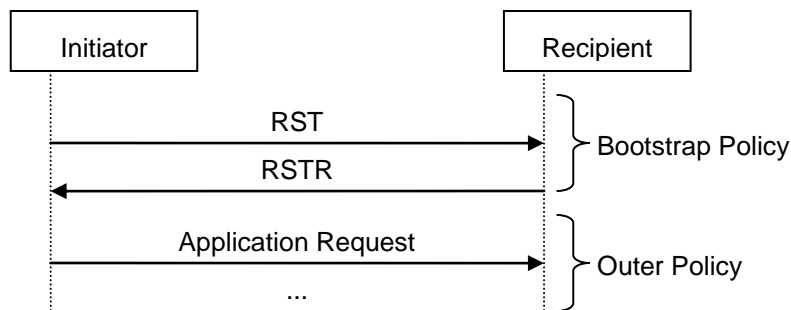
- 1224 This identifies a SecurityContextToken assertion.
- 1225 /sp:SecurityContextToken/@sp:IncludeToken
- 1226 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 1227 /sp:SecurityContextToken/sp:Issuer
- 1228 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1229 of the sp:SecurityContextToken.
- 1230 /sp:SecurityContextToken/sp:IssuerName
- 1231 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1232 sp:SecurityContextToken issuer.
- 1233 /sp:SecurityContextToken/wst:Claims
- 1234 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1235 order to satisfy the token assertion requirements.
- 1236 /sp:SecurityContextToken/wsp:Policy
- 1237 This REQUIRED element identifies additional requirements for use of the  
1238 sp:SecurityContextToken assertion.
- 1239 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys
- 1240 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1241 and [Implied Derived Keys] properties for this token to 'true'.
- 1242 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1243 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1244 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1245 'false'.
- 1246 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1247 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1248 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1249 'false'.
- 1250 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference
- 1251 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
1252 REQUIRED when referencing this token.
- 1253 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken
- 1254 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should  
1255 be used as defined in [\[WS-SecureConversation\]](#).
- 1256
- 1257 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that  
1258 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If  
1259 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the  
1260 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

## 1261 **5.4.7 SecureConversationToken Assertion**

- 1262 This element represents a requirement for a Security Context Token retrieved from the indicated issuer  
1263 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the  
1264 service endpoint address.

1265

1266 Note: This assertion describes the token accepted by the target service. Because this token is issued by  
 1267 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD  
 1268 contain a bootstrap policy indicating the security binding and policy that is used when requesting this  
 1269 token from the target service. That is, the bootstrap policy is used to obtain the token and then the  
 1270 current (outer) policy is used when making requests with the token. This is illustrated in the diagram  
 1271 below.



1272

1273 **Syntax**

```

1274 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1275 (
1276   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1277   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1278 ) ?
1279 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1280 <wsp:Policy xmlns:wsp="...">
1281   (
1282     <sp:RequireDerivedKeys ... /> |
1283     <sp:RequireImpliedDerivedKeys ... /> |
1284     <sp:RequireExplicitDerivedKeys ... />
1285   ) ?
1286   <sp:RequireExternalUriReference ... /> ?
1287   <sp:SC13SecurityContextToken ... /> ?
1288   <sp:MustNotSendCancel ... /> ?
1289   <sp:MustNotSendAmend ... /> ?
1290   <sp:MustNotSendRenew ... /> ?
1291   <sp:BootstrapPolicy ... >
1292     <wsp:Policy> ... </wsp:Policy>
1293   </sp:BootstrapPolicy> ?
1294 </wsp:Policy>
1295   ...
1296 </sp:SecureConversationToken>
  
```

1297

1298 The following describes the attributes and elements listed in the schema outlined above:

1299 /sp:SecureConversationToken

1300         This identifies a SecureConversationToken assertion.

1301 /sp:SecureConversationToken/@sp:IncludeToken

1302         This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1303 /sp:SecureConversationToken/sp:Issuer

1304         This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
 1305         for the Security Context Token.

1306 /sp:SecureConversationToken/sp:IssuerName

1307         This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
 1308         sp:SecureConversationToken issuer.

1309 /sp:SpnegoContextToken/wst:Claims  
1310 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1311 order to satisfy the token assertion requirements.

1312 /sp:SecureConversationToken/wsp:Policy  
1313 This REQUIRED element identifies additional requirements for use of the  
1314 sp:SecureConversationToken assertion.

1315 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys  
1316 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1317 and [Implied Derived Keys] properties for this token to 'true'.

1318 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
1319 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1320 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1321 'false'.

1322 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
1323 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1324 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1325 'false'.

1326 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference  
1327 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
1328 REQUIRED when referencing this token.

1329 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken  
1330 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should  
1331 be used as obtained using the protocol defined in [[WS-SecureConversation](#)].

1332 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel  
1333 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
1334 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it  
1335 means that SCT/Cancel RST messages are supported by the STS.

1336 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend  
1337 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
1338 conversation token does not support SCT/Amend RST messages. If this assertion is missing it  
1339 means that SCT/Amend RST messages are supported by the STS.

1340 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew  
1341 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
1342 conversation token does not support SCT/Renew RST messages. If this assertion is missing it  
1343 means that SCT/Renew RST messages are supported by the STS.

1344 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy  
1345 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements  
1346 for obtaining the Security Context Token.

1347 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy  
1348 This element contains the security binding requirements for obtaining the Security Context Token.  
1349 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with  
1350 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that  
1351 are to be protected.

1352 **Example**



```

1353 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1354   <sp:SymmetricBinding>
1355     <wsp:Policy>
1356       <sp:ProtectionToken>
1357         <wsp:Policy>
1358           <sp:SecureConversationToken>
1359             <sp:Issuer>
1360               <wsa:Address>http://example.org/sts</wsa:Address>
1361             </sp:Issuer>
1362           <wsp:Policy>
1363             <sp:SC13SecurityContextToken />
1364           <sp:BootstrapPolicy>
1365             <wsp:Policy>
1366               <sp:AsymmetricBinding>
1367                 <wsp:Policy>
1368                   <sp:InitiatorToken>
1369                     ...
1370                   </sp:InitiatorToken>
1371                   <sp:RecipientToken>
1372                     ...
1373                   </sp:RecipientToken>
1374                 </wsp:Policy>
1375               </sp:AsymmetricBinding>
1376             <sp:SignedParts>
1377               ...
1378             </sp:SignedParts>
1379             ...
1380           </wsp:Policy>
1381         </sp:BootstrapPolicy>
1382       </wsp:Policy>
1383     </sp:SecureConversationToken>
1384   </wsp:Policy>
1385 </sp:ProtectionToken>
1386   ...
1387 </wsp:Policy>
1388 </sp:SymmetricBinding>
1389 <sp:SignedParts>
1390   ...
1391 </sp:SignedParts>
1392   ...
1393 </wsp:Policy>

```

## 1394 5.4.8 SamlToken Assertion

1395 This element represents a requirement for a SAML token.

### 1396 Syntax

```

1397 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1398   (
1399     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1400     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1401   ) ?
1402   <wst:Claims Dialect="..."> ... </wst:Claims> ?

```

```

1403 <wsp:Policy xmlns:wsp="...">
1404   (
1405     <sp:RequireDerivedKeys ... /> |
1406     <sp:RequireImpliedDerivedKeys ... /> |
1407     <sp:RequireExplicitDerivedKeys ... />
1408   ) ?
1409   <sp:RequireKeyIdentifierReference ... /> ?
1410   (
1411     <sp:WssSamlV11Token10 ... /> |
1412     <sp:WssSamlV11Token11 ... /> |
1413     <sp:WssSamlV20Token11 ... />
1414   ) ?
1415   ...
1416 </wsp:Policy>
1417 ...
1418 </sp:SamlToken>

```

1419

1420 The following describes the attributes and elements listed in the schema outlined above:

1421 /sp:SamlToken

1422 This identifies a SamlToken assertion.

1423 /sp:SamlToken/@sp:IncludeToken

1424 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1425 /sp:SamlToken/sp:Issuer

1426 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1427 of the sp:SamlToken.

1428 /sp:SamlToken/sp:IssuerName

1429 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken  
1430 issuer.

1431 /sp:SamlToken/wst:Claims

1432 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1433 order to satisfy the token assertion requirements.

1434 /sp:SamlToken/wsp:Policy

1435 This REQUIRED element identifies additional requirements for use of the sp:SamlToken  
1436 assertion.

1437 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

1438 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1439 and [Implied Derived Keys] properties for this token to 'true'.

1440 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1441 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1442 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1443 'false'.

1444 /sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1445 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1446 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1447 'false'.

1448 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1449 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1450 REQUIRED when referencing this token.

1451 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10  
1452 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1453 should be used as defined in [WSS:SAMLTOKENPROFILE1.0].

1454 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11  
1455 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1456 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1457 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11  
1458 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token  
1459 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1460  
1461 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties  
1462 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
1463 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion  
1464 SHOULD be used instead.

## 1465 5.4.9 RelToken Assertion

1466 This element represents a requirement for a REL token.

### 1467 Syntax

```
1468 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1469 (   
1470   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1471   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1472 ) ?  
1473 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1474 <wsp:Policy xmlns:wsp="...">  
1475   (   
1476     <sp:RequireDerivedKeys ... /> |  
1477     <sp:RequireImpliedDerivedKeys ... /> |  
1478     <sp:RequireExplicitDerivedKeys ... />  
1479   ) ?  
1480   <sp:RequireKeyIdentifierReference ... /> ?  
1481   (   
1482     <sp:WssRelV10Token10 ... /> |  
1483     <sp:WssRelV20Token10 ... /> |  
1484     <sp:WssRelV10Token11 ... /> |  
1485     <sp:WssRelV20Token11 ... />  
1486   ) ?  
1487   ...  
1488 </wsp:Policy>  
1489   ...  
1490 </sp:RelToken>
```

1491  
1492 The following describes the attributes and elements listed in the schema outlined above:

1493 /sp:RelToken

1494 This identifies a RelToken assertion.

1495 /sp:RelToken/@sp:IncludeToken

1496 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1497 /sp:RelToken/sp:Issuer

1498 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1499 of the sp:RelToken.

1500 /sp:RelToken/sp:IssuerName  
 1501 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken  
 1502 issuer.

1503 /sp:RelToken/wst:Claims  
 1504 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
 1505 order to satisfy the token assertion requirements.

1506 /sp:RelToken/wsp:Policy  
 1507 This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

1508 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys  
 1509 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1510 and [Implied Derived Keys] property for this token to 'true'.

1511 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
 1512 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1513 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1514 'false'.

1515 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1516 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1517 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1518 'false'.

1519 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference  
 1520 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
 1521 REQUIRED when referencing this token.

1522 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10  
 1523 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
 1524 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1525 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10  
 1526 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
 1527 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1528 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11  
 1529 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
 1530 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1531 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11  
 1532 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
 1533 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1534  
 1535 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties  
 1536 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
 1537 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion  
 1538 SHOULD be used instead.

## 1539 **5.4.10 HttpsToken Assertion**

1540 This element represents a requirement for a transport binding to support the use of HTTPS.

### 1541 **Syntax**

```

1542 <sp:HttpsToken xmlns:sp="..." ... >
1543 (
1544   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1545   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1546 ) ?
1547 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1548 <wsp:Policy xmlns:wsp="...">
1549 (
1550   <sp:HttpBasicAuthentication /> |
1551   <sp:HttpDigestAuthentication /> |
1552   <sp:RequireClientCertificate /> |
1553   ...
1554 ) ?
1555   ...
1556 </wsp:Policy>
1557   ...
1558 </sp:HttpsToken>

```

1559 The following describes the attributes and elements listed in the schema outlined above:

1560 /sp:HttpsToken

1561 This identifies an Https assertion stating that use of the HTTPS protocol specification is  
1562 supported.

1563 /sp:HttpsToken/sp:Issuer

1564 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1565 of the sp:HttpsToken.

1566 /sp:HttpsToken/sp:IssuerName

1567 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken  
1568 issuer.

1569 /sp:HttpsToken/wst:Claims

1570 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1571 order to satisfy the token assertion requirements.

1572 /sp:HttpsToken/wsp:Policy

1573 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken  
1574 assertion.

1575 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1576 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic  
1577 Authentication [[RFC2068](#)] to authenticate to the service.

1578 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1579 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP  
1580 Digest Authentication [[RFC2068](#)] to authenticate to the service.

1581 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1582 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a  
1583 certificate when negotiating the HTTPS session.

### 1584 **5.4.11 KeyValueToken Assertion**

1585 This element represents a requirement for a KeyValue token. The next section defines the KeyValue  
1586 security token abstraction for purposes of this token assertion.  
1587

1588 This document defines requirements for KeyValueType when used in combination with RSA  
1589 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by  
1590 introducing new nested assertions besides *sp:RsaKeyValue*.

#### 1591 **Syntax**

```
1592 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1593   <wsp:Policy xmlns:wsp="...">  
1594     <sp:RsaKeyValue ... /> ?  
1595     ...  
1596   </wsp:Policy>  
1597   ...  
1598 </sp:KeyValueToken>
```

1599 The following describes the attributes listed in the schema outlined above:

1600 /sp:KeyValueToken

1601         This identifies a RsaToken assertion.

1602 /sp:KeyValueToken/@sp:IncludeToken

1603         This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1604 /sp:KeyValueToken/wsp:Policy

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

1607 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1608         This OPTIONAL element is a policy assertion that indicates that the ds:RSAKeyValue element  
1609         must be present in the KeyValueType token. This indicates that an RSA key pair must be used.

#### 1610 **5.4.11.1 KeyValueType Token**

1611 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key  
1612 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this  
1613 section is to define the KeyValueType token abstraction that represents such key pair referencing mechanism.

1614  
1615 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be  
1616 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*  
1617 element in combination with RSA cryptographic algorithm.

1618  
1619 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the  
1620 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1621 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1622   <ds:KeyValue>  
1623     <ds:RSAKeyValue>  
1624       <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1625       <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1626     </ds:RSAKeyValue>  
1627   </ds:KeyValue>  
1628 </ds:KeyInfo>
```

1629  
1630 When the KeyValueType token is used the corresponding public key value appears directly in the signature or  
1631 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValueType token  
1632 manifestation outside the *ds:KeyInfo* element.

