

# WS-SecurityPolicy 1.3

## OASIS Standard incorporating Approved Errata 01

25 April 2012

### Specification URIs

#### This version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.doc> (Authoritative)  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.html>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.pdf>

#### Previous version:

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

#### Latest version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/ws-securitypolicy-1.3-errata01-complete.doc> (Authoritative)  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/ws-securitypolicy-1.3-errata01-complete.html>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/ws-securitypolicy-1.3-errata01-complete.pdf>

#### Technical Committee:

OASIS Web Services Secure Exchange (WS-SX) TC

#### Chairs:

Kelvin Lawrence ([klawrenc@us.ibm.com](mailto:klawrenc@us.ibm.com)), IBM  
Chris Kaler ([ckaler@microsoft.com](mailto:ckaler@microsoft.com)), Microsoft

#### Editors:

Anthony Nadalin ([tonynad@microsoft.com](mailto:tonynad@microsoft.com)), Microsoft  
Marc Goodner ([mgoodner@microsoft.com](mailto:mgoodner@microsoft.com)), Microsoft  
Martin Gudgin ([mgudgin@microsoft.com](mailto:mgudgin@microsoft.com)), Microsoft  
David Turner ([david.turner@microsoft.com](mailto:david.turner@microsoft.com)), Microsoft  
Abbie Barbir ([abbie.barbir@bankofamerica.com](mailto:abbie.barbir@bankofamerica.com)), Bank of America  
Hans Granqvist ([hgranqvist@verisign.com](mailto:hgranqvist@verisign.com)), VeriSign

#### Additional artifacts:

This OASIS Standard incorporating Approved Errata is one component of a Work Product that also includes:

- *WS-SecurityPolicy 1.3 Errata 01*. 25 April 2012. OASIS Approved Errata. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os.html>.

#### Related work:

This specification is related to:

- *WS-SecurityPolicy* 1.3. 2 February 2009. OASIS Standard. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html>.

**Declared XML namespace:**

<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]. This document incorporates Approved Errata approved by the Technical Committee on 25 April 2012.

**Status:**

This document was last revised or approved by the OASIS Web Services Secure Exchange (WS-SX) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

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>).

**Citation format:**

When referencing this specification the following citation format should be used:

**[WS-SecurityPolicy-1.3]**

*WS-SecurityPolicy* 1.3. 25 April 2012. OASIS Standard incorporating Approved Errata. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.html>.

---

## Notices

Copyright © OASIS Open 2012. All Rights Reserved.

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

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

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

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

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

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

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

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

---

# Table of Contents

1	Introduction.....	7
1.1	Example.....	7
1.2	Namespaces.....	8
1.3	Schema Files.....	9
1.4	Terminology.....	9
1.4.1	Notational Conventions.....	9
1.5	Normative References.....	10
1.6	Non-Normative References.....	13
2	Security Policy Model.....	14
2.1	Security Assertion Model.....	14
2.2	Nested Policy Assertions.....	15
2.3	Security Binding Abstraction.....	15
3	Policy Considerations.....	17
3.1	Nested Policy.....	17
3.2	Policy Subjects.....	17
4	Protection Assertions.....	19
4.1	Integrity Assertions.....	19
4.1.1	SignedParts Assertion.....	19
4.1.2	SignedElements Assertion.....	20
4.2	Confidentiality Assertions.....	21
4.2.1	EncryptedParts Assertion.....	21
4.2.2	EncryptedElements Assertion.....	22
4.2.3	ContentEncryptedElements Assertion.....	23
4.3	Required Elements Assertion.....	23
4.3.1	RequiredElements Assertion.....	24
4.3.2	RequiredParts Assertion.....	24
5	Token Assertions.....	26
5.1	Token Inclusion.....	26
5.1.1	Token Inclusion Values.....	26
5.1.2	Token Inclusion and Token References.....	27
5.2	Token Issuer and Required Claims.....	27
5.2.1	Token Issuer.....	27
5.2.2	Token Issuer Name.....	27
5.2.3	Required Claims.....	27
5.2.4	Processing Rules and Token Matching.....	28
5.3	Token Properties.....	28
5.3.1	[Derived Keys] Property.....	28
5.3.2	[Explicit Derived Keys] Property.....	28
5.3.3	[Implied Derived Keys] Property.....	28
5.4	Token Assertion Types.....	28
5.4.1	UsernameToken Assertion.....	28
5.4.2	ICreatessuedToken Assertion.....	30
5.4.3	X509Token Assertion.....	32

5.4.4	KerberosToken Assertion .....	34
5.4.5	SpnegoContextToken Assertion .....	36
5.4.6	SecurityContextToken Assertion .....	37
5.4.7	SecureConversationToken Assertion .....	38
5.4.8	SamlToken Assertion .....	42
5.4.9	RelToken Assertion .....	44
5.4.10	HttpsToken Assertion .....	45
5.4.11	KeyValueToken Assertion .....	46
6	Security Binding Properties .....	49
6.1	[Algorithm Suite] Property .....	49
6.2	[Timestamp] Property .....	51
6.3	[Protection Order] Property .....	51
6.4	[Signature Protection] Property .....	51
6.5	[Token Protection] Property .....	51
6.6	[Entire Header and Body Signatures] Property .....	52
6.7	[Security Header Layout] Property .....	52
6.7.1	Strict Layout Rules for WSS 1.0 .....	52
7	Security Binding Assertions .....	54
7.1	AlgorithmSuite Assertion .....	54
7.2	Layout Assertion .....	56
7.3	TransportBinding Assertion .....	57
7.4	SymmetricBinding Assertion .....	58
7.5	AsymmetricBinding Assertion .....	60
8	Supporting Tokens .....	63
8.1	SupportingTokens Assertion .....	64
8.2	SignedSupportingTokens Assertion .....	66
8.3	EndorsingSupportingTokens Assertion .....	67
8.4	SignedEndorsingSupportingTokens Assertion .....	69
8.5	SignedEncryptedSupportingTokens Assertion .....	71
8.6	EncryptedSupportingTokens Assertion .....	71
8.7	EndorsingEncryptedSupportingTokens Assertion .....	72
8.8	SignedEndorsingEncryptedSupportingTokens Assertion .....	72
8.9	Interaction between [Token Protection] property and supporting token assertions .....	72
8.10	Example .....	72
9	WSS: SOAP Message Security Options .....	74
9.1	Wss10 Assertion .....	75
9.2	Wss11 Assertion .....	76
10	WS-Trust Options .....	78
10.1	Trust13 Assertion .....	79
11	Guidance on creating new assertions and assertion extensibility .....	81
11.1	General Design Points .....	81
11.2	Detailed Design Guidance .....	81
12	Security Considerations .....	83
13	Conformance .....	84
Appendix A.	Assertions and WS-PolicyAttachment .....	85