```
1633 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1634   <SignedInfo>  
1635     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1636     c14n#" />  
1637     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1638     <Reference URI="#_1">  
1639       <Transforms>
```

```

1640     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1641   </Transforms>
1642   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1643   <DigestValue>...</DigestValue>
1644   </Reference>
1645 </SignedInfo>
1646 <SignatureValue>...</SignatureValue>
1647 <KeyInfo>
1648   <KeyValue>
1649     <RSAKeyValue>
1650       <Modulus>...</Modulus>
1651       <Exponent>...</Exponent>
1652     </RSAKeyValue>
1653   </KeyValue>
1654 </KeyInfo>
1655 </Signature>

```

1656  
1657 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no  
1658 identifier can be associated with the token, the KeyValue token cannot be referenced by using  
1659 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the KeyValue  
1660 token can be used whenever a security token can be used as illustrated on the following example:

```

1661 <t:RequestSecurityToken xmlns:t="...">
1662   <t:RequestType>...</t:RequestType>
1663   ...
1664   <t:UseKey>
1665     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1666       <KeyValue>
1667         <RSAKeyValue>
1668           <Modulus>...</Modulus>
1669           <Exponent>...</Exponent>
1670         </RSAKeyValue>
1671       </KeyValue>
1672     </KeyInfo>
1673   </t:UseKey>
1674 </t:RequestSecurityToken>

```

1675

## 6 Security Binding Properties

1676 This section defines the various properties or conditions of a security binding, their semantics, values and  
1677 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are  
1678 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that  
1679 populates a value of a property appears in a policy, that property is set to the value indicated by the  
1680 assertion. The security binding then uses the value of the property to control its behavior. The properties  
1681 listed here are common to the various security bindings described in Section 7. Assertions that define  
1682 values for these properties are defined in Section 7. The following properties are used by the security  
1683 binding assertions.

### 1684 6.1 [Algorithm Suite] Property

1685 This property specifies the algorithm suite REQUIRED for performing cryptographic operations with  
1686 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and  
1687 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This  
1688 property defines the set of available algorithms. The value of this property is typically referenced by a  
1689 security binding and is used to specify the algorithms used for all message level cryptographic operations  
1690 performed under the security binding.

1691 Note: In some cases, this property MAY be referenced under a context other than a security binding and  
1692 used to control the algorithms used under that context. For example, supporting token assertions define  
1693 such a context. In such contexts, the specified algorithms still apply to message level cryptographic  
1694 operations.

1695 An algorithm suite defines values for each of the following operations and properties:

- 1696 • [Sym Sig] Symmetric Key Signature
- 1697 • [Asym Sig] Signature with an asymmetric key
- 1698 • [Dig] Digest
- 1699 • [Enc] Encryption
- 1700 • [Sym KW] Symmetric Key Wrap
- 1701 • [Asym KW] Asymmetric Key Wrap
- 1702 • [Comp Key] Computed key
- 1703 • [Enc KD] Encryption key derivation
- 1704 • [Sig KD] Signature key derivation
- 1705 • [Min SKL] Minimum symmetric key length
- 1706 • [Max SKL] Maximum symmetric key length
- 1707 • [Min AKL] Minimum asymmetric key length
- 1708 • [Max AKL] Maximum asymmetric key length

1709

1710 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>

1711

1712 The tables below show all the base algorithm suites defined by this specification. This table defines  
 1713 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

1714 This table defines additional properties whose values can be specified along with the default value for that  
 1715 property.

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

1716 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

## 1717 6.2 [Timestamp] Property

1718 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`  
1719 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected  
1720 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be  
1721 present. The default value for this property is 'false'.

## 1722 6.3 [Protection Order] Property

1723 This property indicates the order in which integrity and confidentiality are applied to the message, in  
1724 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.

1725 The default value for this property is 'SignBeforeEncrypting'.

## 1726 6.4 [Signature Protection] Property

1727 This boolean property specifies whether the signature MUST be encrypted. If the value is 'true', the  
1728 primary signature MUST be encrypted and any signature confirmation elements MUST also be encrypted.  
1729 The primary signature element is NOT REQUIRED to be encrypted if the value is 'true' when there is  
1730 nothing in the message that is covered by this signature that is encrypted. If the value is 'false', the  
1731 primary signature MUST NOT be encrypted and any signature confirmation elements MUST NOT be  
1732 encrypted. The default value for this property is 'false'.

## 1733 6.5 [Token Protection] Property

1734 This boolean property specifies whether signatures MUST cover the token used to generate that  
1735 signature. If the value is 'true', then each token used to generate a signature MUST be covered by that  
1736 signature. If the value is 'false', then the token MUST NOT be covered by the signature. Note that in  
1737 cases where derived keys are used the 'main' token, and NOT the derived key token, is covered by the  
1738 signature. It is RECOMMENDED that assertions that define values for this property apply to [Endpoint  
1739 Policy Subject]. The default value for this property is 'false'.

## 1740 6.6 [Entire Header and Body Signatures] Property

1741 This boolean property specifies whether signature digests over the SOAP body and SOAP headers  
1742 MUST only cover the entire body and entire header elements. If the value is 'true', then each digest over  
1743 the SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In  
1744 addition each digest over a SOAP header MUST be over an actual header element and not a descendant  
1745 of a header element. This restriction does not specifically apply to the wsse:Security header. However  
1746 signature digests over child elements of the wsse:Security header MUST be over the entire child element  
1747 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a  
1748 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to  
1749 'true' mitigates against some possible re-writing attacks. It is RECOMENDED that assertions that define  
1750 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

## 1751 6.7 [Security Header Layout] Property

1752 This property indicates which layout rules to apply when adding items to the security header. The  
1753 following table shows which rules are defined by this specification.

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

1754

### 1755 6.7.1 Strict Layout Rules for WSS 1.0

- 1756 1. Tokens that are included in the message MUST be declared before use. For example:
- 1757 a. A local signing token MUST occur before the signature that uses it.
- 1758 b. A local token serving as the source token for a derived key token MUST occur before that
- 1759 derived key token.
- 1760 c. A local encryption token MUST occur before the reference list that points to
- 1761 xenc:EncryptedData elements that use it.
- 1762 d. If the same token is used for both signing and encryption, then it SHOULD appear before
- 1763 the ds:Signature and xenc:ReferenceList elements in the security header that are
- 1764 generated using the token.
- 1765 2. Signed elements inside the security header MUST occur before the signature that signs them.
- 1766 For example:
- 1767 a. A timestamp MUST occur before the signature that signs it.

- 1768           b. A Username token (usually in encrypted form) MUST occur before the signature that  
1769           signs it.
- 1770           c. A primary signature MUST occur before the supporting token signature that signs the  
1771           primary signature's signature value element.
- 1772       3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1773       has the same order requirements as the source plain text element, unless requirement 4  
1774       indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1775       supporting token signature per 2.c above and an encrypted token has the same ordering  
1776       requirements as the unencrypted token.

1777       If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top  
1778       level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the  
1779       security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any  
1780       xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict  
1781       Layout Rules for WSS 1.1

- 1782       1. Tokens that are included in the message MUST be declared before use. For example:
- 1783           a. A local signing token MUST occur before the signature that uses it.
- 1784           b. A local token serving as the source token for a derived key token MUST occur before that  
1785           derived key token.
- 1786           c. A local encryption token MUST occur before the reference list that points to  
1787           xenc:EncryptedData elements that use it.
- 1788           d. If the same token is used for both signing and encryption, then it SHOULD appear before  
1789           the ds:Signature and xenc:ReferenceList elements in the security header that are  
1790           generated using the token.
- 1791       2. Signed elements inside the security header MUST occur before the signature that signs them.  
1792       For example:
- 1793           a. A timestamp MUST occur before the signature that signs it.
- 1794           b. A Username token (usually in encrypted form) MUST occur before the signature that  
1795           signs it.
- 1796           c. A primary signature MUST occur before the supporting token signature that signs the  
1797           primary signature's signature value element.
- 1798           d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1799       3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1800       has the same order requirements as the source plain text element, unless requirement 4  
1801       indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1802       supporting token signature per 2.c above and an encrypted token has the same ordering  
1803       requirements as the unencrypted token.
- 1804       4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element  
1805       MUST be present in the security header. The xenc:ReferenceList MUST occur before any  
1806       xenc:EncryptedData elements in the security header that are referenced from the reference list.  
1807       However, the xenc:ReferenceList is NOT REQUIRED to appear before independently encrypted  
1808       tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1809       5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)  
1810       1.1] MUST obey rule 1 above.

---

## 1811 7 Security Binding Assertions

1812 The appropriate representation of the different facets of security mechanisms requires distilling the  
1813 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy  
1814 scope of assertions defined in this section is the policy scope of their containing element.

### 1815 7.1 AlgorithmSuite Assertion

1816 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]  
1817 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

#### 1818 Syntax

```
1819 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1820   <wsp:Policy xmlns:wsp="...">  
1821     (<sp:Basic256 ... /> |  
1822     <sp:Basic192 ... /> |  
1823     <sp:Basic128 ... /> |  
1824     <sp:TripleDes ... /> |  
1825     <sp:Basic256Rsa15 ... /> |  
1826     <sp:Basic192Rsa15 ... /> |  
1827     <sp:Basic128Rsa15 ... /> |  
1828     <sp:TripleDesRsa15 ... /> |  
1829     <sp:Basic256Sha256 ... /> |  
1830     <sp:Basic192Sha256 ... /> |  
1831     <sp:Basic128Sha256 ... /> |  
1832     <sp:TripleDesSha256 ... /> |  
1833     <sp:Basic256Sha256Rsa15 ... /> |  
1834     <sp:Basic192Sha256Rsa15 ... /> |  
1835     <sp:Basic128Sha256Rsa15 ... /> |  
1836     <sp:TripleDesSha256Rsa15 ... /> |  
1837     ...)  
1838     <sp:InclusiveC14N ... /> ?  
1839     <sp:SOAPNormalization10 ... /> ?  
1840     <sp:STRTransform10 ... /> ?  
1841     (<sp:XPath10 ... /> |  
1842     <sp:XPathFilter20 ... /> |  
1843     <sp:AbsXPath ... /> |  
1844     ...)?  
1845     ...  
1846   </wsp:Policy>  
1847   ...  
1848 </sp:AlgorithmSuite>
```

1849  
1850 The following describes the attributes and elements listed in the schema outlined above:

1851 /sp:AlgorithmSuite

1852       This identifies an AlgorithmSuite assertion.

1853 /sp:AlgorithmSuite/wsp:Policy

1854       This REQUIRED element contains one or more policy assertions that indicate the specific  
1855       algorithm suite to use.

1856 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1857       This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1858       set to 'Basic256'.

1859 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

1860 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1861 set to 'Basic192'.

1862 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1863 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1864 set to 'Basic128'.

1865 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1866 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1867 set to 'TripleDes'.

1868 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1869 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1870 set to 'Basic256Rsa15'.

1871 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1872 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1873 set to 'Basic192Rsa15'.

1874 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1875 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1876 set to 'Basic128Rsa15'.

1877 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1878 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1879 set to 'TripleDesRsa15'.

1880 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1881 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1882 set to 'Basic256Sha256'.

1883 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1884 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1885 set to 'Basic192Sha256'.

1886 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1887 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1888 set to 'Basic128Sha256'.

1889 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1890 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1891 set to 'TripleDesSha256'.

1892 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1893 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1894 set to 'Basic256Sha256Rsa15'.

1895 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1896 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1897 set to 'Basic192Sha256Rsa15'.

1898 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1899 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1900 set to 'Basic128Sha256Rsa15'.

1901 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1902 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1903 set to 'TripleDesSha256Rsa15'.

1904 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1905 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an  
1906 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]  
1907 property is 'ExcC14N'.

1908 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1909 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set  
1910 to 'SNT'.

1911 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1912 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is  
1913 set to 'STRT10'.

1914 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1915 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1916 'XPath'.

1917 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1918 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1919 'XPath20'.

1920 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1921 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1922 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1923

## 1924 7.2 Layout Assertion

1925 This assertion indicates a requirement for a particular security header layout as defined under the  
1926 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its  
1927 containing assertion.

### 1928 Syntax

```
1929 <sp:Layout xmlns:sp="..." ... >  
1930   <wsp:Policy xmlns:wsp="...">  
1931     <sp:Strict ... /> |  
1932     <sp:Lax ... /> |  
1933     <sp:LaxTsFirst ... /> |  
1934     <sp:LaxTsLast ... /> |  
1935     ...  
1936   </wsp:Policy>  
1937   ...  
1938 </sp:Layout>
```

1939

1940 The following describes the attributes and elements listed in the schema outlined above:

1941 /sp:Layout

1942 This identifies a Layout assertion.

1943 /sp:Layout/wsp:Policy

1944 This REQUIRED element contains one or more policy assertions that indicate the specific security  
1945 header layout to use.

1946 /sp:Layout/wsp:Policy/sp:Strict

1947 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1948 property is set to 'Strict'.

1949 /sp:Layout/wsp:Policy/sp:Lax

1950 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1951 property is set to 'Lax'.

1952 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1953 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1954 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to  
1955 'true' by the presence of an sp:IncludeTimestamp assertion.

1956 /sp:Layout/wsp:Policy/sp:LaxTsLast

1957 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1958 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to  
1959 'true' by the presence of an sp:IncludeTimestamp assertion.

## 1960 7.3 TransportBinding Assertion

1961 The TransportBinding assertion is used in scenarios in which message protection and security correlation  
1962 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like  
1963 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by  
1964 the transport. This binding has one binding specific token property; [Transport Token]. This assertion  
1965 MUST apply to [Endpoint Policy Subject].

### 1966 Syntax

```
1967 <sp:TransportBinding xmlns:sp="..." ... >  
1968   <wsp:Policy xmlns:wsp="...">  
1969     <sp:TransportToken ... >  
1970       <wsp:Policy> ... </wsp:Policy>  
1971       ...  
1972     </sp:TransportToken>  
1973     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
1974     <sp:Layout ... > ... </sp:Layout> ?  
1975     <sp:IncludeTimestamp ... /> ?  
1976     ...  
1977   </wsp:Policy>  
1978   ...  
1979 </sp:TransportBinding>
```

1980

1981 The following describes the attributes and elements listed in the schema outlined above:

1982 /sp:TransportBinding

1983 This identifies a TransportBinding assertion.

1984 /sp:TransportBinding/wsp:Policy

1985 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding  
1986 assertion.

1987 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1988 This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.  
1989 The specified token populates the [Transport Token] property and indicates how the transport is  
1990 secured.

1991 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1992 This indicates a nested policy that identifies the type of Transport Token to use.



- 1993 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite
- 1994 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm
- 1995 Suite] property. See Section 6.1 for more details.
- 1996 /sp:TransportBinding/wsp:Policy/sp:Layout
- 1997 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security
- 1998 Header Layout] property. See Section 6.7 for more details.
- 1999 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp
- 2000 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set
- 2001 to 'true'.

## 2002 7.4 SymmetricBinding Assertion

2003 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means

2004 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;

2005 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this

2006 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to

2007 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to

2008 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token

2009 properties and is used as the basis for both encryption and signature in both directions. This assertion

2010 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

### 2011 Syntax

```

2012 <sp:SymmetricBinding xmlns:sp="..." ... >
2013   <wsp:Policy xmlns:wsp="...">
2014     (
2015       <sp:EncryptionToken ... >
2016         <wsp:Policy> ... </wsp:Policy>
2017       </sp:EncryptionToken>
2018       <sp:SignatureToken ... >
2019         <wsp:Policy> ... </wsp:Policy>
2020       </sp:SignatureToken>
2021     ) | (
2022       <sp:ProtectionToken ... >
2023         <wsp:Policy> ... </wsp:Policy>
2024       </sp:ProtectionToken>
2025     )
2026     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2027     <sp:Layout ... > ... </sp:Layout> ?
2028     <sp:IncludeTimestamp ... /> ?
2029     <sp:EncryptBeforeSigning ... /> ?
2030     <sp:EncryptSignature ... /> ?
2031     <sp:ProtectTokens ... /> ?
2032     <sp:OnlySignEntireHeadersAndBody ... /> ?
2033     ...
2034   </wsp:Policy>
2035   ...
2036 </sp:SymmetricBinding>

```

2037

2038 The following describes the attributes and elements listed in the schema outlined above:

- 2039 /sp:SymmetricBinding
- 2040 This identifies a SymmetricBinding assertion.
- 2041 /sp:SymmetricBinding/wsp:Policy
- 2042 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
- 2043 assertion.

2044 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken  
2045 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption  
2046 Token. The specified token populates the [Encryption Token] property and is used for encryption.  
2047 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

2048 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy  
2049 The policy contained here MUST identify exactly one token to use for encryption.

2050 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken  
2051 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.  
2052 The specified token populates the [Signature Token] property and is used for the message  
2053 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be  
2054 specified.

2055 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy  
2056 The policy contained here MUST identify exactly one token to use for signatures.

2057 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken  
2058 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.  
2059 The specified token populates the [Encryption Token] and [Signature Token properties] and is  
2060 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken  
2061 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be  
2062 specified.

2063 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy  
2064 The policy contained here MUST identify exactly one token to use for protection.

2065 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite  
2066 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2067 Suite] property. See Section 6.1 for more details.

2068 /sp:SymmetricBinding/wsp:Policy/sp:Layout  
2069 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2070 Header Layout] property. See Section 6.7 for more details.

2071 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
2072 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2073 to 'true'.

2074 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning  
2075 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
2076 set to 'EncryptBeforeSigning'.

2077 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature  
2078 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
2079 property is set to 'true'.

2080 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens  
2081 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
2082 set to 'true'.

2083 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
2084 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
2085 Signatures] property is set to 'true'.

## 2086 7.5 AsymmetricBinding Assertion

2087 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means  
2088 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly  
2089 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and  
2090 signature. However it is also common practice to use distinct keys for encryption and signature, because  
2091 of their different lifecycles.

2092

2093 This binding enables either of these practices by means of four binding specific token properties: [Initiator  
2094 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption  
2095 Token].

2096

2097 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator  
2098 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient  
2099 Encryption Token] will both refer to the same token.

2100

2101 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator  
2102 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient  
2103 Encryption Token] will refer to different tokens.

2104

2105 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the  
2106 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response  
2107 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the  
2108 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for  
2109 the message encryption from initiator to the recipient. Note that in each case, the token is associated with  
2110 the party (initiator or recipient) who knows the secret.

2111 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy  
2112 Subject].

### 2113 Syntax