A.1 Endpoint Policy Subject Assertions .....	85
A.1.1 Security Binding Assertions .....	85
A.1.2 Token Assertions .....	85
A.1.3 WSS: SOAP Message Security 1.0 Assertions .....	85
A.1.4 WSS: SOAP Message Security 1.1 Assertions .....	85
A.1.5 Trust 1.0 Assertions .....	85
A.2 Operation Policy Subject Assertions .....	85
A.2.1 Security Binding Assertions .....	85
A.2.2 Supporting Token Assertions .....	85
A.3 Message Policy Subject Assertions .....	86
A.3.1 Supporting Token Assertions .....	86
A.3.2 Protection Assertions .....	86
A.4 Assertions With Undefined Policy Subject .....	86
A.4.1 General Assertions .....	86
A.4.2 Token Usage Assertions .....	86
A.4.3 Token Assertions .....	86
Appendix B. Issued Token Policy .....	88
Appendix C. Strict Security Header Layout Examples .....	90
C.1 Transport Binding .....	90
C.1.1 Policy .....	90
C.1.2 Initiator to Recipient Messages .....	91
C.1.3 Recipient to Initiator Messages .....	92
C.2 Symmetric Binding .....	93
C.2.1 Policy .....	94
C.2.2 Initiator to Recipient Messages .....	95
C.2.3 Recipient to Initiator Messages .....	99
C.3 Asymmetric Binding .....	102
C.3.1 Policy .....	102
C.3.2 Initiator to Recipient Messages .....	104
C.3.3 Recipient to Initiator Messages .....	108
Appendix D. Signed and Encrypted Elements in the Security Header .....	112
D.1 Elements signed by the message signature .....	112
D.2 Elements signed by all endorsing signatures .....	112
D.3 Elements signed by a specific endorsing signature .....	112
D.4 Elements that are encrypted .....	112
Appendix E. Acknowledgements .....	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]. Within this specification the use of the namespace prefix wsp refers to the WS-Policy 1.5 namespace. This document takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms and mechanisms used, including using transport level security is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

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

## 1.1 Example

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

Table 1: Example security policy.

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

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

46  
 47 Line 1 in Table 1 indicates that this is a policy statement and that all assertions contained by the  
 48 wsp:Policy element are required to be satisfied. Line 2 indicates the kind of security binding in force. Line  
 49 3 indicates a nested wsp:Policy element which contains assertions that qualify the behavior of the  
 50 SymmetricBinding assertion. Line 4 indicates a ProtectionToken assertion. Line 5 indicates a nested  
 51 wsp:Policy element which contains assertions indicating the type of token to be used for the  
 52 ProtectionToken. Lines 6 to 10 indicate that a Kerberos V5 APREQ token is to be used by both parties in  
 53 a message exchange for protection. Line 13 indicates that signatures are generated over plaintext rather  
 54 than ciphertext. Line 14 indicates that the signature over the signed messages parts is required to be  
 55 encrypted. Lines 17-20 indicate which message parts are to be covered by the primary signature; in this  
 56 case the soap:Body element, indicated by Line 18 and any SOAP headers in the WS-Addressing  
 57 namespace, indicated by line 19. Lines 21-23 indicate which message parts are to be encrypted; in this  
 58 case just the soap:Body element, indicated by Line 22.

## 59 1.2 Namespaces

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

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

63  
 64 Table 2 lists XML namespaces that are used in this specification. The choice of any namespace prefix is  
 65 arbitrary and not semantically significant.

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

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



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

## 67 1.3 Schema Files

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

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

## 72 1.4 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

### 89 1.4.1 Notational Conventions

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

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

- 94 • The syntax appears as an XML instance, but values in italics indicate data types instead of literal  
95 values.
- 96 • Characters are appended to elements and attributes to indicate cardinality:
  - 97 ○ "?" (0 or 1)
  - 98 ○ "\*" (0 or more)
  - 99 ○ "+" (1 or more)
- 100 • The character "|" is used to indicate a choice between alternatives.
- 101 • The characters "(" and ")" are used to indicate that contained items are to be treated as a group  
102 with respect to cardinality or choice.

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

- 111
- 112 Elements and Attributes defined by this specification are referred to in the text of this document using
- 113 XPath 1.0 expressions. Extensibility points are referred to using an extended version of this syntax:
- An element extensibility point is referred to using {any} in place of the element name. This indicates that any element name can be used, from any namespace other than the namespace of this specification.
  - An attribute extensibility point is referred to using @{any} in place of the attribute name. This indicates that any attribute name can be used, from any namespace other than the namespace of this specification.

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

121 In this document reference is made to the wsu:Id attribute and the wsu:Created and wsu:Expires

122 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

124 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp

125 element could reference it (as is done here).

127 WS-SecurityPolicy is designed to work with the general Web Services framework including WSDL service

128 descriptions, UDDI businessServices and bindingTemplates and SOAP message structure and message

129 processing model, and WS-SecurityPolicy SHOULD be applicable to any version of SOAP. The current

130 SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit

131 the applicability of this specification to a single version of SOAP.

## 132 1.5 Normative References

133 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement

134 Levels", RFC 2119, Harvard University, March 1997.

135 <http://www.ietf.org/rfc/rfc2119.txt>

136

137 [SOAP] W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.

138 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

139

140 [SOAP12] W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24

141 June 2003.

142 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

143

144 [SOAPNorm] W3C Working Group Note, "SOAP Version 1.2 Message

145 Normalization", 8 October 2003.

146 <http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>

147

148	[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
149		
150		
151		<a href="http://www.ietf.org/rfc/rfc3986.txt">http://www.ietf.org/rfc/rfc3986.txt</a>
152		
153	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
154		
155		<a href="http://www.ietf.org/rfc/rfc2068.txt">http://www.ietf.org/rfc/rfc2068.txt</a>
156		
157	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
158		<a href="http://www.ietf.org/rfc/rfc2246.txt">http://www.ietf.org/rfc/rfc2246.txt</a>
159		
160	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
161		<a href="http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211">http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211</a>
162		
163	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
164		
165		<a href="http://www.w3.org/TR/2006/REC-ws-addr-core-20060509">http://www.w3.org/TR/2006/REC-ws-addr-core-20060509</a>
166		
167	[WS-Policy]	W3C Recommendation, "Web Services Policy 1.5 - Framework", 04 September 2007.
168		
169		<a href="http://www.w3.org/TR/2007/REC-ws-policy-20070904/">http://www.w3.org/TR/2007/REC-ws-policy-20070904/</a>
170		W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
171		
172		<a href="http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/">http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/</a>
173		
174	[WS-PolicyAttachment]	W3C Recommendation, "Web Services Policy 1.5 - Attachment", 04 September 2007.
175		
176		<a href="http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/">http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/</a>
177		W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
178		
179		<a href="http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/">http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/</a>
180		
181		
182	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.4", 2008
183		<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200802">http://docs.oasis-open.org/ws-sx/ws-trust/200802</a>
184		OASIS Standard, "WS-Trust 1.3", March 2007
185		<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512</a>
186		
187	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.4", July 2008
188		<a href="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512">http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512</a>
189		
190	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
191		

192		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf</a>
193		
194		
195	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006.
196		
197		<a href="http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf">http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf</a>
198		
199		
200	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004
201		
202		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf</a>
203		
204		
205	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006
206		
207		<a href="http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf</a>
208		
209		
210	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004
211		
212		<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf</a>
213		
214		
215	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006
216		
217		<a href="http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf</a>
218		
219		
220	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", February 2006
221		
222		<a href="http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf</a>
223		
224		
225	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile", December 2004
226		
227		<a href="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf</a>
228		
229	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1", February 2006
230		
231		<a href="http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf">http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf</a>
232		
233		
234	[WSS:RELTTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language (REL) Token Profile", December 2004
235		
236		<a href="http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf">http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf</a>
237		

238 [WSS:RELTOKENProfile1.1] OASIS Standard, "Web Services Security Rights Expression Language  
239 (REL) Token Profile 1.1", February 2006  
240 [http://www.oasis-open.org/committees/download.php/16687/oasis-  
242 wss-rel-token-profile-1.1.pdf](http://www.oasis-open.org/committees/download.php/16687/oasis-<br/>241 wss-rel-token-profile-1.1.pdf)

243 [WSS:SwAProfile1.1] OASIS Standard, "Web Services Security SOAP Messages with  
244 Attachments (SwA) Profile 1.1", February 2006  
245 [http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-  
247 spec-os-SwAProfile.pdf](http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-<br/>246 spec-os-SwAProfile.pdf)

248 [XML-Encrypt] W3C Recommendation, "XML Encryption Syntax and Processing", 10  
249 December 2002.  
250 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>  
251

252 [XML-Signature] W3C Recommendation, "XML-Signature Syntax and Processing", 12  
253 February 2002.  
254 <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>  
255  
256 W3C Recommendation, D. Eastlake et al. XML Signature Syntax and  
257 Processing (Second Edition). 10 June 2008.  
258 <http://www.w3.org/TR/2008/REC-xmlsig-core-20080610/>  
259  
260

261 [XPath] W3C Recommendation "XML Path Language (XPath) Version 1.0", 16  
262 November 1999.  
263 <http://www.w3.org/TR/1999/REC-xpath-19991116>  
264

265 [XPath 2.0 Filter] W3C Recommendation "XML-Signature XPath Filter 2.0" 8 November  
266 2002.  
267 <http://www.w3.org/TR/2002/REC-xmlsig-filter2-20021108/>  
268

269 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second  
270 Edition", 28 October 2004.  
271 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>  
272

273 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second  
274 Edition", 28 October 2004.  
275 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>  
276

## 277 **1.6 Non-Normative References**

278 None.  
279

---

## 280 **2 Security Policy Model**

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

285  
286 The primary goal of this specification is to define an initial set of patterns or sets of assertions that  
287 represent common ways to describe how messages are secured on a communication path. The intent is  
288 to allow flexibility in terms of the tokens, cryptography, and mechanisms used, including leveraging  
289 transport security, but to be specific enough to ensure interoperability based on assertion matching.

290  
291 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for  
292 selecting policy alternatives and the attachment mechanism for associating policy assertions with web  
293 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters  
294 or attributes. This enables first-level, QName based assertion matching without security domain-specific  
295 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed  
296 set of policy alternatives that are shared by the two parties attempting to establish a secure  
297 communication path. Parameters defined by this specification represent additional information for  
298 engaging behaviors that do not need to participate in matching. When multiple security policy assertions  
299 of the same type with parameters present occur in the same policy alternative the parameters should be  
300 treated as a union. Note that a service may choose to accept messages that do not match its policy.

301  
302 In general, assertions defined in this specification allow additional attributes, based on schemas, to be  
303 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not  
304 match based on these attributes. Attributes specified on the assertion element that are not defined in this  
305 specification or in WS-Policy are to be treated as informational properties.

### 306 **2.1 Security Assertion Model**

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

314  
315 To indicate the scope of protection, assertions identify message parts that are to be protected in a  
316 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

317  
318 The general aspects of security includes the relationships between or characteristics of the environment  
319 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality  
320 protection and which are supporting, the applicable algorithms to use, etc.

321



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

327

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

332

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

## 335 2.2 Nested Policy Assertions

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

## 342 2.3 Security Binding Abstraction

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

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

348

349 A binding defines the following security characteristics:

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

358

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

365

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

- 369 • TransportBinding (Section 7.3)
- 370 • SymmetricBinding (Section 7.4)
- 371 • AsymmetricBinding (Section 7.5)



---

## 372 3 Policy Considerations

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

### 375 3.1 Nested Policy

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

378

### 379 3.2 Policy Subjects

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

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

#### 384 [Message Policy Subject]

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

387

388 wsdl:message

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

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

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

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

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

#### 397 [Operation Policy Subject]

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

399 wsdl:portType/wsdl:operation

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

402 wsdl:binding/wsdl:operation

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

405

406

#### 407 [Endpoint Policy Subject]

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

410 wsdl:portType

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

413 wsdl:binding

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

416 wsdl:port

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

419

## 4 Protection Assertions

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

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

### 4.1 Integrity Assertions

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

#### 4.1.1 SignedParts Assertion

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

437

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

#### Syntax

```
443 <sp:SignedParts xmlns:sp="..." ... >  
444   <sp:Body />?  
445   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />?  
446   <sp:Attachments>  
447     <sp13:ContentSignatureTransform /> ?  
448     <sp13:AttachmentCompleteSignatureTransform /> ?  
449   </sp:Attachments> ?  
450   ...  
451 </sp:SignedParts>
```

452

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

454 /sp:SignedParts

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

458 /sp:SignedParts/sp:Body

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

461 /sp:SignedParts/sp:Header

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

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

472 /sp:SignedParts/sp:Header/@Name

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

476 /sp:SignedParts/sp:Header/@Namespace

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

479 /sp:SignedParts/sp:Attachments

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

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

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

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

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

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

## 492 4.1.2 SignedElements Assertion

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

498

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

### 502 Syntax

```
503 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
504   <sp:XPath>xs:string</sp:XPath>+  
505   <sp13:Xpath2 Filter="xs:string">xs:string</sp13:Xpath2>+  
506   ...  
507 </sp:SignedElements>
```

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

## 526 4.2 Confidentiality Assertions

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

### 530 4.2.1 EncryptedParts Assertion

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

536  
537 There MAY be multiple EncryptedParts assertions present. Multiple EncryptedParts assertions present  
538 within a policy alternative are equivalent to a single EncryptedParts assertion containing the union of all  
539 specified message parts. Note that this assertion does not require that a given part appear in a message,  
540 just that if such a part appears, it requires confidentiality protection.

#### 541 Syntax

```
542 <sp:EncryptedParts xmlns:sp="..." ... >  
543   <sp:Body/>?  
544   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />?  
545   <sp:Attachments />?  
546   ...  
547 </sp:EncryptedParts>
```

548  
549 The following describes the attributes and elements listed in the schema outlined above:  
550 /sp:EncryptedParts

551 This assertion specifies the parts of the message that need confidentiality protection. The single  
552 child element of this assertion specifies the set of message parts using an extensible dialect.  
553 If no child elements are specified, the body of the message MUST be confidentiality protected.  
554 /sp:EncryptedParts/sp:Body  
555 Presence of this OPTIONAL empty element indicates that the entire body of the message needs  
556 to be confidentiality protected. In the case where mechanisms from WSS: SOAP Message  
557 Security are used to satisfy this assertion, then the soap:Body element is encrypted using the  
558 #Content encryption type.

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

569 /sp:EncryptedParts/sp:Header/@Name  
570 This OPTIONAL attribute indicates the local name of the SOAP header to be confidentiality  
571 protected. If this attribute is not specified, all SOAP headers whose namespace matches the  
572 Namespace attribute are to be protected.

573 /sp:EncryptedParts/sp:Header/@Namespace  
574 This REQUIRED attribute indicates the namespace of the SOAP header(s) to be confidentiality  
575 protected.