```
2114 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2115   <wsp:Policy xmlns:wsp="...">  
2116     (  
2117       <sp:InitiatorToken>  
2118         <wsp:Policy> ... </wsp:Policy>  
2119       </sp:InitiatorToken>  
2120     ) | (  
2121       <sp:InitiatorSignatureToken>  
2122         <wsp:Policy> ... </wsp:Policy>  
2123       </sp:InitiatorSignatureToken>  
2124       <sp:InitiatorEncryptionToken>  
2125         <wsp:Policy> ... </wsp:Policy>  
2126       </sp:InitiatorEncryptionToken>  
2127     )  
2128     (  
2129       <sp:RecipientToken>  
2130         <wsp:Policy> ... </wsp:Policy>  
2131       </sp:RecipientToken>  
2132     ) | (  
2133       <sp:RecipientSignatureToken>  
2134         <wsp:Policy> ... </wsp:Policy>  
2135       </sp:RecipientSignatureToken>  
2136       <sp:RecipientEncryptionToken>  
2137         <wsp:Policy> ... </wsp:Policy>
```

```

2138     </sp:RecipientEncryptionToken>
2139   )
2140   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2141   <sp:Layout ... > ... </sp:Layout> ?
2142   <sp:IncludeTimestamp ... /> ?
2143   <sp:EncryptBeforeSigning ... /> ?
2144   <sp:EncryptSignature ... /> ?
2145   <sp:ProtectTokens ... /> ?
2146   <sp:OnlySignEntireHeadersAndBody ... /> ?
2147   ...
2148 </wsp:Policy>
2149   ...
2150 </sp:AsymmetricBinding>

```

2151

2152 The following describes the attributes and elements listed in the schema outlined above:

2153 /sp:AsymmetricBinding

2154 This identifies a AsymmetricBinding assertion.

2155 /sp:AsymmetricBinding/wsp:Policy

2156 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding  
2157 assertion.

2158 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2159 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.  
2160 The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]  
2161 properties and is used for the message signature from initiator to recipient, and encryption from  
2162 recipient to initiator.

2163 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2164 The policy contained here MUST identify one or more token assertions.

2165 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2166 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2167 Signature Token. The specified token populates the [Initiator Signature Token] property and is  
2168 used for the message signature from initiator to recipient.

2169 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2170 The policy contained here MUST identify one or more token assertions.

2171 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2172 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2173 Encryption Token. The specified token populates the [Initiator Encryption Token] property and is  
2174 used for the message encryption from recipient to initiator.

2175 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2176 The policy contained here MUST identify one or more token assertions.

2177 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2178 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.  
2179 The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]  
2180 property and is used for encryption from initiator to recipient, and for the message signature from  
2181 recipient to initiator.

2182 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2183 The policy contained here MUST identify one or more token assertions.

2184 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2185 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
2186 Signature Token. The specified token populates the [Recipient Signature Token] property and is  
2187 used for the message signature from recipient to initiator.

2188 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy  
2189 The policy contained here MUST identify one or more token assertions.

2190 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken  
2191 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
2192 Encryption Token. The specified token populates the [Recipient Encryption Token] property and  
2193 is used for the message encryption from initiator to recipient.

2194 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy  
2195 The policy contained here MUST identify one or more token assertions.

2196 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite  
2197 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2198 Suite] property. See Section 6.1 for more details.

2199 /sp:AsymmetricBinding/wsp:Policy/sp:Layout  
2200 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2201 Header Layout] property. See Section 6.7 for more details.

2202 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
2203 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2204 to 'true'.

2205 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning  
2206 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
2207 set to 'EncryptBeforeSigning'.

2208 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature  
2209 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
2210 property is set to 'true'.

2211 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens  
2212 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
2213 set to 'true'.

2214 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
2215 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
2216 Signatures] property is set to 'true'.

2217

## 8 Supporting Tokens

2218 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to  
2219 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore  
2220 be referred to as the “message signature”. In case of Transport Binding the message is signed outside of  
2221 the message XML by the underlying transport protocol and the signature itself is not part of the message.  
2222 Additional tokens MAY be specified to augment the claims provided by the token associated with the  
2223 “message signature” provided by the Security Binding. This section defines seven properties related to  
2224 supporting token requirements which MAY be referenced by a Security Binding: [Supporting Tokens],  
2225 [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens],  
2226 [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing  
2227 Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties:  
2228 SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,  
2229 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,  
2230 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These  
2231 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy  
2232 Subject] or [Operation Policy Subject].

2233

2234 Supporting tokens MAY be specified at a different scope than the binding assertion which provides  
2235 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while  
2236 the supporting tokens might be defined at the scope of a message. When assertions that populate this  
2237 property are defined in overlapping scopes, the sender SHOULD merge the requirements by including all  
2238 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

2239

2240 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the  
2241 tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements  
2242 (tokens, signatures, reference lists etc.) in the security header would be used to determine which order  
2243 signature and encryptions occurred in.

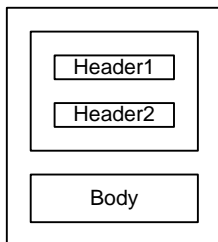
2244

2245 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional  
2246 constraints defined by the supporting token assertion. For example, if the supporting token assertion  
2247 specifies message parts that need to be encrypted, the specified tokens need to be capable of  
2248 encryption.

2249

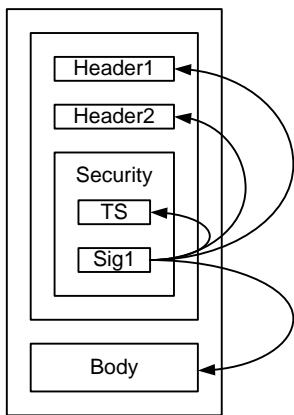
2250 To illustrate the different ways that supporting tokens MAY be bound to the message, let’s consider a  
2251 message with three components: Header1, Header2, and Body.

2252

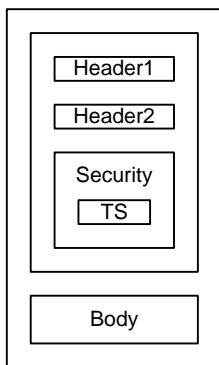


2253

2254 Even before any supporting tokens are added, each binding requires that the message is signed using a  
 2255 token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important  
 2256 parts of the message including the message timestamp (TS) facilitate replay detection. The signature is  
 2257 then included as part of the Security header as illustrated below:  
 2258



2259  
 2260 Note: if REQUIRED, the initiator may also include in the Security header the token used as the basis for  
 2261 the message signature (Sig1), not shown in the diagram.  
 2262 If transport security is used, only the message timestamp (TS) is included in the Security header as  
 2263 illustrated below. The “message signature” is provided by the underlying transport protocol and is not part  
 2264 of the message XML.



2265  
 2266 **8.1 SupportingTokens Assertion**

2267 Supporting tokens are included in the security header and MAY OPTIONALLY include additional  
 2268 message parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and  
 2269 do not require any protection (signature or encryption) to be applied to the message before they are  
 2270 added. More specifically there is no requirement on “message signature” being present before the  
 2271 supporting tokens are added. However it is RECOMMENDED to employ underlying protection  
 2272 mechanism to ensure that the supporting tokens are cryptographically bound to the message during the  
 2273 transmission.

2274 **Syntax**

```
2275 <sp:SupportingTokens xmlns:sp="..." ... >
2276   <wsp:Policy xmlns:wsp="...">
2277     [Token Assertion]+
2278     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2279     (
2280       <sp:SignedParts ... > ... </sp:SignedParts> |
```

2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288

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

2289

2290 The following describes the attributes and elements listed in the schema outlined above:

2291

/sp:SupportingTokens

2292

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

2293

2294

/sp:SupportingTokens/wsp:Policy

2295

This describes additional requirements for satisfying the SupportingTokens assertion.

2296

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

2297

The policy MUST identify one or more token assertions.

2298

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

2299

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

2300

2301

2302

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

2303

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

2304

2305

2306

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

2307

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

2308

2309

2310

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

2311

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

2312

2313

2314

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

2315

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

2316

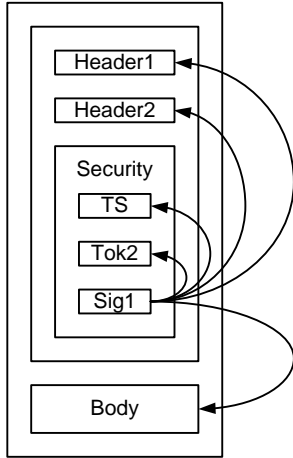
2317

## 2318 8.2 SignedSupportingTokens Assertion

2319 Signed tokens are included in the “message signature” as defined above and MAY OPTIONALLY include  
2320 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token  
2321 (Tok2) is signed by the message signature (Sig1):

2322

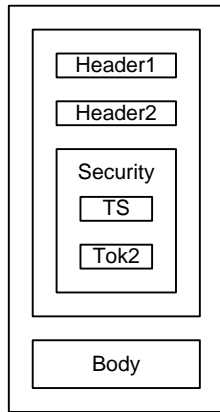




2323

2324 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2325



2326

2327 **Syntax**

```

2328 <sp:SignedSupportingTokens xmlns:sp="..." ... >
2329   <wsp:Policy xmlns:wsp="...">
2330     [Token Assertion]+
2331     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2332     (
2333       <sp:SignedParts ... > ... </sp:SignedParts> |
2334       <sp:SignedElements ... > ... </sp:SignedElements> |
2335       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2336       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2337     ) *
2338     ...
2339   </wsp:Policy>
2340   ...
2341 </sp:SignedSupportingTokens>

```

2342

2343 The following describes the attributes and elements listed in the schema outlined above:

2344 /sp:SignedSupportingTokens

2345 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed  
2346 Supporting Tokens] property.

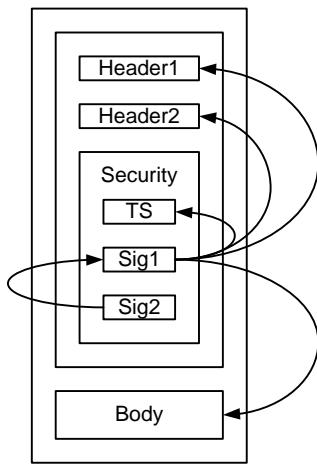
2347 /sp:SignedSupportingTokens/wsp:Policy

2348 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

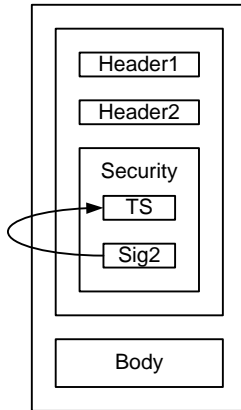
- 2349 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]  
 2350 The policy MUST identify one or more token assertions.
- 2351 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite  
 2352 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
 2353 describes the algorithms to use for cryptographic operations performed with the tokens identified  
 2354 by this policy assertion.
- 2355 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts  
 2356 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
 2357 and describes additional message parts that MUST be included in the signature generated with  
 2358 the token identified by this policy assertion.
- 2359 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements  
 2360 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
 2361 and describes additional message elements that MUST be included in the signature generated  
 2362 with the token identified by this policy assertion.
- 2363 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts  
 2364 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
 2365 and describes additional message parts that MUST be encrypted using the token identified by  
 2366 this policy assertion.
- 2367 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements  
 2368 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
 2369 and describes additional message elements that MUST be encrypted using the token identified  
 2370 by this policy assertion.

2371 **8.3 EndorsingSupportingTokens Assertion**

2372 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element  
 2373 produced from the message signature and MAY OPTIONALLY include additional message parts to sign  
 2374 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message  
 2375 signature (Sig1):  
 2376



2377  
 2378 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated  
 2379 below:  
 2380



2381

2382 **Syntax**

2383  
2384  
2385  
2386  
2387  
2388  
2389  
2390  
2391  
2392  
2393  
2394  
2395  
2396

```

<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>

```

2397

2398 The following describes the attributes and elements listed in the schema outlined above:

2399 /sp:EndorsingSupportingTokens

2400 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the  
2401 [Endorsing Supporting Tokens] property.

2402 /sp:EndorsingSupportingTokens/wsp:Policy

2403 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2404 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2405 The policy MUST identify one or more token assertions.

2406 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2407 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2408 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2409 by this policy assertion.

2410 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2411 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2412 and describes additional message parts that MUST be included in the signature generated with  
2413 the token identified by this policy assertion.

2414 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2415 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2416 and describes additional message elements that MUST be included in the signature generated  
2417 with the token identified by this policy assertion.

2418 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2419 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2420 and describes additional message parts that MUST be encrypted using the token identified by  
2421 this policy assertion.

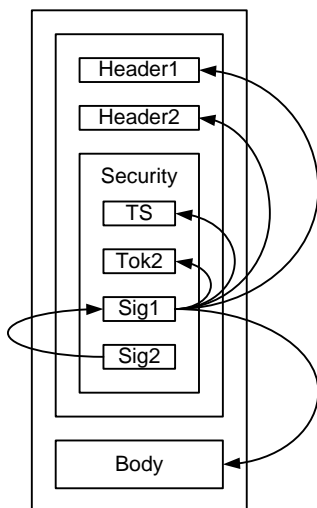
2422 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2423 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2424 and describes additional message elements that MUST be encrypted using the token identified  
2425 by this policy assertion.

## 2426 8.4 SignedEndorsingSupportingTokens Assertion

2427 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature  
2428 and are themselves signed by that message signature, that is both tokens (the token used for the  
2429 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY  
2430 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed  
2431 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the  
2432 message signature (Sig1):

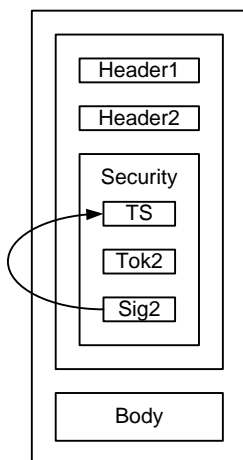
2433



2434

2435 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)  
2436 SHOULD cover the message timestamp as illustrated below:

2437



2438

2439 **Syntax**