576 /sp:EncryptedParts/sp:Attachments  
577 Presence of this OPTIONAL empty element indicates that all SwA (SOAP Messages with  
578 Attachments) attachments [SwA] are to be confidentiality protected. When SOAP Message  
579 Security is used to accomplish this, all message parts other than the part containing the primary  
580 SOAP envelope are to be confidentiality protected as outlined in WSS: SOAP Message Security  
581 [WSS:SwAProfile1.1].

## 582 4.2.2 EncryptedElements Assertion

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

588  
589 There MAY be multiple EncryptedElements assertions present. Multiple EncryptedElements assertions  
590 present within a policy alternative are equivalent to a single EncryptedElements assertion containing the  
591 union of all specified XPath expressions.

### 592 Syntax

```
593 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
594   <sp:XPath>xs:string</sp:XPath>+
595   ...
596 </sp:EncryptedElements>
```

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

### 609 **4.2.3 ContentEncryptedElements Assertion**

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

615  
616 There MAY be multiple ContentEncryptedElements assertions present. Multiple  
617 ContentEncryptedElements assertions present within a policy alternative are equivalent to a single  
618 ContentEncryptedElements assertion containing the union of all specified XPath expressions.

#### 619 **Syntax**

```
620 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
621 <sp:XPath>xs:string</sp:XPath>+  
622 ...  
623 </sp:ContentEncryptedElements>
```

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

### 636 **4.3 Required Elements Assertion**

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

639



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

### 643 4.3.1 RequiredElements Assertion

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

646

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

#### 650 Syntax

```
651 <sp:RequiredElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
652 <sp:XPath>xs:string</sp:XPath> +  
653 ...  
654 </sp:RequiredElements>
```

655

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

657 /sp:RequiredElements

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

659 /sp:RequiredElements/@XPathVersion

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

662 /sp:RequiredElements/sp:XPath

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

### 667 4.3.2 RequiredParts Assertion

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

671

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

#### 675 Syntax

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

679

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

681 /sp:RequiredParts/sp:Header

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

683 /sp:RequiredParts/sp:Header/@Name



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

## 689 5 Token Assertions

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

### 694 5.1 Token Inclusion

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

#### 700 5.1.1 Token Inclusion Values

701 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.

702  
703 Note: In examples, the namespace URI is replaced with "...". For example,  
704 `.../IncludeToken/Never` is actually `http://docs.oasis-open.org/ws-sx/ws-`  
705 `securitypolicy/200702/IncludeToken/Never`. Other token inclusion URI values MAY be defined but are out-  
706 of-scope of this specification.

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

## 709 **5.1.2 Token Inclusion and Token References**

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

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

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

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

## 726 **5.2 Token Issuer and Required Claims**

### 727 **5.2.1 Token Issuer**

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

### 732 **5.2.2 Token Issuer Name**

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

737  
738 It is out of scope of this specification how the relationship between the issuer's logical name and the  
739 physical manifestation of the issuer in the security token is defined.

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

### 742 **5.2.3 Required Claims**

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

746  
747 This element indicates the REQUIRED claims that the security token must contain in order to satisfy the  
748 requirements of the token assertion.

749  
750 Individual token assertions MAY further limit what claims MAY be specified for that specific token  
751 assertion.

## 752 **5.2.4 Processing Rules and Token Matching**

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

757 For example if the receiver's policy contains two token assertions, one requires IssuedToken from issuer  
758 A with claims C1 and C2 and the second requires IssuedToken from issuer B with claims C3 and C4, the  
759 sender can satisfy such requirements with any of the following security token decomposition:  
760

- 761 1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and T2 is issued  
762 by issuer B and contains claims C3 and C4.
- 763 2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is also issued  
764 by issuer A and contains claim C2 and T3 is issued by issuer B and contains claims C3 and C4.
- 765 3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2, T2 is  
766 issued by issuer B and contains claim C3 and T3 is also issued by issuer B and contains claim  
767 C4.
- 768 4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is also issued  
769 by issuer A and contains claim C2, T3 is issued by issuer B and contains claim C3 and T4 is also  
770 issued by issuer B and contains claim C4.

## 771 **5.3 Token Properties**

### 772 **5.3.1 [Derived Keys] Property**

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

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

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

### 781 **5.3.2 [Explicit Derived Keys] Property**

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

### 785 **5.3.3 [Implied Derived Keys] Property**

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

## 789 **5.4 Token Assertion Types**

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

### 791 **5.4.1 UsernameToken Assertion**

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

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

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

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

801

802 **Syntax**

```

803 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
804 (
805   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
806   <sp:IssuerName>xs:anyURI</sp:IssuerName>
807 ) ?
808 <wst:Claims Dialect="..."> ... </wst:Claims> ?
809 <wsp:Policy xmlns:wsp="...">
810   ( (
811     <sp:NoPassword ... /> |
812     <sp:HashPassword ... />
813   ) |
814   (
815     <sp13:Created .../> ?
816     <sp13:Nonce .../> ?
817   ) ) ?
818   (
819     <sp:RequireDerivedKeys /> |
820     <sp:RequireImpliedDerivedKeys ... /> |
821     <sp:RequireExplicitDerivedKeys ... />
822   ) ?
823   (
824     <sp:WssUsernameToken10 ... /> |
825     <sp:WssUsernameToken11 ... />
826   ) ?
827   ...
828 </wsp:Policy>
829   ...
830 </sp:UsernameToken>
  
```

831

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

833 /sp:UsernameToken

834 This identifies a UsernameToken assertion.

835 /sp:UsernameToken/@sp:IncludeToken

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

837 /sp:UsernameToken/sp:Issuer

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

840 /sp:UsernameToken/sp:IssuerName

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

843 /sp:UsernameToken/wst:Claims

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

846 /sp:UsernameToken/wsp:Policy

847 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken  
848 assertion.

849 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

852 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

857 /sp13:UsernameToken/wsp:Policy/sp13:Created

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

861 /sp13:UsernameToken/wsp:Policy/sp13:Nonce

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

865 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

868 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

872 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

876 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

879 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

## 882 5.4.2 ICreatessuedToken Assertion

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

### 887 Syntax

```

888 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
889 (
890 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
891 <sp:IssuerName>xs:anyURI</sp:IssuerName>
892 ) ?
893 <wst:Claims Dialect="..."> ... </wst:Claims> ?
894 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
895 ...
896 </sp:RequestSecurityTokenTemplate>
897 <wsp:Policy xmlns:wsp="...">
898 (
899 <sp:RequireDerivedKeys ... /> |
900 <sp:RequireImpliedDerivedKeys ... /> |
901 <sp:RequireExplicitDerivedKeys ... />
902 ) ?
903 <sp:RequireExternalReference ... /> ?
904 <sp:RequireInternalReference ... /> ?
905 ...
906 </wsp:Policy>
907 ...
908 </sp:IssuedToken>

```

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

910 /sp:IssuedToken

911 This identifies an IssuedToken assertion.

912 /sp:IssuedToken/@sp:IncludeToken

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

914 /sp:IssuedToken/sp:Issuer

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

917 /sp:IssuedToken/sp:IssuerName

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

920 /sp:IssuedToken/wst:Claims

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

923 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

928 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

929 This OPTIONAL attribute contains a WS-Trust specification namespace URI identifying the  
930 version of WS-Trust referenced by the contents of this element. For example, when using Trust  
931 1.3 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200512> should be used and when using  
932 Trust 1.4 the URI <http://docs.oasis-open.org/ws-sx/ws-trust/200802> should be used.

933 /sp:IssuedToken/wsp:Policy

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

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

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

939 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

943 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

947 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

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

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

951 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

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

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

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

### 960 5.4.3 X509Token Assertion

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

#### 962 Syntax

```
963 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
964   (  
965     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
966     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
967   ) ?  
968   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```



```

969 <wsp:Policy xmlns:wsp="...">
970   (
971     <sp:RequireDerivedKeys ... /> |
972     <sp:RequireExplicitDerivedKeys ... /> |
973     <sp:RequireImpliedDerivedKeys ... />
974   ) ?
975   <sp:RequireKeyIdentifierReference ... /> ?
976   <sp:RequireIssuerSerialReference ... /> ?
977   <sp:RequireEmbeddedTokenReference ... /> ?
978   <sp:RequireThumbprintReference ... /> ?
979   (
980     <sp:WssX509V3Token10 ... /> |
981     <sp:WssX509Pkcs7Token10 ... /> |
982     <sp:WssX509PkiPathV1Token10 ... /> |
983     <sp:WssX509V1Token11 ... /> |
984     <sp:WssX509V3Token11 ... /> |
985     <sp:WssX509Pkcs7Token11 ... /> |
986     <sp:WssX509PkiPathV1Token11 ... />
987   ) ?
988   ...
989 </wsp:Policy>
990   ...
991 </sp:X509Token>

```

992

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

994 /sp:X509Token

995 This identifies an X509Token assertion.

996 /sp:X509Token/@sp:IncludeToken

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

998 /sp:X509Token/sp:Issuer

999 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1000 of the sp:X509Token.

1001 /sp:X509Token/sp:IssuerName

1002 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:X509Token  
1003 issuer.

1004 /sp:X509Token/wst:Claims

1005 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1006 order to satisfy the token assertion requirements.

1007 /sp:X509Token/wsp:Policy

1008 This REQUIRED element identifies additional requirements for use of the sp:X509Token  
1009 assertion.

1010 /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

1011 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1012 and [Implied Derived Keys] properties for this token to 'true'.

1013 /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1017 /sp:X509Token/wsp:Policy/sp:RequireImpliedDerivedKeys

1018 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1019 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1020 'false'.

1021 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

1022 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1023 REQUIRED when referencing this token.

1024 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

1025 This OPTIONAL element is a policy assertion that indicates that an issuer serial reference is  
1026 REQUIRED when referencing this token.

1027 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

1028 This OPTIONAL element is a policy assertion that indicates that an embedded token reference is  
1029 REQUIRED when referencing this token.

1030 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1031 This OPTIONAL element is a policy assertion that indicates that a thumbprint reference is  
1032 REQUIRED when referencing this token.

1033 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1034 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should  
1035 be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1036 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1037 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be  
1038 used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1039 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1040 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1  
1041 token should be used as defined in [\[WSS:X509TokenProfile1.0\]](#).

1042 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1043 This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should  
1044 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1045 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1046 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should  
1047 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1048 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1049 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be  
1050 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1051 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1052 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1  
1053 token should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

## 1054 5.4.4 KerberosToken Assertion

1055 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

### 1056 Syntax

```
1057 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1058 (  
1059   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1060   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1061 ) ?
```

```

1062 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1063 <wsp:Policy xmlns:wsp="...">
1064   (
1065     <sp:RequireDerivedKeys ... /> |
1066     <sp:RequireImpliedDerivedKeys ... /> |
1067     <sp:RequireExplicitDerivedKeys ... />
1068   ) ?
1069   <sp:RequireKeyIdentifierReference ... /> ?
1070   (
1071     <sp:WssKerberosV5ApReqToken11 ... /> |
1072     <sp:WssGssKerberosV5ApReqToken11 ... />
1073   ) ?
1074
1075   ...
1076 </wsp:Policy>
1077 ...
1078 </sp:KerberosToken>

```

1079  
1080 The following describes the attributes and elements listed in the schema outlined above:

1081 /sp:KerberosToken

1082 This identifies a KerberosV5ApReqToken assertion.

1083 /sp:KerberosToken/@sp:IncludeToken

1084 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1085 /sp:KerberosToken/sp:Issuer

1086 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1087 of the sp:KerberosToken.

1088 /sp:KerberosToken/sp:IssuerName

1089 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken  
1090 issuer.

1091 /sp:KerberosToken/wst:Claims

1092 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1093 order to satisfy the token assertion requirements.

1094 /sp:KerberosToken/wsp:Policy

1095 This REQUIRED element identifies additional requirements for use of the sp:KerberosToken  
1096 assertion.

1097 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1098 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1099 and [Implied Derived Keys] properties for this token to 'true'.

1100 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1101 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1102 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1103 'false'.

1104 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1108 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1109 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1110 REQUIRED when referencing this token.

1111 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken1

1112 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ  
1113 token should be used as defined in [WSS:KerberosTokenProfile1.1].

1114 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken1

1115 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-  
1116 REQ token should be used as defined in [WSS:KerberosTokenProfile1.1].

## 1117 5.4.5 SpnegoContextToken Assertion

1118 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg  
1119 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

### 1120 Syntax

```
1121 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1122 (   
1123 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
1124 <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1125 ) ?   
1126 <wst:Claims Dialect="..."> ... </wst:Claims> ?   
1127 <wsp:Policy xmlns:wsp="...">  
1128 (   
1129 <sp:RequireDerivedKeys ... /> |   
1130 <sp:RequireImpliedDerivedKeys ... /> |   
1131 <sp:RequireExplicitDerivedKeys ... />  
1132 ) ?   
1133 <sp:MustNotSendCancel ... /> ?   
1134 <sp:MustNotSendAmend ... /> ?   
1135 <sp:MustNotSendRenew ... /> ?   
1136 ...   
1137 </wsp:Policy>  
1138 ...   
1139 </sp:SpnegoContextToken>
```

1140

1141 The following describes the attributes and elements listed in the schema outlined above:

1142 /sp:SpnegoContextToken

1143 This identifies a SpnegoContextToken assertion.

1144 /sp:SpnegoContextToken/@sp:IncludeToken

1145 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1146 /sp:SpnegoContextToken/sp:Issuer

1147 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
1148 for the Spnego Context Token.

1149 /sp:SpnegoContextToken/sp:IssuerName

1150 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1151 sp:SpnegoContextToken issuer.

1152 /sp:SpnegoContextToken/wst:Claims

1153 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1154 order to satisfy the token assertion requirements.

1155 /sp:SpnegoContextToken/wsp:Policy

1156 This REQUIRED element identifies additional requirements for use of the  
 1157 sp:SpnegoContextToken assertion.

1158 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

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

1161 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1165 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1169 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

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

1173 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

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

1177 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

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

## 1181 5.4.6 SecurityContextToken Assertion

1182 This element represents a requirement for a SecurityContextToken token.

### 1183 Syntax