```
2440 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2441   <wsp:Policy xmlns:wsp="...">
2442     [Token Assertion]+
2443     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2444     (
2445       <sp:SignedParts ... > ... </sp:SignedParts> |
2446       <sp:SignedElements ... > ... </sp:SignedElements> |
2447       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2448       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2449     ) *
2450     ...
2451   </wsp:Policy>
2452   ...
2453 </sp:SignedEndorsingSupportingTokens>
```

2454

2455 The following describes the attributes and elements listed in the schema outlined above:

2456 /sp:SignedEndorsingSupportingTokens

2457 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the  
2458 [Signed Endorsing Supporting Tokens] property.

2459 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2460 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2461 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2462 The policy MUST identify one or more token assertions.

2463 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2464 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2465 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2466 by this policy assertion.

2467 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2468 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2469 and describes additional message parts that MUST be included in the signature generated with  
2470 the token identified by this policy assertion.

2471 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2472 This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional  
2473 message elements that MUST be included in the signature generated with the token identified by  
2474 this policy assertion.

2475 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2476 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2477 and describes additional message parts that MUST be encrypted using the token identified by  
2478 this policy assertion.

2479 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2480 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2481 and describes additional message elements that MUST be encrypted using the token identified  
2482 by this policy assertion.

## 2483 **8.5 SignedEncryptedSupportingTokens Assertion**

2484 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also  
2485 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2486 encrypting the supporting tokens.

2487 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of  
2488 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2489 sp:SignedSupportingTokens assertion.

## 2490 **8.6 EncryptedSupportingTokens Assertion**

2491 Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in  
2492 the security header and MUST be encrypted when they appear in the security header.  
2493 Element encryption SHOULD be used for encrypting these tokens. The encrypted supporting  
2494 tokens can be added to any SOAP message and do not require the "message signature"  
2495 being present before the encrypted supporting tokens are added.

2496 The syntax for the sp:EncryptedSupportingTokens differs from the syntax of  
2497 sp:SupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2498 sp:SupportingTokens assertion.

2499 The encrypted supporting tokens SHOULD be used only when the sender cannot provide the  
2500 "message signature" and it is RECOMMENDED that the receiver employs some security  
2501 mechanisms external to the message to prevent the spoofing attacks. In all other cases it is  
2502 RECOMMENDED to use signed encrypted supporting tokens instead to ensure that the  
2503 encrypted tokens are cryptographically bound to the message (See section 8.5).

## 2504 **8.7 EndorsingEncryptedSupportingTokens Assertion**

2505 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also  
2506 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2507 encrypting the supporting tokens.

2508 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of  
2509 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2510 sp:EndorsingSupportingTokens assertion.

## 2511 **8.8 SignedEndorsingEncryptedSupportingTokens Assertion**

2512 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section  
2513 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD  
2514 be used for encrypting the supporting tokens.

2515 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of  
2516 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per  
2517 the sp:SignedEndorsingSupportingTokens assertion.

## 2518 **8.9 Interaction between [Token Protection] property and supporting 2519 token assertions**

2520 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that  
2521 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2522 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or  
2523 the [Signature Token] in the Symmetric binding case, covers that token.
- 2524 • Endorsing signatures cover the main signature and the endorsing token.

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

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

## 8.10 Example

Example policy containing supporting token assertions:

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

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

2574

## 9 WSS: SOAP Message Security Options

2575 There are several OPTIONAL aspects to the WSS: SOAP Message Security specification that are  
2576 independent of the trust and token taxonomies. This section describes another class of properties and  
2577 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The  
2578 assertions defined here MUST apply to [Endpoint Policy Subject].

2579 The properties and assertions dealing with token references defined in this section indicate whether the  
2580 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and  
2581 recipient MAY send a fault if such references are encountered.

2582

2583 Note: This approach is chosen because:

2584 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used  
2585 in a single reference.

2586 B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending  
2587 on which of a series of messages is being secured.

2588

2589 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a  
2590 `wsse:InvalidSecurity` fault.

2591

### 2592 WSS: SOAP Message Security 1.0 Properties

#### 2593 **[Direct References]**

2594 This property indicates whether the initiator and recipient MUST be able to process direct token  
2595 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations  
2596 MUST be able to process such references.

2597

#### 2598 **[Key Identifier References]**

2599 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific  
2600 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2601 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST  
2602 NOT generate such references and that the initiator and recipient MAY send a fault if such references are  
2603 encountered. This property has a default value of 'false'.

2604

#### 2605 **[Issuer Serial References]**

2606 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2607 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST  
2608 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT  
2609 generate such references and that the initiator and recipient MAY send a fault if such references are  
2610 encountered. This property has a default value of 'false'.

2611

#### 2612 **[External URI References]**

2613 This boolean property indicates whether the initiator and recipient MUST be able to process references to  
2614 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST  
2615 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT



2616 generate such references and that the initiator and recipient MAY send a fault if such references are  
2617 encountered. This property has a default value of 'false'.

#### 2618 **[Embedded Token References]**

2619 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2620 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to  
2621 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2622 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2623 This property has a default value of 'false'.

2624

### 2625 **WSS: SOAP Message Security 1.1 Properties**

#### 2626 **[Thumbprint References]**

2627 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2628 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to  
2629 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2630 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2631 This property has a default value of 'false'.

2632

#### 2633 **[EncryptedKey References]**

2634 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2635 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2636 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2637 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2638 This property has a default value of 'false'.

2639

#### 2640 **[Signature Confirmation]**

2641 This boolean property specifies whether `wss11:SignatureConfirmation` elements SHOULD be  
2642 used as defined in WSS: Soap Message Security 1.1. If the value is 'true',  
2643 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If  
2644 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property  
2645 applies to all signatures that are included in the security header. This property has a default value of  
2646 'false'.

## 2647 **9.1 Wss10 Assertion**

2648 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are  
2649 supported.

### 2650 **Syntax**

```
2651 <sp:Wss10 xmlns:sp="..." ... >  
2652   <wsp:Policy xmlns:wsp="...">  
2653     <sp:MustSupportRefKeyIdentifier ... /> ?  
2654     <sp:MustSupportRefIssuerSerial ... /> ?  
2655     <sp:MustSupportRefExternalURI ... /> ?  
2656     <sp:MustSupportRefEmbeddedToken ... /> ?  
2657     ...  
2658   </wsp:Policy>  
2659   ...  
2660 </sp:Wss10>
```

2661

2662 The following describes the attributes and elements listed in the schema outlined above:

2663 /sp:Wss10  
 2664 This identifies a WSS10 assertion.

2665 /sp:Wss10/wsp:Policy  
 2666 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2667 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2668 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2669 property is set to 'true'.

2670 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2671 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2672 property is set to 'true'.

2673 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI  
 2674 This OPTIONAL element is a policy assertion indicates that the [External URI References]  
 2675 property is set to 'true'.

2676 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken  
 2677 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
 2678 property is set to 'true'.

## 2679 9.2 Wss11 Assertion

2680 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are  
 2681 supported.

### 2682 Syntax

```

2683 <sp:Wss11 xmlns:sp="..." ... >
2684   <wsp:Policy xmlns:wsp="...">
2685     <sp:MustSupportRefKeyIdentifier ... /> ?
2686     <sp:MustSupportRefIssuerSerial ... /> ?
2687     <sp:MustSupportRefExternalURI ... /> ?
2688     <sp:MustSupportRefEmbeddedToken ... /> ?
2689     <sp:MustSupportRefThumbprint ... /> ?
2690     <sp:MustSupportRefEncryptedKey ... /> ?
2691     <sp:RequireSignatureConfirmation ... /> ?
2692     ...
2693   </wsp:Policy>
2694 </sp:Wss11>
  
```

2695  
 2696 The following describes the attributes and elements listed in the schema outlined above:

2697 /sp:Wss11  
 2698 This identifies an WSS11 assertion.

2699 /sp:Wss11/wsp:Policy  
 2700 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2701 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2702 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2703 property is set to 'true'.

2704 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2705 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2706 property is set to 'true'.

2707 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI  
2708       This OPTIONAL element is a policy assertion indicates that the [External URI References]  
2709       property is set to 'true'.

2710 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken  
2711       This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
2712       property is set to 'true'.

2713 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint  
2714       This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property  
2715       is set to 'true'.

2716 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey  
2717       This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]  
2718       property is set to 'true'.

2719 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation  
2720       This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property  
2721       is set to 'true'.

---

## 2722 10 WS-Trust Options

2723 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically  
2724 with client and server challenges and entropy behaviors. These assertions relate to interactions with a  
2725 Security Token Service and MAY augment the behaviors defined by the Binding Property Assertions  
2726 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2727

### 2728 **WS-Trust 1.3 Properties**

#### 2729 **[Client Challenge]**

2730 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a  
2731 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of  
2732 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of  
2733 messages exchanged by the client and service in satisfying the RST. This property has a default value of  
2734 'false'.

2735

#### 2736 **[Server Challenge]**

2737 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a  
2738 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of  
2739 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server MAY  
2740 increase the number of messages exchanged by the client and service in order to accommodate the  
2741 `wst:SignChallengeResponse` element sent by the client to the server in response to the  
2742 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the  
2743 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2744

#### 2745 **[Client Entropy]**

2746 This boolean property indicates whether client entropy is REQUIRED to be used as key material for a  
2747 requested proof token. A value of 'true' indicates that client entropy is REQUIRED. A value of 'false'  
2748 indicates that client entropy is NOT REQUIRED. This property has a default value of 'false'.

2749

#### 2750 **[Server Entropy]**

2751 This boolean property indicates whether server entropy is REQUIRED to be used as key material for a  
2752 requested proof token. A value of 'true' indicates that server entropy is REQUIRED. A value of 'false'  
2753 indicates that server entropy is NOT REQUIRED. This property has a default value of 'false'.

2754 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy  
2755 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm  
2756 Suite] property.

2757

#### 2758 **[Issued Tokens]**

2759 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in  
2760 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'  
2761 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of  
2762 'false'.

#### 2763 **[Collection]**

2764 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A  
2765 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and  
2766 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that  
2767 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default  
2768 value of 'false'.

2769

### 2770 **[Scope Policy 1.5]**

2771 This boolean property indicates whether the wsp:AppliesTo element in the [WS-Policy] 1.5 namespace is  
2772 supported as described in [WS-Trust]. A value of 'true' indicates that the wsp:AppliesTo element in the  
2773 [WS-Policy] 1.5 namespace is supported. A value of 'false' indicates that the wsp:AppliesTo element in  
2774 the [WS-Policy] 1.5 namespace is not supported, the [WS-Policy] 1.2 namespace is used instead in this  
2775 case. This property has a default value of 'false'.

2776

### 2777 **[Interactive Challenge]**

2778 This boolean property indicates whether interactive challenges are supported. A value of 'true' indicates  
2779 that a wst14:InteractiveChallenge element is supported inside of an RSTR sent by the server to the client.  
2780 A value of 'false' indicates that wst14:InteractiveChallenge is not supported. A challenge issued by the  
2781 server may increase the number of messages exchanged by the client and service in order to  
2782 accommodate the wst14:InteractiveChallengeResponse element sent by the client to the server in  
2783 response to the wst14:InteractiveChallenge element. There is an optimization in which a client MAY send  
2784 the wst14:InteractiveChallengeResponse element in an initial RST to the server. A final RSTR containing  
2785 the issued token will follow subsequent to the server receiving the wst14:InteractiveChallengeResponse  
2786 element. This property has a default value of 'false'.

2787

## 2788 **10.1 Trust13 Assertion**

2789 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.

### 2790 **Syntax**

```
2791 <sp:Trust13 xmlns:sp="..." ... >  
2792   <wsp:Policy xmlns:wsp="...">  
2793     <sp:MustSupportClientChallenge ... />?  
2794     <sp:MustSupportServerChallenge ... />?  
2795     <sp:RequireClientEntropy ... />?  
2796     <sp:RequireServerEntropy ... />?  
2797     <sp:MustSupportIssuedTokens ... />?  
2798     <sp:RequireRequestSecurityTokenCollection />?  
2799     <sp:RequireAppliesTo />?  
2800     <sp13:ScopePolicy15 />?  
2801     <sp13:MustSupportInteractiveChallenge />?  
2802     ...  
2803   </wsp:Policy>  
2804   ...  
2805 </sp:Trust13 ... >
```

2806

2807 The following describes the attributes and elements listed in the schema outlined above:

2808 /sp:Trust13

2809       This identifies a Trust13 assertion.

2810 /sp:Trust13/wsp:Policy

2811       This indicates a policy that controls WS-Trust 1.3 options.

2812 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge  
2813 This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set  
2814 to 'true'.

2815 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge  
2816 This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set  
2817 to 'true'.

2818 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy  
2819 This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to  
2820 'true'.

2821 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy  
2822 This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to  
2823 'true'.

2824 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens  
2825 This OPTIONAL element is a policy assertion indicates that the [Issued Tokens] property is set to  
2826 'true'.

2827 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection  
2828 This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to  
2829 'true'.

2830 /sp:Trust10/wsp:Policy/sp:RequireAppliesTo  
2831 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor  
2832 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2833 [/sp:Trust13/wsp:Policy/sp13:ScopePolicy15](#)  
2834 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5]  
2835 property is set to 'true'.

2836 [/sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge](#)  
2837 This optional element is a policy assertion indicates that the [Interactive Challenge]  
2838 property is set to 'true'.

---

## 2839 11 Guidance on creating new assertions and assertion 2840 extensibility

2841 This non-normative appendix provides guidance for designers of new assertions intended for use with this  
2842 specification.

### 2843 11.1 General Design Points

- 2844 • Prefer Distinct QNames
- 2845 • Parameterize using nested policy where possible.
- 2846 • Parameterize using attributes and/or child elements where necessary.

### 2847 11.2 Detailed Design Guidance

2848 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per  
2849 WS-Policy is performed by matching element QNames. Matching does not take into account attributes  
2850 that are present on the assertion element. Nor does it take into account child elements except for  
2851 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions  
2852 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2853  
2854 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken  
2855 into account. In general, multiple assertions with distinct QNames are preferably to a single assertion that  
2856 uses attributes and/or content to distinguish different cases. For example, given two possible assertion  
2857 designs;