```

1184 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1185 (
1186   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1187   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1188 ) ?
1189 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1190 <wsp:Policy xmlns:wsp="...">
1191   (
1192     <sp:RequireDerivedKeys ... /> |
1193     <sp:RequireImpliedDerivedKeys ... /> |
1194     <sp:RequireExplicitDerivedKeys ... />
1195   ) ?
1196   <sp:RequireExternalUriReference ... /> ?
1197   <sp:SC13SecurityContextToken... /> ?
1198   ...
1199 </wsp:Policy>
1200 ...
1201 </sp:SecurityContextToken>

```

1202

1203 The following describes the attributes and elements listed in the schema outlined above:

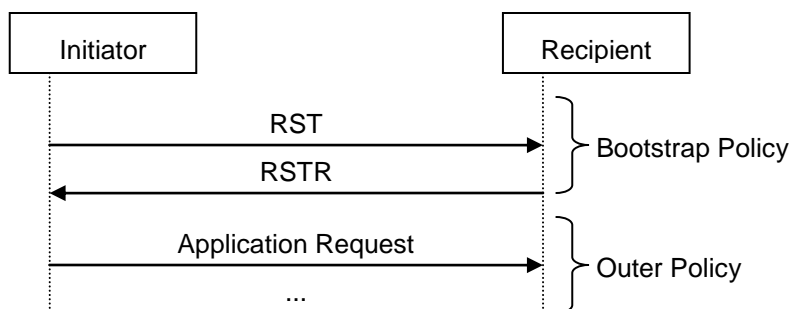
1204 /sp:SecurityContextToken

- 1205 This identifies a SecurityContextToken assertion.
- 1206 /sp:SecurityContextToken/@sp:IncludeToken
- 1207 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 1208 /sp:SecurityContextToken/sp:Issuer
- 1209 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1210 of the sp:SecurityContextToken.
- 1211 /sp:SecurityContextToken/sp:IssuerName
- 1212 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1213 sp:SecurityContextToken issuer.
- 1214 /sp:SecurityContextToken/wst:Claims
- 1215 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1216 order to satisfy the token assertion requirements.
- 1217 /sp:SecurityContextToken/wsp:Policy
- 1218 This REQUIRED element identifies additional requirements for use of the  
1219 sp:SecurityContextToken assertion.
- 1220 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys
- 1221 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1222 and [Implied Derived Keys] properties for this token to 'true'.
- 1223 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1224 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1225 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1226 'false'.
- 1227 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1228 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1229 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1230 'false'.
- 1231 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference
- 1232 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
1233 REQUIRED when referencing this token.
- 1234 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken
- 1235 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should  
1236 be used as defined in [\[WS-SecureConversation\]](#).
- 1237
- 1238 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that  
1239 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If  
1240 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the  
1241 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

## 1242 **5.4.7 SecureConversationToken Assertion**

- 1243 This element represents a requirement for a Security Context Token retrieved from the indicated issuer  
1244 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the  
1245 service endpoint address.
- 1246

1247 Note: This assertion describes the token accepted by the target service. Because this token is issued by  
 1248 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD  
 1249 contain a bootstrap policy indicating the security binding and policy that is used when requesting this  
 1250 token from the target service. That is, the bootstrap policy is used to obtain the token and then the  
 1251 current (outer) policy is used when making requests with the token. This is illustrated in the diagram  
 1252 below.



1253 1.

1254

1255 If the bootstrap policy assertion is used to indicate the security binding and policy in effect when  
 1256 requesting a secure conversation token from the target service, then subsequent Amend, Renew and  
 1257 Cancel messages MUST comply with the following rules.

1258 **Amending Context**

1259 To amend an existing secure conversation token, a requestor uses the context amending mechanism as  
 1260 described by the WS-SecureConversation specification. The message exchange MUST be secured  
 1261 using the existing (to be amended) SCT in accordance with the target service (outer) policy, combined  
 1262 with endorsing supporting tokens carrying the new claims to be associated with the amended context with  
 1263 the inclusion mode set to:

1264 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1265 See the EndorsingSupportingTokens Assertion section for more details on the usage of the endorsing  
 1266 supporting tokens.

1267 **Renewing Context**

1268 To renew an existing secure conversation token, a requestor uses the context renewal mechanism as  
 1269 described by the WS-SecureConversation specification. The message exchange MUST be secured  
 1270 according to the requirements of the bootstrap policy assertion, combined with the existing (to be  
 1271 renewed) SCT used as an endorsing supporting token with the inclusion mode set to:

1272 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient>

1273 See the EndorsingSupportingTokens Assertion section for more details on the usage of endorsing  
 1274 support tokens.

1275 **Canceling Context**

1276 To cancel an existing secure conversation token, a requestor uses the context cancelling mechanism as  
 1277 described by the WS-SecureConversation specification. The message exchange MUST be secured  
 1278 using the existing (to be cancelled) SCT in accordance with the target service (outer) policy.

1279 **Handling Policy Alternatives**

1280 If there are policy alternatives present in either the bootstrap policy assertion or the target service (outer)  
 1281 policy assertion, the following rules MUST be followed.

- 1282 • The policy alternative used as a basis for the context renewal MUST be the same as the policy  
 1283 alternative which was previously used for the context issuance.

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

1286 Policy

1287 Policy-alternative-1

1288 SecureConversationToken-assertion-1

1289 Policy-alternative-2

1290 SecureConversationToken-assertion-2

1291 The policy alternative used as basis for context amend and cancel MUST be the same as the policy  
1292 alternative that was used to obtain the context. This means that Policy-alternative-1 above cannot be  
1293 used to amend and cancel SecureConversationToken-assertion-2 and vice-versa.

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

1296 Policy

1297 SecureConversationToken-assertion-1

1298 Policy-alternative-1

1299 Policy-alternative-2

1300 Any policy alternative can be used to amend or cancel the context. This means that either Policy-  
1301 alternative-1 or Policy-alternative-2 can be used to amend or cancel SecureConversationToken-  
1302 assertion-1.

1303

### 1304 Syntax

```

1305 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1306 (
1307   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1308   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1309 ) ?
1310 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1311 <wsp:Policy xmlns:wsp="...">
1312   (
1313     <sp:RequireDerivedKeys ... /> |
1314     <sp:RequireImpliedDerivedKeys ... /> |
1315     <sp:RequireExplicitDerivedKeys ... />
1316   ) ?
1317   <sp:RequireExternalUriReference ... /> ?
1318   <sp:SC13SecurityContextToken ... /> ?
1319   <sp:MustNotSendCancel ... /> ?
1320   <sp:MustNotSendAmend ... /> ?
1321   <sp:MustNotSendRenew ... /> ?
1322   <sp:BootstrapPolicy ... >
1323     <wsp:Policy> ... </wsp:Policy>
1324   </sp:BootstrapPolicy> ?
1325 </wsp:Policy>
1326 ...
1327 </sp:SecureConversationToken>

```

1328

1329 The following describes the attributes and elements listed in the schema outlined above:

1330 /sp:SecureConversationToken

1331 This identifies a SecureConversationToken assertion.

1332 /sp:SecureConversationToken/@sp:IncludeToken

1333 This OPTIONAL attribute identifies the token inclusion value for this token assertion.



- 1334 /sp:SecureConversationToken/sp:Issuer  
 1335 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
 1336 for the Security Context Token.
- 1337 /sp:SecureConversationToken/sp:IssuerName  
 1338 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
 1339 sp:SecureConversationToken issuer.
- 1340 /sp:SpnegoContextToken/wst:Claims  
 1341 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
 1342 order to satisfy the token assertion requirements.
- 1343 /sp:SecureConversationToken/wsp:Policy  
 1344 This REQUIRED element identifies additional requirements for use of the  
 1345 sp:SecureConversationToken assertion.
- 1346 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys  
 1347 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1348 and [Implied Derived Keys] properties for this token to 'true'.
- 1349 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
 1350 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1351 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1352 'false'.
- 1353 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1354 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1355 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1356 'false'.
- 1357 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference  
 1358 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
 1359 REQUIRED when referencing this token.
- 1360 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken  
 1361 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should  
 1362 be used as obtained using the protocol defined in [[WS-SecureConversation](#)].
- 1363 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel  
 1364 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1365 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it  
 1366 means that SCT/Cancel RST messages are supported by the STS.
- 1367 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend  
 1368 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1369 conversation token does not support SCT/Amend RST messages. If this assertion is missing it  
 1370 means that SCT/Amend RST messages are supported by the STS.
- 1371 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew  
 1372 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1373 conversation token does not support SCT/Renew RST messages. If this assertion is missing it  
 1374 means that SCT/Renew RST messages are supported by the STS.
- 1375 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy  
 1376 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements  
 1377 for obtaining the Security Context Token.

1378 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy  
1379 This element contains the security binding requirements for obtaining the Security Context Token.  
1380 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with  
1381 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that  
1382 are to be protected.

### 1383 Example