```
2858  
2859 Design 1  
2860  
2861 <A1/>  
2862 <A2/>  
2863 <A3/>  
2864  
2865 Design 2.  
2866  
2867 <A Parameter='1' />  
2868 <A Parameter='2' />  
2869 <A Parameter='3' />  
2870
```

2871 then design 1. would generally be preferred because it allows the policy matching logic to provide more  
2872 accurate matches between policies.

2873  
2874 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct  
2875 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These  
2876 distinct token assertions make policy matching much more useful as less false positives are generated  
2877 when performing policy matching.

2878  
2879 There are cases where using attributes or child elements as parameters in assertion design is  
2880 reasonable. Examples include cases when implementations are expected to understand all the values for  
2881 a given parameter and when encoding the parameter information into the assertion QName would result  
2882 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2883 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken  
2884 attribute and implementations are expected to understand the meaning of all 5 values. If this information  
2885 was encoded into the assertion QNames, each existing token assertion would require five variants, one  
2886 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2887  
2888 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For  
2889 example, the token version assertions defined in Section 5 use such an approach. The overall token type  
2890 assertion is parameterized by the nested token version assertions. Policy matching can use these  
2891 parameters to find matches between policies where the broad token type is support by both parties but  
2892 they might not support the same specific versions.

2893  
2894 Note, when designing assertions for new token types such assertions SHOULD allow the  
2895 sp:IncludeToken attribute and SHOULD allow nested policy.

2896



---

## 2897 **12 Security Considerations**

2898 It is strongly recommended that policies and assertions be signed to prevent tampering.

2899 It is recommended that policies should not be accepted unless they are signed and have an associated  
2900 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely  
2901 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that  
2902 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2903  
2904 It should be noted that the mechanisms described in this document could be secured as part of a SOAP  
2905 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using  
2906 object-specific security mechanisms.

2907  
2908 It is recommended that policies not specify two (or more) SignedSupportingTokens or  
2909 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are  
2910 subject to modification which may be undetectable.

2911  
2912 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest  
2913 of the policy in order to combat certain XML substitution attacks.

---

## 2914 **A. Assertions and WS-PolicyAttachment**

2915 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per  
2916 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical  
2917 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any  
2918 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy  
2919 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

### 2920 **A.1 Endpoint Policy Subject Assertions**

#### 2921 **A.1.1 Security Binding Assertions**

2922 [TransportBinding Assertion](#) (Section 7.3)

2923 [SymmetricBinding Assertion](#) (Section 7.4)

2924 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2925 **A.1.2 Token Assertions**

2926 [SupportingTokens Assertion](#) (Section 8.1)

2927 [SignedSupportingTokens Assertion](#) (Section 8.2)

2928 [EndorsingSupportingTokens Assertion](#) (Section 8.3)

2929 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)

2930 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)

2931 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)

2932 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

#### 2933 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

2934 [Wss10 Assertion](#) (Section 9.1)

#### 2935 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

2936 [Wss11 Assertion](#) (Section 9.2)

#### 2937 **A.1.5 Trust 1.0 Assertions**

2938 [Trust13 Assertion](#) (Section 10.1)

### 2939 **A.2 Operation Policy Subject Assertions**

#### 2940 **A.2.1 Security Binding Assertions**

2941 [SymmetricBinding Assertion](#) (Section 7.4)

2942 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2943 **A.2.2 Supporting Token Assertions**

2944 [SupportingTokens Assertion](#) (Section 8.1)

2945 [SignedSupportingTokens Assertion](#) (Section 8.2)

2946	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2947	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2948	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2949	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2950	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

## 2951 **A.3 Message Policy Subject Assertions**

### 2952 **A.3.1 Supporting Token Assertions**

2953	<a href="#">SupportingTokens Assertion</a>	(Section 8.1)
2954	<a href="#">SignedSupportingTokens Assertion</a>	(Section 8.2)
2955	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2956	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2957	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2958	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2959	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

### 2960 **A.3.2 Protection Assertions**

2961	<a href="#">SignedParts Assertion</a>	(Section 4.1.1)
2962	<a href="#">SignedElements Assertion</a>	(Section 4.1.2)
2963	<a href="#">EncryptedParts Assertion</a>	(Section 4.2.1)
2964	<a href="#">EncryptedElements Assertion</a>	(Section 4.2.2)
2965	<a href="#">ContentEncryptedElements Assertion</a>	(Section 4.2.3)
2966	<a href="#">RequiredElements Assertion</a>	(Section 4.3.1)
2967	<a href="#">RequiredParts Assertion</a>	(Section 4.3.2)

## 2968 **A.4 Assertions With Undefined Policy Subject**

2969 The assertions listed in this section do not have a defined policy subject because they appear nested  
 2970 inside some other assertion which does have a defined policy subject. This list is derived from nested  
 2971 assertions in the specification that have independent sections. It is not a complete list of nested  
 2972 assertions. Many of the assertions previously listed in this appendix as well as the ones below have  
 2973 additional nested assertions.

### 2974 **A.4.1 General Assertions**

2975	<a href="#">AlgorithmSuite Assertion</a>	(Section 7.1)
2976	<a href="#">Layout Assertion</a>	(Section 7.2)

### 2977 **A.4.2 Token Usage Assertions**

2978 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)  
 2979 assertions.

### 2980 **A.4.3 Token Assertions**

2981	<a href="#">UsernameToken Assertion</a>	(Section 5.3.1)
------	---	-----------------

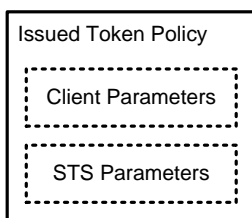
2982	<a href="#">IssuedToken Assertion</a>	(Section 5.3.2)
2983	<a href="#">X509Token Assertion</a>	(Section 5.3.3)
2984	<a href="#">KerberosToken Assertion</a>	(Section 5.3.4)
2985	<a href="#">SpnegoContextToken Assertion</a>	(Section 5.3.5)
2986	<a href="#">SecurityContextToken Assertion</a>	(Section 5.3.6)
2987	<a href="#">SecureConversationToken Assertion</a>	(Section 5.3.7)
2988	<a href="#">SamlToken Assertion</a>	(Section 5.3.8)
2989	<a href="#">RelToken Assertion</a>	(Section 5.3.9)
2990	<a href="#">HttpsToken Assertion</a>	(Section 5.3.10)

## 2991 B. Issued Token Policy

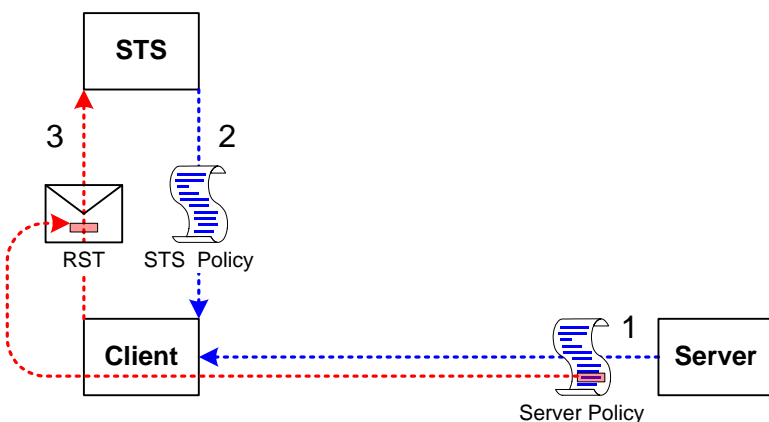
2992 The section provides further detail about behavior associated with the IssuedToken assertion in section  
2993 5.3.2.

2994  
2995 The issued token security model involves a three-party setup. There's a target Server, a Client, and a  
2996 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from  
2997 STS to Client. Policy MAY be embedded inside an Issued Token assertion, or acquired out-of-band.  
2998 There MAY be an explicit trust relationship between the Server and the STS. There MUST be a trust  
2999 relationship between the Client and the STS.

3000  
3001 The Issued Token policy assertion includes two parts: 1) client-specific parameters that MUST be  
3002 understood and processed by the client and 2) STS specific parameters which are to be processed by the  
3003 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



3004  
3005 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server  
3006 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are  
3007 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the  
3008 RST request sent by the Client to the STS as illustrated in the figure below.



3010  
3011 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to  
3012 formulate the RST request and will include any security-specific requirements of the STS.

3013  
3014 The Client MAY augment or replace the contents of the RST made to the STS based on the Client-  
3015 specific parameters received from the Issued Token policy assertion contained in the Server policy, from  
3016 policy it received for the STS, or any other local parameters.

3018 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The  
3019 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along  
3020 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST  
3021 request sent by the Client to the STS following the protocol defined in WS-Trust.

3022

3023 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)  
3024 [Trust\]](#). All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship  
3025 which specifies some or all of the conditions and constraints for issued tokens.

3026

## C. Strict Security Header Layout Examples

3027 The following sections describe the security header layout for specific bindings when applying the 'Strict'  
3028 layout rules defined in Section 6.7.

### 3029 C.1 Transport Binding

3030 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

#### 3031 C.1.1 Policy

3032 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport  
3033 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username  
3034 token attached to the message, and finally an X509 token attached to the message and endorsing the  
3035 message signature. No message protection requirements are described since the transport covers all  
3036 message parts.

```
3037 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3038   <sp:TransportBinding>
3039     <wsp:Policy>
3040       <sp:TransportToken>
3041         <wsp:Policy>
3042           <sp:HttpsToken />
3043         </wsp:Policy>
3044       </sp:TransportToken>
3045       <sp:AlgorithmSuite>
3046         <wsp:Policy>
3047           <sp:Basic256 />
3048         </wsp:Policy>
3049       </sp:AlgorithmSuite>
3050       <sp:Layout>
3051         <wsp:Policy>
3052           <sp:Strict />
3053         </wsp:Policy>
3054       </sp:Layout>
3055       <sp:IncludeTimestamp />
3056     </wsp:Policy>
3057   </sp:TransportBinding>
3058   <sp:SignedSupportingTokens>
3059     <wsp:Policy>
3060       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3061     </wsp:Policy>
3062   </sp:SignedSupportingTokens>
3063   <sp:SignedEndorsingSupportingTokens>
3064     <wsp:Policy>
3065       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3066         <wsp:Policy>
3067           <sp:WssX509v3Token10 />
3068         </wsp:Policy>
3069       </sp:X509Token>
3070     </wsp:Policy>
3071   </sp:SignedEndorsingSupportingTokens>
3072   <sp:Wss11>
3073     <sp:RequireSignatureConfirmation />
3074   </sp:Wss11>
3075 </wsp:Policy>
```

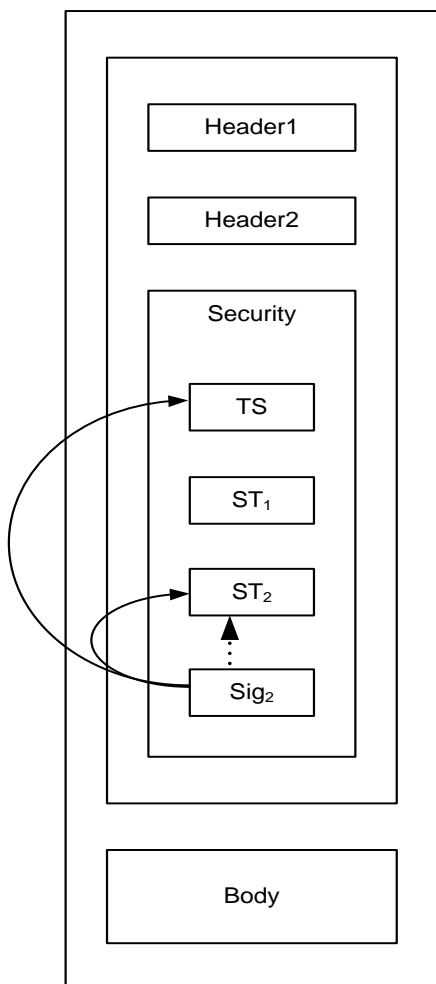
3076 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3077 header layout for this binding.

3078 **C.1.2 Initiator to Recipient Messages**

3079 Messages sent from initiator to recipient have the following layout for the security header:

- 3080 1. A `wsu:Timestamp` element.
- 3081 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 3082 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the  
3083 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1  
3084 above and **SHOULD** cover any other unique identifier for the message in order to prevent  
3085 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If  
3086 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a  
3087 Derived Key Token, based on the supporting token, appears between the supporting token and  
3088 the signature.
- 3089 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each  
3090 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least  
3091 some other unique identifier for the message in order to prevent replays. If [Token Protection] is  
3092 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the  
3093 supporting token is associated with a symmetric key, then a Derived Key Token, based on the  
3094 supporting token, appears before the signature.

3095 The following diagram illustrates the security header layout for the initiator to recipient message:



3096



3097 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The  
3098 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>,  
3099 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST<sub>2</sub>.  
3100 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering  
3101 of the items in the security header follows the most optimal layout for a receiver to process its contents.

3102 *Example:*

3103 Initiator to recipient message

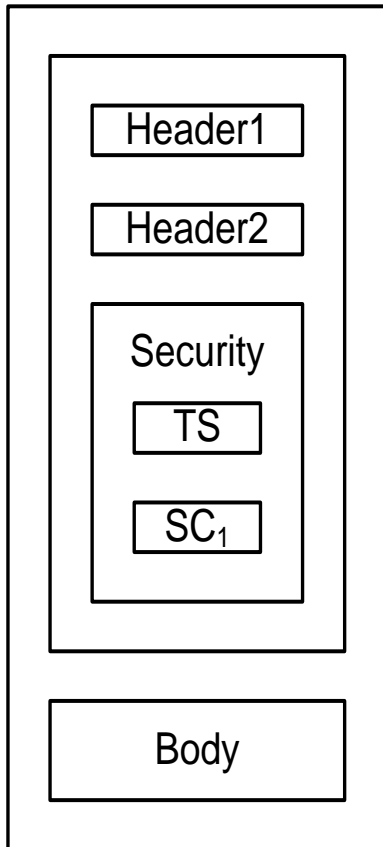
```
3104 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">  
3105 <S:Header>  
3106 ...  
3107 <wsse:Security>  
3108 <wsu:Timestamp wsu:Id="timestamp">  
3109 <wsu:Created>[datetime]</wsu:Created>  
3110 <wsu:Expires>[datetime]</wsu:Expires>  
3111 </wsu:Timestamp>  
3112 <wsse:UsernameToken wsu:Id='SomeSignedToken' >  
3113 ...  
3114 </wsse:UsernameToken>  
3115 <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >  
3116 ...  
3117 </wsse:BinarySecurityToken>  
3118 <ds:Signature>  
3119 <ds:SignedInfo>  
3120 <ds:References>  
3121 <ds:Reference URI="#timestamp" />  
3122 <ds:Reference URI="#SomeSignedEndorsingToken" />  
3123 </ds:References>  
3124 </ds:SignedInfo>  
3125 <ds:SignatureValue>...</ds:SignatureValue>  
3126 <ds:KeyInfo>  
3127 <wsse:SecurityTokenReference>  
3128 <wsse:Reference URI="#SomeSignedEndorsingToken" />  
3129 </wsse:SecurityTokenReference>  
3130 </ds:KeyInfo>  
3131 </ds:Signature>  
3132 ...  
3133 </wsse:Security>  
3134 ...  
3135 </S:Header>  
3136 <S:Body>  
3137 ...  
3138 </S:Body>  
3139 </S:Envelope>
```

### 3140 C.1.3 Recipient to Initiator Messages

3141 Messages sent from recipient to initiator have the following layout for the security header:

- 3142 1. A `wsu:Timestamp` element.
- 3143 2. If the [Signature Confirmation] property has a value of 'true', then a  
3144 `wsse11:SignatureConfirmation` element for each signature in the corresponding message  
3145 sent from initiator to recipient. If there are no signatures in the corresponding message from the  
3146 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no `Value`  
3147 attribute.

3148 The following diagram illustrates the security header layout for the recipient to initiator message:



3149

3150 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One  
 3151 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial  
 3152 message illustrated previously is included. In general, the ordering of the items in the security header  
 3153 follows the most optimal layout for a receiver to process its contents.

3154 *Example:*

3155 Recipient to initiator message

```

3156 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3157   <S:Header>
3158     ...
3159     <wsse:Security>
3160       <wsu:Timestamp wsu:Id="timestamp">
3161         <wsu:Created>[datetime]</wsu:Created>
3162         <wsu:Expires>[datetime]</wsu:Expires>
3163       </wsu:Timestamp>
3164       <wsse11:SignatureConfirmation Value="..." />
3165     ...
3166   </wsse:Security>
3167   ...
3168 </S:Header>
3169 <S:Body>
3170   ...
3171 </S:Body>
3172 </S:Envelope>
  
```

## 3173 C.2 Symmetric Binding

3174 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

## 3175 C.2.1 Policy

3176 The following example shows a policy indicating a Symmetric Binding, a symmetric key based  
3177 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message  
3178 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in  
3179 the message signature and the supporting signatures, a username token attached to the message, and  
3180 finally an X509 token attached to the message and endorsing the message signature. Minimum message  
3181 protection requirements are described as well.