```
1384 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
1385   <sp:SymmetricBinding>  
1386     <wsp:Policy>  
1387       <sp:ProtectionToken>  
1388         <wsp:Policy>  
1389           <sp:SecureConversationToken>  
1390             <sp:Issuer>  
1391               <wsa:Address>http://example.org/sts</wsa:Address>  
1392             </sp:Issuer>  
1393           <wsp:Policy>  
1394             <sp:SC13SecurityContextToken />  
1395           <sp:BootstrapPolicy>  
1396             <wsp:Policy>  
1397               <sp:AsymmetricBinding>  
1398                 <wsp:Policy>  
1399                   <sp:InitiatorToken>  
1400                     ...  
1401                   </sp:InitiatorToken>  
1402                   <sp:RecipientToken>  
1403                     ...  
1404                   </sp:RecipientToken>  
1405                 </wsp:Policy>  
1406               </sp:AsymmetricBinding>  
1407             <sp:SignedParts>  
1408               ...  
1409             </sp:SignedParts>  
1410             ...  
1411           </wsp:Policy>  
1412         </sp:BootstrapPolicy>  
1413       </wsp:Policy>  
1414     </sp:SecureConversationToken>  
1415   </wsp:Policy>  
1416 </sp:ProtectionToken>  
1417   ...  
1418 </wsp:Policy>  
1419 </sp:SymmetricBinding>  
1420 <sp:SignedParts>  
1421   ...  
1422 </sp:SignedParts>  
1423   ...  
1424 </wsp:Policy>
```

### 1425 5.4.8 SamlToken Assertion

1426 This element represents a requirement for a SAML token.

#### 1427 Syntax

```
1428 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1429   (  
1430     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1431     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1432   ) ?  
1433   <wst:Claims Dialect="..."> ... </wst:Claims> ?
```

```

1434 <wsp:Policy xmlns:wsp="...">
1435   (
1436     <sp:RequireDerivedKeys ... /> |
1437     <sp:RequireImpliedDerivedKeys ... /> |
1438     <sp:RequireExplicitDerivedKeys ... />
1439   ) ?
1440   <sp:RequireKeyIdentifierReference ... /> ?
1441   (
1442     <sp:WssSamlV11Token10 ... /> |
1443     <sp:WssSamlV11Token11 ... /> |
1444     <sp:WssSamlV20Token11 ... />
1445   ) ?
1446   ...
1447 </wsp:Policy>
1448   ...
1449 </sp:SamlToken>

```

1450

1451 The following describes the attributes and elements listed in the schema outlined above:

1452 /sp:SamlToken

1453 This identifies a SamlToken assertion.

1454 /sp:SamlToken/@sp:IncludeToken

1455 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1456 /sp:SamlToken/sp:Issuer

1457 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1458 of the sp:SamlToken.

1459 /sp:SamlToken/sp:IssuerName

1460 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamlToken  
1461 issuer.

1462 /sp:SamlToken/wst:Claims

1463 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1464 order to satisfy the token assertion requirements.

1465 /sp:SamlToken/wsp:Policy

1466 This REQUIRED element identifies additional requirements for use of the sp:SamlToken  
1467 assertion.

1468 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

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

1471 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1475 /sp:SamlToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1479 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1480 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1481 REQUIRED when referencing this token.

1482 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10  
1483 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1484 should be used as defined in [WSS:SAMLTOKENPROFILE1.0].

1485 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11  
1486 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1487 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1488 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11  
1489 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token  
1490 should be used as defined in [WSS:SAMLTOKENPROFILE1.1].

1491  
1492 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties  
1493 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
1494 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion  
1495 SHOULD be used instead.

## 1496 5.4.9 RelToken Assertion

1497 This element represents a requirement for a REL token.

### 1498 Syntax

```
1499 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1500 (   
1501   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1502   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1503 ) ?  
1504 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1505 <wsp:Policy xmlns:wsp="...">  
1506   (   
1507     <sp:RequireDerivedKeys ... /> |  
1508     <sp:RequireImpliedDerivedKeys ... /> |  
1509     <sp:RequireExplicitDerivedKeys ... />  
1510   ) ?  
1511   <sp:RequireKeyIdentifierReference ... /> ?  
1512   (   
1513     <sp:WssRelV10Token10 ... /> |  
1514     <sp:WssRelV20Token10 ... /> |  
1515     <sp:WssRelV10Token11 ... /> |  
1516     <sp:WssRelV20Token11 ... />  
1517   ) ?  
1518   ...  
1519 </wsp:Policy>  
1520   ...  
1521 </sp:RelToken>
```

1522  
1523 The following describes the attributes and elements listed in the schema outlined above:

1524 /sp:RelToken

1525 This identifies a RelToken assertion.

1526 /sp:RelToken/@sp:IncludeToken

1527 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1528 /sp:RelToken/sp:Issuer

1529 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1530 of the sp:RelToken.  
1531

- 1531 /sp:RelToken/sp:IssuerName
- 1532 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken  
1533 issuer.
- 1534 /sp:RelToken/wst:Claims
- 1535 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1536 order to satisfy the token assertion requirements.
- 1537 /sp:RelToken/wsp:Policy
- 1538 This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.
- 1539 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys
- 1540 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1541 and [Implied Derived Keys] property for this token to 'true'.
- 1542 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1543 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1544 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1545 'false'.
- 1546 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1547 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1548 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1549 'false'.
- 1550 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference
- 1551 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1552 REQUIRED when referencing this token.
- 1553 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10
- 1554 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
1555 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).
- 1556 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10
- 1557 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
1558 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).
- 1559 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11
- 1560 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
1561 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).
- 1562 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11
- 1563 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
1564 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).
- 1565
- 1566 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties  
1567 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
1568 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion  
1569 SHOULD be used instead.

## 1570 **5.4.10 HttpsToken Assertion**

1571 This element represents a requirement for a transport binding to support the use of HTTPS.

### 1572 **Syntax**