```
3182 <!-- Example Endpoint Policy -->
3183 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3184   <sp:SymmetricBinding>
3185     <wsp:Policy>
3186       <sp:ProtectionToken>
3187         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3188           <sp:Issuer>...</sp:Issuer>
3189           <sp:RequestSecurityTokenTemplate>
3190             ...
3191           </sp:RequestSecurityTokenTemplate>
3192         </sp:IssuedToken>
3193       </sp:ProtectionToken>
3194       <sp:AlgorithmSuite>
3195         <wsp:Policy>
3196           <sp:Basic256 />
3197         </wsp:Policy>
3198       </sp:AlgorithmSuite>
3199       <sp:Layout>
3200         <wsp:Policy>
3201           <sp:Strict />
3202         </wsp:Policy>
3203       </sp:Layout>
3204       <sp:IncludeTimestamp />
3205       <sp:EncryptBeforeSigning />
3206       <sp:EncryptSignature />
3207       <sp:ProtectTokens />
3208     </wsp:Policy>
3209   </sp:SymmetricBinding>
3210   <sp:SignedSupportingTokens>
3211     <wsp:Policy>
3212       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3213     </wsp:Policy>
3214   </sp:SignedSupportingTokens>
3215   <sp:SignedEndorsingSupportingTokens>
3216     <wsp:Policy>
3217       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3218         <wsp:Policy>
3219           <sp:WssX509v3Token10 />
3220         </wsp:Policy>
3221       </sp:X509Token>
3222     </wsp:Policy>
3223   </sp:SignedEndorsingSupportingTokens>
3224   <sp:Wss11>
3225     <wsp:Policy>
3226       <sp:RequireSignatureConfirmation />
3227     </wsp:Policy>
3228   </sp:Wss11>
3229 </wsp:Policy>
3230
```

```

3231 <!-- Example Message Policy -->
3232 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3233   <sp:SignedParts>
3234     <sp:Header Name="Header1" Namespace="..." />
3235     <sp:Header Name="Header2" Namespace="..." />
3236     <sp:Body/>
3237   </sp:SignedParts>
3238   <sp:EncryptedParts>
3239     <sp:Header Name="Header2" Namespace="..." />
3240     <sp:Body/>
3241   </sp:EncryptedParts>
3242 </wsp:Policy>
3243

```

3244 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3245 header layout for this binding.

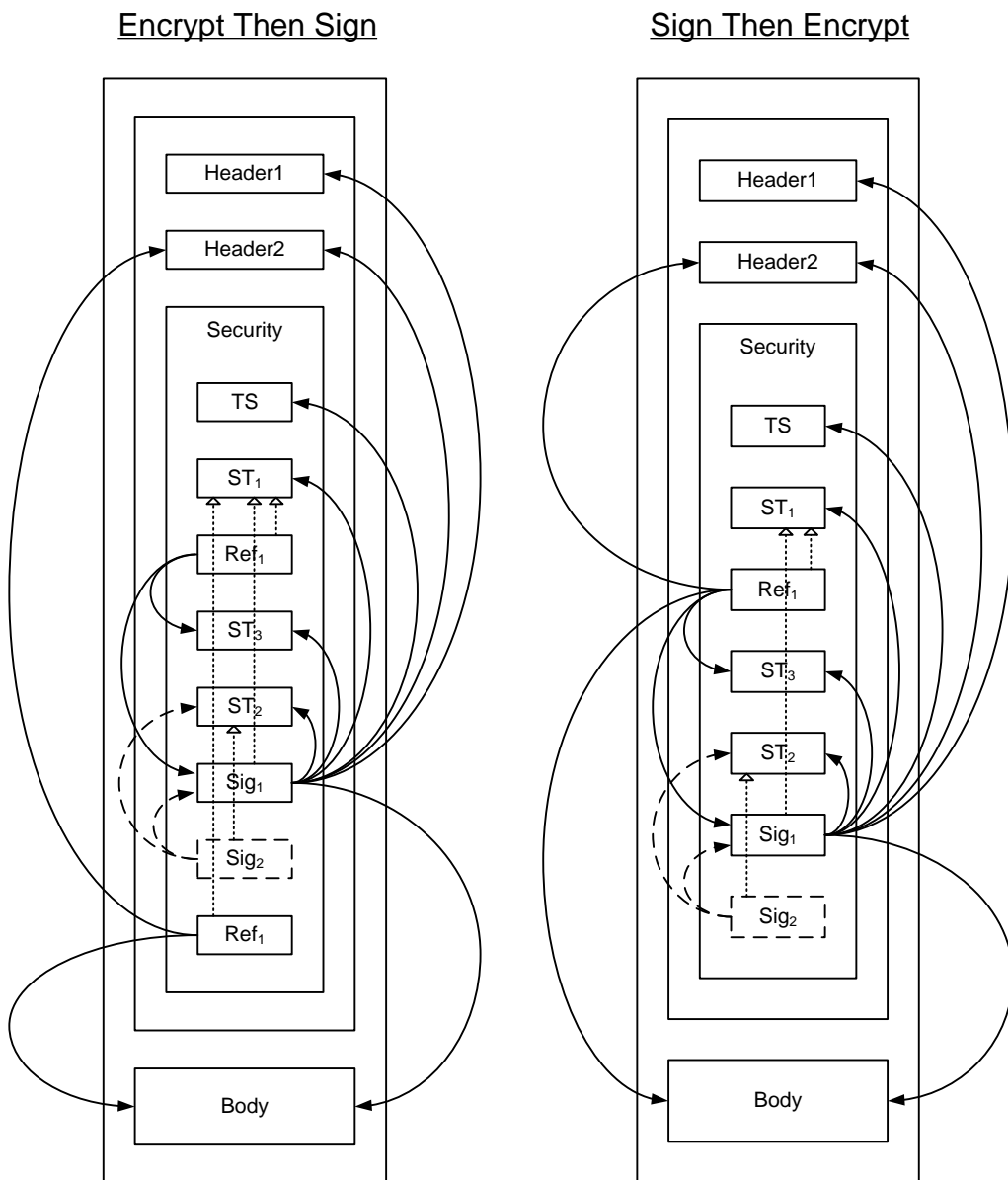
## 3246 C.2.2 Initiator to Recipient Messages

3247 Messages sent from initiator to recipient have the following layout for the security header:

- 3248 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3249 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or  
3250 `.../IncludeToken/Always`, then the [Encryption Token].
- 3251 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3252 Derived Key Token is used for encryption.
- 3253 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3254 reference list MUST include a reference to the message signature. If [Protection Order] is  
3255 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3256 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3257 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3258 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]  
3259 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or  
3260 `.../IncludeToken/Always`.
- 3261 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3262 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the  
3263 [Signature Token].
- 3264 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This  
3265 Derived Key Token is used for signature.
- 3266 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of  
3267 whether they are included in the message, and any message parts specified in SignedParts  
3268 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature  
3269 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in  
3270 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3271 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing  
3272 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]  
3273 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the  
3274 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the  
3275 endorsing token, appears before the signature.
- 3276 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message  
3277 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
3278 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3279 above.

3280

3281 The following diagram illustrates the security header layout for the initiator to recipient message:



3282

3283 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
3284 The dashed arrows on the left from the box labeled Sig<sub>2</sub> indicate the parts signed by the supporting token  
3285 labeled ST<sub>2</sub>, namely the message signature labeled Sig<sub>1</sub> and the token used as the basis for the  
3286 signature labeled ST<sub>2</sub>. The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts  
3287 encrypted using a key based on the Shared Secret Token labeled ST<sub>1</sub>. The dotted arrows inside the box  
3288 labeled Security indicate the token that was used as the basis for each cryptographic operation. In  
3289 general, the ordering of the items in the security header follows the most optimal layout for a receiver to  
3290 process its contents.

3291 *Example:*

3292 Initiator to recipient message using EncryptBeforeSigning:

```
3293 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3294   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3295   xmlns:xenc="..." xmlns:ds="...">
3296   <S:Header>
3297     <x:Header1 wsu:Id="Header1" >
3298       ...
3299     </x:Header1>
3300
```

```

3301 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3302   <!-- Plaintext Header2
3303   <x:Header2 wsu:Id="Header2" >
3304     ...
3305   </x:Header2>
3306   -->
3307   ...
3308 </wsse1:EncryptedHeader>
3309 ...
3310 <wsse:Security>
3311   <wsu:Timestamp wsu:Id="Timestamp">
3312     <wsu:Created>...</wsu:Created>
3313     <wsu:Expires>...</wsu:Expires>
3314   </wsu:Timestamp>
3315   <saml:Assertion AssertionId="_SharedSecretToken" ...>
3316     ...
3317   </saml:Assertion>
3318   <xenc:ReferenceList>
3319     <xenc:DataReference URI="#enc_Signature" />
3320     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3321     ...
3322   </xenc:ReferenceList>
3323   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3324     <!-- Plaintext UsernameToken
3325     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3326       ...
3327     </wsse:UsernameToken>
3328     -->
3329     ...
3330     <ds:KeyInfo>
3331       <wsse:SecurityTokenReference>
3332         <wsse:Reference URI="#_SharedSecretToken" />
3333       </wsse:SecurityTokenReference>
3334     </ds:KeyInfo>
3335   </xenc:EncryptedData>
3336   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3337     ...
3338   </wsse:BinarySecurityToken>
3339   <xenc:EncryptedData ID="enc_Signature">
3340     <!-- Plaintext Signature
3341     <ds:Signature Id="Signature">
3342       <ds:SignedInfo>
3343         <ds:References>
3344           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3345           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3346           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3347           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3348           <ds:Reference URI="#Header1" >...</ds:Reference>
3349           <ds:Reference URI="#Header2" >...</ds:Reference>
3350           <ds:Reference URI="#Body" >...</ds:Reference>
3351         </ds:References>
3352       </ds:SignedInfo>
3353     </ds:SignatureValue>...</ds:SignatureValue>
3354     <ds:KeyInfo>
3355       <wsse:SecurityTokenReference>
3356         <wsse:Reference URI="#_SharedSecretToken" />
3357       </wsse:SecurityTokenReference>
3358     </ds:KeyInfo>
3359   </xenc:EncryptedData>
3360   -->
3361   ...
3362   <ds:KeyInfo>
3363     <wsse:SecurityTokenReference>
3364       <wsse:Reference URI="#_SharedSecretToken" />

```

```

3365     </wsse:SecurityTokenReference>
3366     </ds:KeyInfo>
3367 </xenc:EncryptedData>
3368 <ds:Signature>
3369   <ds:SignedInfo>
3370     <ds:References>
3371       <ds:Reference URI="#Signature" >...</ds:Reference>
3372       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3373     </ds:References>
3374   </ds:SignedInfo>
3375 <ds:SignatureValue>...</ds:SignatureValue>
3376 <ds:KeyInfo>
3377   <wsse:SecurityTokenReference>
3378     <wsse:Reference URI="#SomeSupportingToken" />
3379   </wsse:SecurityTokenReference>
3380 </ds:KeyInfo>
3381 </ds:Signature>
3382 <xenc:ReferenceList>
3383   <xenc:DataReference URI="#enc_Body" />
3384   <xenc:DataReference URI="#enc_Header2" />
3385   ...
3386 </xenc:ReferenceList>
3387 </wsse:Security>
3388 </S:Header>
3389 <S:Body wsu:Id="Body">
3390   <xenc:EncryptedData Id="enc_Body">
3391     ...
3392     <ds:KeyInfo>
3393       <wsse:SecurityTokenReference>
3394         <wsse:Reference URI="#_SharedSecretToken" />
3395       </wsse:SecurityTokenReference>
3396     </ds:KeyInfo>
3397   </xenc:EncryptedData>
3398 </S:Body>
3399 </S:Envelope>

```

### 3400 C.2.3 Recipient to Initiator Messages

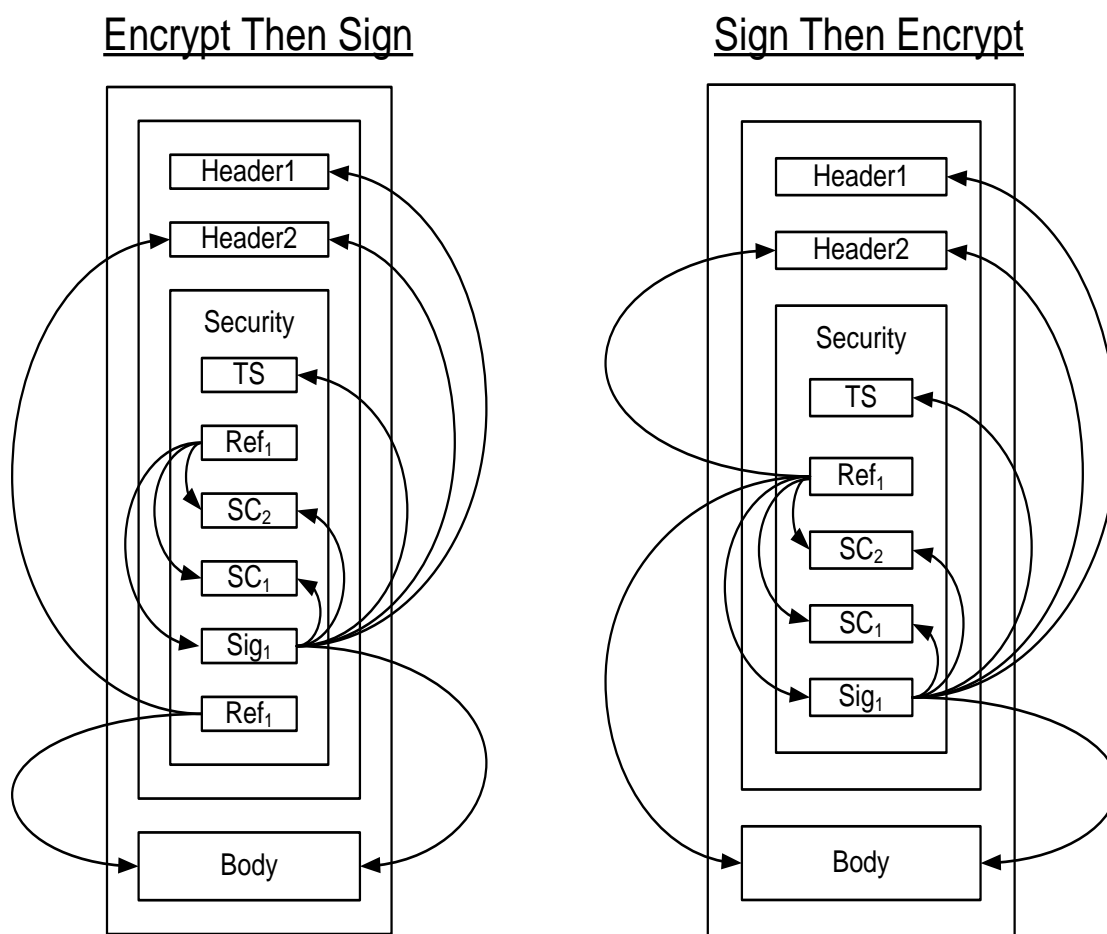
3401 Messages send from recipient to initiator have the following layout for the security header:

- 3402 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3403 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the  
3404 [Encryption Token].
- 3405 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3406 Derived Key Token is used for encryption.
- 3407 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3408 reference list MUST include a reference to the message signature from 6 below, and the  
3409 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is  
3410 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3411 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3412 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3413 above.
- 3414 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each  
3415 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3416 in the corresponding message from the initiator to the recipient, then a  
3417 `wss11:SignatureConfirmation` element with no Value attribute.
- 3418 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3419 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].



- 3420 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This  
 3421 Derived Key Token is used for signature.
- 3422 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation  
 3423 elements from 5 above, and all the message parts specified in SignedParts assertions in the  
 3424 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]  
 3425 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token  
 3426 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 3427 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message  
 3428 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
 3429 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption  
 3430 Token].

3431 The following diagram illustrates the security header layout for the recipient to initiator message:



3432

3433 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3434 The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts encrypted using a key based  
 3435 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two  
 3436 wssell:SignatureConfirmation elements labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures  
 3437 in the initial message illustrated previously is included. In general, the ordering of the items in the security  
 3438 header follows the most optimal layout for a receiver to process its contents. The rules used to determine  
 3439 this ordering are described in Appendix C.

3440 *Example:*

3441 Recipient to initiator message using EncryptBeforeSigning:

```
3442 <S:Envelope>
3443   <S:Header>
3444     <x:Header1 wsu:Id="Header1" >
3445       ...
3446     </x:Header1>
3447     <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3448       <!-- Plaintext Header2
3449       <x:Header2 wsu:Id="Header2" >
3450         ...
3451       </x:Header2>
3452       -->
3453       ...
3454     </wsse11:EncryptedHeader>
3455     ...
3456     <wsse:Security>
3457       <wsu:Timestamp wsu:Id="Timestamp">
3458         <wsu:Created>...</wsu:Created>
3459         <wsu:Expires>...</wsu:Expires>
3460       </wsu:Timestamp>
3461       <xenc:ReferenceList>
3462         <xenc:DataReference URI="#enc_Signature" />
3463         <xenc:DataReference URI="#enc_SigConf1" />
3464         <xenc:DataReference URI="#enc_SigConf2" />
3465         ...
3466       </xenc:ReferenceList>
3467       <xenc:EncryptedData ID="enc_SigConf1" >
3468         <!-- Plaintext SignatureConfirmation
3469         <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3470           ...
3471         </wsse11:SignatureConfirmation>
3472         -->
3473         ...
3474       </xenc:EncryptedData>
3475       <xenc:EncryptedData ID="enc_SigConf2" >
3476         <!-- Plaintext SignatureConfirmation
3477         <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3478           ...
3479         </wsse11:SignatureConfirmation>
3480         -->
3481         ...
3482       </xenc:EncryptedData>
```

```

3483
3484 <xenc:EncryptedData Id="enc_Signature">
3485   <!-- Plaintext Signature
3486   <ds:Signature Id="Signature">
3487     <ds:SignedInfo>
3488       <ds:References>
3489         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3490         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3491         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3492         <ds:Reference URI="#Header1" >...</ds:Reference>
3493         <ds:Reference URI="#Header2" >...</ds:Reference>
3494         <ds:Reference URI="#Body" >...</ds:Reference>
3495       </ds:References>
3496     </ds:SignedInfo>
3497     <ds:SignatureValue>...</ds:SignatureValue>
3498     <ds:KeyInfo>
3499       <wsse:SecurityTokenReference>
3500         <wsse:Reference URI="#_SomeIssuedToken" />
3501       </wsse:SecurityTokenReference>
3502     </ds:KeyInfo>
3503   </ds:Signature>
3504   -->
3505 </xenc:EncryptedData>
3506   ...
3507 <ds:KeyInfo>
3508   <wsse:SecurityTokenReference>
3509     <wsse:Reference URI="#_SomeIssuedToken" />
3510   </wsse:SecurityTokenReference>
3511 </ds:KeyInfo>
3512 <xenc:EncryptedData>
3513 <xenc:ReferenceList>
3514   <xenc:DataReference URI="#enc_Body" />
3515   <xenc:DataReference URI="#enc_Header2" />
3516   ...
3517 </xenc:ReferenceList>
3518 </xenc:EncryptedData>
3519 </wsse:Security>
3520 </S:Header>
3521 <S:Body wsu:Id="Body">
3522   <xenc:EncryptedData Id="enc_Body">
3523     ...
3524     <ds:KeyInfo>
3525       <wsse:SecurityTokenReference>
3526         <wsse:Reference URI="#_SomeIssuedToken" />
3527       </wsse:SecurityTokenReference>
3528     </ds:KeyInfo>
3529   </xenc:EncryptedData>
3530 </S:Body>
3531 </S:Envelope>

```

## 3532 C.3 Asymmetric Binding

3533 This section describes how the ‘Strict’ security header layout rules apply to the Asymmetric Binding.

### 3534 C.3.1 Policy

3535 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator  
3536 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the  
3537 message parts before signing, a requirement to encrypt the message signature, a requirement to include  
3538 tokens in the message signature and the supporting signatures, a requirement to include  
3539 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3540 an X509 token attached to the message and endorsing the message signature. Minimum message  
3541 protection requirements are described as well.

```
3542 <!-- Example Endpoint Policy -->
3543 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3544   <sp:AsymmetricBinding>
3545     <wsp:Policy>
3546       <sp:RecipientToken>
3547         <wsp:Policy>
3548           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3549         </wsp:Policy>
3550       </sp:RecipientToken>
3551       <sp:InitiatorToken>
3552         <wsp:Policy>
3553           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3554         </wsp:Policy>
3555       </sp:InitiatorToken>
3556       <sp:AlgorithmSuite>
3557         <wsp:Policy>
3558           <sp:Basic256 />
3559         </wsp:Policy>
3560       </sp:AlgorithmSuite>
3561       <sp:Layout>
3562         <wsp:Policy>
3563           <sp:Strict />
3564         </wsp:Policy>
3565       </sp:Layout>
3566       <sp:IncludeTimestamp />
3567       <sp:EncryptBeforeSigning />
3568       <sp:EncryptSignature />
3569       <sp:ProtectTokens />
3570     </wsp:Policy>
3571   </sp:AsymmetricBinding>
3572   <sp:SignedEncryptedSupportingTokens>
3573     <wsp:Policy>
3574       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3575     </wsp:Policy>
3576   </sp:SignedEncryptedSupportingTokens>
3577   <sp:SignedEndorsingSupportingTokens>
3578     <wsp:Policy>
3579       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3580         <wsp:Policy>
3581           <sp:WssX509v3Token10 />
3582         </wsp:Policy>
3583       </sp:X509Token>
3584     </wsp:Policy>
3585   </sp:SignedEndorsingSupportingTokens>
3586   <sp:Wss11>
3587     <wsp:Policy>
3588       <sp:RequireSignatureConfirmation />
3589     </wsp:Policy>
3590   </sp:Wss11>
3591 </wsp:Policy>
3592
```

3593

```
3594 <!-- Example Message Policy -->
3595 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3596   <sp:SignedParts>
3597     <sp:Header Name="Header1" Namespace="..." />
3598     <sp:Header Name="Header2" Namespace="..." />
3599     <sp:Body/>
3600   </sp:SignedParts>
3601   <sp:EncryptedParts>
3602     <sp:Header Name="Header2" Namespace="..." />
3603     <sp:Body/>
3604   </sp:EncryptedParts>
3605 </wsp:All>
```

3606

3607 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3608 header layout for this binding.

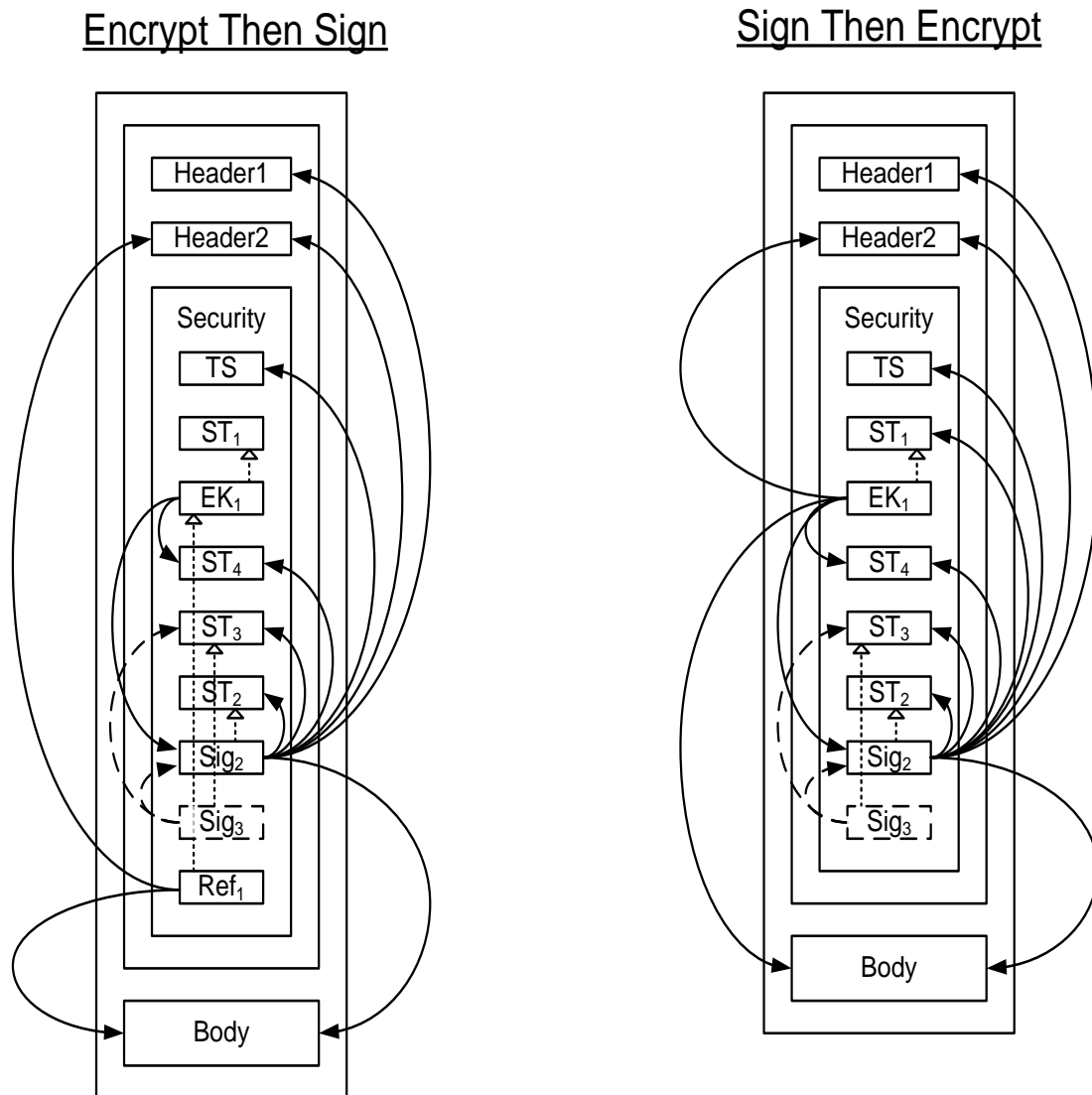
### 3609 **C.3.2 Initiator to Recipient Messages**

3610 Messages sent from initiator to recipient have the following layout:

- 3611 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3612 2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3613 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
- 3614 3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3615 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3616 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3617 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If  
3618 [Signature Protection] is 'true' then the reference list MUST contain a reference to the message  
3619 signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message  
3620 signature.
- 3621 4. Any tokens from the supporting tokens properties (as defined in section 8) whose  
3622 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3623 5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3624 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
- 3625 6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from  
3626 1 above, any tokens from 4 above regardless of whether they are included in the message, and  
3627 any message parts specified in `SignedParts` assertions in the policy. If [Token Protection] is 'true',  
3628 the signature MUST also cover the [Initiator Token] regardless of whether it is included in the  
3629 message.
- 3630 7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting  
3631 Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a  
3632 symmetric key, then a Derived Key Token, based on the supporting token, appears before the  
3633 signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token  
3634 regardless of whether it is included in the message.
- 3635 8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
3636 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
3637 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
3638 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions  
3639 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
3640 element from 3 above.

3641

3642 The following diagram illustrates the security header layout for the initiator to recipient messages:



3643  
 3644 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3645 using the [Initiator Token] labeled ST<sub>2</sub>. The dashed arrows on the left from the box labeled Sig<sub>3</sub> indicate  
 3646 the parts signed by the supporting token ST<sub>3</sub>, namely the message signature Sig<sub>2</sub> and the token used as  
 3647 the basis for the signature labeled ST<sub>3</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate references  
 3648 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on the left  
 3649 from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained in the  
 3650 encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token used as  
 3651 the basis for each cryptographic operation. In general, the ordering of the items in the security header  
 3652 follows the most optimal layout for a receiver to process its contents. The rules used to determine this  
 3653 ordering are described in Appendix C.

3654  
 3655 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted  
 3656 key contains an external reference to the token containing the encryption key. The diagram illustrates  
 3657 how one might attach a security token related to the encrypted key for completeness. One possible use-

3658 case for this approach might be a stack which does not support the STR Dereferencing Transform, but  
3659 wishes to include the encryption token in the message signature.

3660 Initiator to recipient message *Example*

3661 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3662     xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3663 <S:Header>
3664   <x:Header1 wsu:Id="Header1" >
3665     ...
3666   </x:Header1>
3667   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3668     <!-- Plaintext Header2
3669     <x:Header2 wsu:Id="Header2" >
3670       ...
3671     </x:Header2>
3672     -->
3673     ...
3674   </wssell1:EncryptedHeader>
3675   ...
3676   <wsse:Security>
3677     <wsu:Timestamp wsu:Id="Timestamp">
3678       <wsu:Created>...</wsu:Created>
3679       <wsu:Expires>...</wsu:Expires>
3680     </wsu:Timestamp>
3681     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3682       ...
3683     </wsse:BinarySecurityToken>
3684     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3685       ...
3686     <xenc:ReferenceList>
3687       <xenc:DataReference URI="#enc_Signature" />
3688       <xenc:DataReference URI="#enc_SomeUsernameToken" />
3689       ...
3690     </xenc:ReferenceList>
3691   </xenc:EncryptedKey>
3692   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3693     <!-- Plaintext UsernameToken
3694     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3695       ...
3696     </wsse:UsernameToken>
3697     -->
3698     ...
3699   </xenc:EncryptedData>
3700   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3701     ...
3702   </wsse:BinarySecurityToken>
3703   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3704     ...
3705   </wsse:BinarySecurityToken>
3706   <xenc:EncryptedData ID="enc_Signature">
3707     <!-- Plaintext Signature
3708     <ds:Signature Id="Signature">
3709       <ds:SignedInfo>
3710         <ds:References>
3711           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3712           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3713           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3714           <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3715           <ds:Reference URI="#Header1" >...</ds:Reference>
3716           <ds:Reference URI="#Header2" >...</ds:Reference>
3717           <ds:Reference URI="#Body" >...</ds:Reference>
3718         </ds:References>
3719       </ds:SignedInfo>
3720     <ds:SignatureValue>...</ds:SignatureValue>
3721     <ds:KeyInfo>
3722       <wsse:SecurityTokenReference>
3723         <wsse:Reference URI="#InitiatorToken" />
3724       </wsse:SecurityTokenReference>
3725     </ds:KeyInfo>

```



```

3726     </ds:Signature>
3727     -->
3728     ...
3729 </xenc:EncryptedData>
3730 <ds:Signature>
3731   <ds:SignedInfo>
3732     <ds:References>
3733       <ds:Reference URI="#Signature" >...</ds:Reference>
3734       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3735     </ds:References>
3736   </ds:SignedInfo>
3737   <ds:SignatureValue>...</ds:SignatureValue>
3738   <ds:KeyInfo>
3739     <wsse:SecurityTokenReference>
3740       <wsse:Reference URI="#SomeSupportingToken" />
3741     </wsse:SecurityTokenReference>
3742   </ds:KeyInfo>
3743 </ds:Signature>
3744 <xenc:ReferenceList>
3745   <xenc:DataReference URI="#enc_Body" />
3746   <xenc:DataReference URI="#enc_Header2" />
3747   ...
3748 </xenc:ReferenceList>
3749 </wsse:Security>
3750 </S:Header>
3751 <S:Body wsu:Id="Body">
3752   <xenc:EncryptedData Id="enc_Body">
3753     ...
3754     <ds:KeyInfo>
3755       <wsse:SecurityTokenReference>
3756         <wsse:Reference URI="#RecipientEncryptedKey" />
3757       </wsse:SecurityTokenReference>
3758     </ds:KeyInfo>
3759   </xenc:EncryptedData>
3760 </S:Body>
3761 </S:Envelope>

```

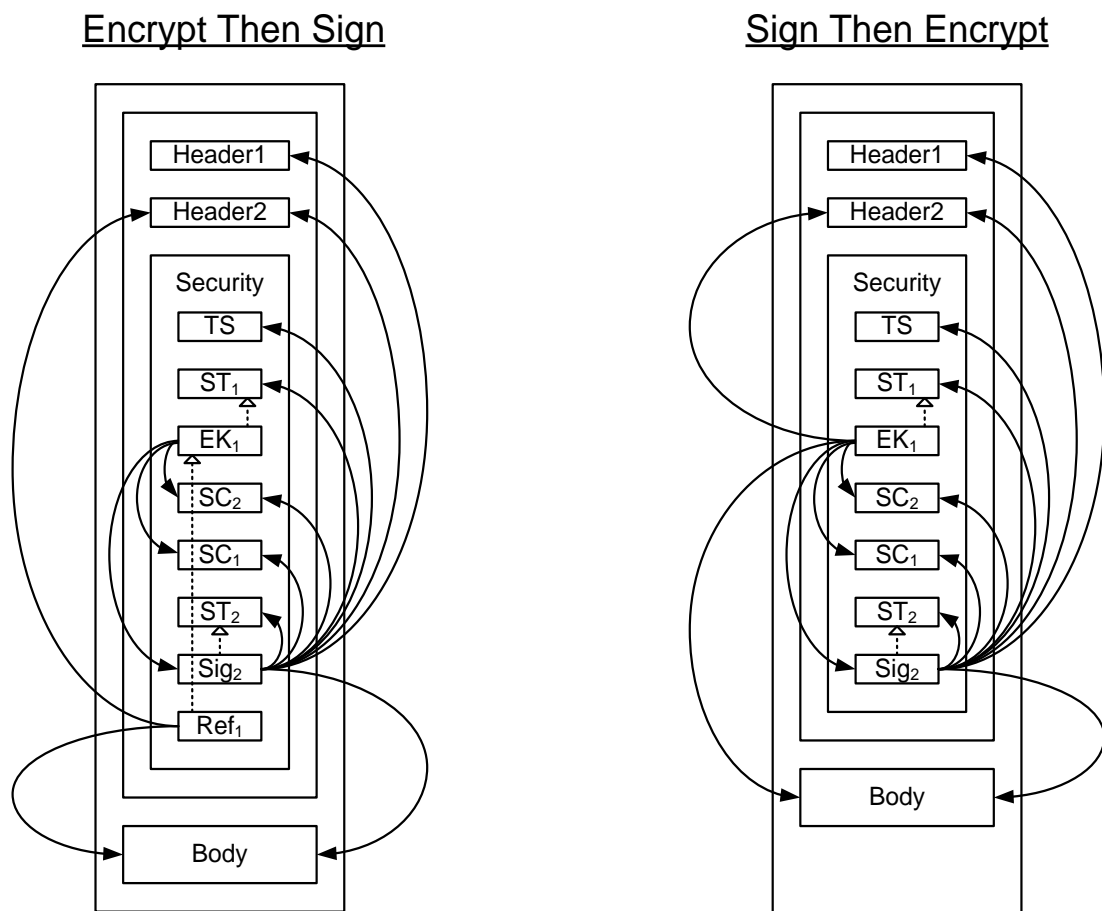
### 3762 C.3.3 Recipient to Initiator Messages

3763 Messages sent from recipient to initiator have the following layout:

- 3764 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3765 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3766 `.../IncludeToken/Always`, then the [Initiator Token].
- 3767 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3768 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3769 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3770 reference to all the message parts specified in EncryptedParts assertions in the policy. If  
3771 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the  
3772 message signature from 6 below, if any and references to the  
3773 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3774 4. If [Signature Confirmation] is 'true', then a `wss11:SignatureConfirmation` element for each  
3775 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3776 in the corresponding message from the initiator to the recipient, then a  
3777 `wss11:SignatureConfirmation` element with no Value attribute.
- 3778 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3779 `.../IncludeToken/Always`, then the [Recipient Token].

- 3780 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],  
 3781 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements  
 3782 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token  
 3783 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3784 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
 3785 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
 3786 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
 3787 reference list includes a reference to all the message parts specified in EncryptedParts assertions  
 3788 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
 3789 element from 3 above.

3790  
 3791 The following diagram illustrates the security header layout for the recipient to initiator messages:



3792

3793 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3794 using the [Recipient Token] labeled ST<sub>2</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate  
 3795 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on  
 3796 the left from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained  
 3797 in the encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token  
 3798 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements  
 3799 labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures in the initial message illustrated previously is  
 3800 included. In general, the ordering of the items in the security header follows the most optimal layout for a  
 3801 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.  
 3802 Recipient to initiator message *Example*:

```

3803 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3804     xmlns:wssell="..." xmlns:wsse="..."
3805     xmlns:xenc="..." xmlns:ds="...">
3806 <S:Header>
3807   <x:Header1 wsu:Id="Header1" >
3808     ...
3809   </x:Header1>
3810   <wssell:EncryptedHeader wsu:Id="enc_Header2">
3811     <!-- Plaintext Header2
3812     <x:Header2 wsu:Id="Header2" >
3813       ...
3814     </x:Header2>
3815     -->
3816     ...
3817   </wssell:EncryptedHeader>
3818   ...
3819 <wsse:Security>
3820   <wsu:Timestamp wsu:Id="Timestamp">
3821     <wsu:Created>...</wsu:Created>
3822     <wsu:Expires>...</wsu:Expires>
3823   </wsu:Timestamp>
3824   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3825     ...
3826   </wsse:BinarySecurityToken>
3827   <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3828     ...
3829     <xenc:ReferenceList>
3830       <xenc:DataReference URI="#enc_Signature" />
3831       <xenc:DataReference URI="#enc_SigConf1" />
3832       <xenc:DataReference URI="#enc_SigConf2" />
3833       ...
3834     </xenc:ReferenceList>
3835   </xenc:EncryptedKey>
3836   <xenc:EncryptedData ID="enc_SigConf2" >
3837     <!-- Plaintext SignatureConfirmation
3838     <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3839     ...
3840     </wssell:SignatureConfirmation>
3841     -->
3842     ...
3843   </xenc:EncryptedData>
3844   <xenc:EncryptedData ID="enc_SigConf1" >
3845     <!-- Plaintext SignatureConfirmation
3846     <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3847     ...
3848     </wssell:SignatureConfirmation>
3849     -->
3850     ...
3851   </xenc:EncryptedData>
3852   <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3853     ...
3854   </wsse:BinarySecurityToken>
3855

```

```

3856 <xenc:EncryptedData ID="enc_Signature">
3857   <!-- Plaintext Signature
3858   <ds:Signature Id="Signature">
3859     <ds:SignedInfo>
3860       <ds:References>
3861         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3862         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3863         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3864         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3865         <ds:Reference URI="#Header1" >...</ds:Reference>
3866         <ds:Reference URI="#Header2" >...</ds:Reference>
3867         <ds:Reference URI="#Body" >...</ds:Reference>
3868       </ds:References>
3869     </ds:SignedInfo>
3870     <ds:SignatureValue>...</ds:SignatureValue>
3871     <ds:KeyInfo>
3872       <wsse:SecurityTokenReference>
3873         <wsse:Reference URI="#RecipientToken" />
3874       </wsse:SecurityTokenReference>
3875     </ds:KeyInfo>
3876   </ds:Signature>
3877   -->
3878   ...
3879 </xenc:EncryptedData>
3880 <xenc:ReferenceList>
3881   <xenc:DataReference URI="#enc_Body" />
3882   <xenc:DataReference URI="#enc_Header2" />
3883   ...
3884 </xenc:ReferenceList>
3885 </wsse:Security>
3886 </S:Header>
3887 <S:Body wsu:Id="Body">
3888   <xenc:EncryptedData Id="enc_Body">
3889     ...
3890     <ds:KeyInfo>
3891       <wsse:SecurityTokenReference>
3892         <wsse:Reference URI="#InitiatorEncryptedKey" />
3893       </wsse:SecurityTokenReference>
3894     </ds:KeyInfo>
3895   </xenc:EncryptedData>
3896 </S:Body>
3897 </S:Envelope>

```

3898  
3899

---

## D. Signed and Encrypted Elements in the Security Header

3900  
3901  
3902  
3903  
3904

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

3905

### D.1 Elements signed by the message signature

3906  
3907  
3908  
3909  
3910  
3911  
3912

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

3913

### D.2 Elements signed by all endorsing signatures

3914  
3915

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

3916

### D.3 Elements signed by a specific endorsing signature

3917  
3918

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

3919

### D.4 Elements that are encrypted

3920  
3921  
3922  
3923  
3924  
3925  
3926

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

3927

---

## E. Acknowledgements

3928 The following individuals have participated in the creation of this specification and are gratefully  
3929 acknowledged:

3930 **Original Authors of the initial contribution:**

3931 Giovanni Della-Libera, Microsoft

3932 Martin Gudgin, Microsoft

3933 Phillip Hallam-Baker, VeriSign

3934 Maryann Hondo, IBM

3935 Hans Granqvist, Verisign

3936 Chris Kaler, Microsoft (editor)

3937 Hiroshi Maruyama, IBM

3938 Michael McIntosh, IBM

3939 Anthony Nadalin, IBM (editor)

3940 Nataraj Nagaratnam, IBM

3941 Rob Philpott, RSA Security

3942 Hemma Prafullchandra, VeriSign

3943 John Shewchuk, Microsoft

3944 Doug Walter, Microsoft

3945 Riaz Zolfonoon, RSA Security

3946

3947 **Original Acknowledgements of the initial contribution:**

3948 Vaithialingam B. Balayoghan, Microsoft

3949 Francisco Curbera, IBM

3950 Christopher Ferris, IBM

3951 Cédric Fournet, Microsoft

3952 Andy Gordon, Microsoft

3953 Tomasz Janczuk, Microsoft

3954 David Melgar, IBM

3955 Mike Perks, IBM

3956 Bruce Rich, IBM

3957 Jeffrey Schlimmer, Microsoft

3958 Chris Sharp, IBM

3959 Kent Tamura, IBM

3960 T.R. Vishwanath, Microsoft

3961 Elliot Waingold, Microsoft

3962

3963 **TC Members during the development of this specification:**

3964 Don Adams, Tibco Software Inc.

3965 Jan Alexander, Microsoft Corporation

3966 Steve Anderson, BMC Software

3967 Donal Arundel, IONA Technologies

3968 Howard Bae, Oracle Corporation

3969 Abbie Barbir, Nortel Networks Limited

3970 Charlton Barreto, Adobe Systems

3971 Mighael Botha, Software AG, Inc.

3972 Toufic Boubez, Layer 7 Technologies Inc.

3973 Norman Brickman, Mitre Corporation

3974 Melissa Brumfield, Booz Allen Hamilton

3975 Lloyd Burch, Novell  
3976 Scott Cantor, Internet2  
3977 Greg Carpenter, Microsoft Corporation  
3978 Steve Carter, Novell  
3979 Symon Chang, BEA Systems, Inc.  
3980 Ching-Yun (C.Y.) Chao, IBM  
3981 Martin Chapman, Oracle Corporation  
3982 Kate Cherry, Lockheed Martin  
3983 Henry (Hyenvui) Chung, IBM  
3984 Luc Clement, Systinet Corp.  
3985 Paul Cotton, Microsoft Corporation  
3986 Glen Daniels, Sonic Software Corp.  
3987 Peter Davis, Neustar, Inc.  
3988 Martijn de Boer, SAP AG  
3989 Werner Dittmann, Siemens AG  
3990 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
3991 Fred Dushin, IONA Technologies  
3992 Petr Dvorak, Systinet Corp.  
3993 Colleen Evans, Microsoft Corporation  
3994 Ruchith Fernando, WSO2  
3995 Mark Fussell, Microsoft Corporation  
3996 Vijay Gajjala, Microsoft Corporation  
3997 Marc Goodner, Microsoft Corporation  
3998 Hans Granqvist, VeriSign  
3999 Martin Gudgin, Microsoft Corporation  
4000 Tony Gullotta, SOA Software Inc.  
4001 Jiandong Guo, Sun Microsystems  
4002 Phillip Hallam-Baker, VeriSign  
4003 Patrick Harding, Ping Identity Corporation  
4004 Heather Hinton, IBM  
4005 Frederick Hirsch, Nokia Corporation  
4006 Jeff Hodges, Neustar, Inc.  
4007 Will Hopkins, BEA Systems, Inc.  
4008 Alex Hristov, Otecia Incorporated  
4009 John Hughes, PA Consulting  
4010 Diane Jordan, IBM  
4011 Venugopal K, Sun Microsystems  
4012 Chris Kaler, Microsoft Corporation  
4013 Dana Kaufman, Forum Systems, Inc.  
4014 Paul Knight, Nortel Networks Limited  
4015 Ramanathan Krishnamurthy, IONA Technologies  
4016 Christopher Kurt, Microsoft Corporation  
4017 Kelvin Lawrence, IBM  
4018 Hubert Le Van Gong, Sun Microsystems  
4019 Jong Lee, BEA Systems, Inc.  
4020 Rich Levinson, Oracle Corporation  
4021 Tommy Lindberg, Dajeil Ltd.  
4022 Mark Little, JBoss Inc.  
4023 Hal Lockhart, BEA Systems, Inc.  
4024 Mike Lyons, Layer 7 Technologies Inc.  
4025 Eve Maler, Sun Microsystems  
4026 Ashok Malhotra, Oracle Corporation  
4027 Anand Mani, CrimsonLogic Pte Ltd  
4028 Jonathan Marsh, Microsoft Corporation  
4029 Robin Martherus, Oracle Corporation  
4030 Miko Matsumura, Infravio, Inc.  
4031 Gary McAfee, IBM

4032 Michael McIntosh, IBM  
4033 John Merrells, Sxip Networks SRL  
4034 Jeff Mischkinsky, Oracle Corporation  
4035 Prateek Mishra, Oracle Corporation  
4036 Bob Morgan, Internet2  
4037 Vamsi Motukuru, Oracle Corporation  
4038 Raajmohan Na, EDS  
4039 Anthony Nadalin, IBM  
4040 Andrew Nash, Reactivity, Inc.  
4041 Eric Newcomer, IONA Technologies  
4042 Duane Nickull, Adobe Systems  
4043 Toshihiro Nishimura, Fujitsu Limited  
4044 Rob Philpott, RSA Security  
4045 Denis Pilipchuk, BEA Systems, Inc.  
4046 Darren Platt, Ping Identity Corporation  
4047 Martin Raepfle, SAP AG  
4048 Nick Ragouzis, Enosis Group LLC  
4049 Prakash Reddy, CA  
4050 Alain Regnier, Ricoh Company, Ltd.  
4051 Irving Reid, Hewlett-Packard  
4052 Bruce Rich, IBM  
4053 Tom Rutt, Fujitsu Limited  
4054 Maneesh Sahu, Actional Corporation  
4055 Frank Siebenlist, Argonne National Laboratory  
4056 Joe Smith, Apani Networks  
4057 Davanum Srinivas, WSO2  
4058 Yakov Sverdlov, CA  
4059 Gene Thurston, AmberPoint  
4060 Victor Valle, IBM  
4061 Asir Vedamuthu, Microsoft Corporation  
4062 Greg Whitehead, Hewlett-Packard  
4063 Ron Williams, IBM  
4064 Corinna Witt, BEA Systems, Inc.  
4065 Kyle Young, Microsoft Corporation  
4066



4067

## F. Revision History

4068

<b>Revision</b>	<b>Date</b>	<b>Editor</b>	<b>Changes Made</b>
0.1	02-01-2008	Marc Goodner	Created from 1.2 errata, ed-03, all changes accepted. i141 – Section 5.4.1 i148 – Section 1.5, 4.1.2 i150 – Section 1.7 i151 – Section 2.1 i152 – Sections 1, 1.2, 1.5, 5.4.2, 10 i153 – Section 10

4069