```

1573 <sp:HttpsToken xmlns:sp="..." ... >
1574 (
1575   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1576   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1577 ) ?
1578 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1579 <wsp:Policy xmlns:wsp="...">
1580 (
1581   <sp:HttpBasicAuthentication /> |
1582   <sp:HttpDigestAuthentication /> |
1583   <sp:RequireClientCertificate /> |
1584   ...
1585 ) ?
1586 ...
1587 </wsp:Policy>
1588 ...
1589 </sp:HttpsToken>

```

1590 The following describes the attributes and elements listed in the schema outlined above:

1591 /sp:HttpsToken

1592 This identifies an Https assertion stating that use of the HTTPS protocol specification is  
1593 supported.

1594 /sp:HttpsToken/sp:Issuer

1595 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1596 of the sp:HttpsToken.

1597 /sp:HttpsToken/sp:IssuerName

1598 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken  
1599 issuer.

1600 /sp:HttpsToken/wst:Claims

1601 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1602 order to satisfy the token assertion requirements.

1603 /sp:HttpsToken/wsp:Policy

1604 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken  
1605 assertion.

1606 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1607 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic  
1608 Authentication [[RFC2068](#)] to authenticate to the service.

1609 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1610 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP  
1611 Digest Authentication [[RFC2068](#)] to authenticate to the service.

1612 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1613 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a  
1614 certificate when negotiating the HTTPS session.

### 1615 5.4.11 KeyValueToken Assertion

1616 This element represents a requirement for a KeyValue token. The next section defines the KeyValue  
1617 security token abstraction for purposes of this token assertion.

1618

1619 This document defines requirements for KeyValueType when used in combination with RSA  
1620 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by  
1621 introducing new nested assertions besides *sp:RsaKeyValue*.

## 1622 Syntax

```
1623 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1624 <wsp:Policy xmlns:wsp="...">  
1625 <sp:RsaKeyValue ... /> ?  
1626 ...  
1627 </wsp:Policy>  
1628 ...  
1629 </sp:KeyValueToken>
```

1630 The following describes the attributes listed in the schema outlined above:

1631 /sp:KeyValueToken

1632 This identifies a RsaToken assertion.

1633 /sp:KeyValueToken/@sp:IncludeToken

1634 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1635 /sp:KeyValueToken/wsp:Policy

1636 This REQUIRED element identifies additional requirements for use of the sp:KeyValueToken  
1637 assertion.

1638 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1639 This OPTIONAL element is a policy assertion that indicates that the ds:RSAKeyValue element  
1640 must be present in the KeyValueType token. This indicates that an RSA key pair must be used.

### 1641 5.4.11.1 KeyValueType Token

1642 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key  
1643 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this  
1644 section is to define the KeyValueType abstraction that represents such key pair referencing mechanism.

1645  
1646 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be  
1647 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*  
1648 element in combination with RSA cryptographic algorithm.

1649  
1650 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the  
1651 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1652 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1653 <ds:KeyValue>  
1654 <ds:RSAKeyValue>  
1655 <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1656 <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1657 </ds:RSAKeyValue>  
1658 <ds:KeyValue>  
1659 </ds:KeyInfo>
```

1660  
1661 When the KeyValueType token is used the corresponding public key value appears directly in the signature or  
1662 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValueType token  
1663 manifestation outside the *ds:KeyInfo* element.

```
1664 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1665 <SignedInfo>  
1666 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1667 c14n#" />  
1668 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1669 <Reference URI="#_1">  
1670 <Transforms>
```

```

1671     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1672   </Transforms>
1673   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1674   <DigestValue>...</DigestValue>
1675   </Reference>
1676 </SignedInfo>
1677 <SignatureValue>...</SignatureValue>
1678 <KeyInfo>
1679   <KeyValue>
1680     <RSAKeyValue>
1681       <Modulus>...</Modulus>
1682       <Exponent>...</Exponent>
1683     </RSAKeyValue>
1684   </KeyValue>
1685 </KeyInfo>
1686 </Signature>

```

1687  
1688 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no  
1689 identifier can be associated with the token, the KeyValue token cannot be referenced by using  
1690 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the KeyValue  
1691 token can be used whenever a security token can be used as illustrated on the following example:

```

1692 <t:RequestSecurityToken xmlns:t="...">
1693   <t:RequestType>...</t:RequestType>
1694   ...
1695   <t:UseKey>
1696     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1697       <KeyValue>
1698         <RSAKeyValue>
1699           <Modulus>...</Modulus>
1700           <Exponent>...</Exponent>
1701         </RSAKeyValue>
1702       </KeyValue>
1703     </KeyInfo>
1704   </t:UseKey>
1705 </t:RequestSecurityToken>

```



---

## 6 Security Binding Properties

1706

1707 This section defines the various properties or conditions of a security binding, their semantics, values and  
1708 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are  
1709 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that  
1710 populates a value of a property appears in a policy, that property is set to the value indicated by the  
1711 assertion. The security binding then uses the value of the property to control its behavior. The properties  
1712 listed here are common to the various security bindings described in Section 7. Assertions that define  
1713 values for these properties are defined in Section 7. The following properties are used by the security  
1714 binding assertions.

### 6.1 [Algorithm Suite] Property

1715

1716 This property specifies the algorithm suite REQUIRED for performing cryptographic operations with  
1717 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and  
1718 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This  
1719 property defines the set of available algorithms. The value of this property is typically referenced by a  
1720 security binding and is used to specify the algorithms used for all message level cryptographic operations  
1721 performed under the security binding.

1722 Note: In some cases, this property MAY be referenced under a context other than a security binding and  
1723 used to control the algorithms used under that context. For example, supporting token assertions define  
1724 such a context. In such contexts, the specified algorithms still apply to message level cryptographic  
1725 operations.

1726 An algorithm suite defines values for each of the following operations and properties:

- 1727 • [Sym Sig] Symmetric Key Signature
- 1728 • [Asym Sig] Signature with an asymmetric key
- 1729 • [Dig] Digest
- 1730 • [Enc] Encryption
- 1731 • [Sym KW] Symmetric Key Wrap
- 1732 • [Asym KW] Asymmetric Key Wrap
- 1733 • [Comp Key] Computed key
- 1734 • [Enc KD] Encryption key derivation
- 1735 • [Sig KD] Signature key derivation
- 1736 • [Min SKL] Minimum symmetric key length
- 1737 • [Max SKL] Maximum symmetric key length
- 1738 • [Min AKL] Minimum asymmetric key length
- 1739 • [Max AKL] Maximum asymmetric key length

1740

1741 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/xmlsig#hmac-sha1">http://www.w3.org/2000/09/xmlsig#hmac-sha1</a>
RsaSha1	<a href="http://www.w3.org/2000/09/xmlsig#rsa-sha1">http://www.w3.org/2000/09/xmlsig#rsa-sha1</a>
Sha1	<a href="http://www.w3.org/2000/09/xmlsig#sha1">http://www.w3.org/2000/09/xmlsig#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/TR/2001/REC-xml-c14n-20010315>  
 C14N11 <http://www.w3.org/2006/12/xml-c14n11>  
 ExC14N <http://www.w3.org/2001/10/xml-exc-c14n#>  
 SNT <http://www.w3.org/TR/soap12-n11n>  
 STRT10 <http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-soap-message-security-1.0#STR-Transform>  
 AbsXPath <http://docs.oasis-open.org/...TBD.../AbsXPath>

1742

1743 The tables below show all the base algorithm suites defined by this specification. This table defines  
 1744 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

1745 This table defines additional properties whose values can be specified along with the default value for that  
 1746 property.

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

1747 This table defines values for the remaining components for each algorithm suite.

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

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

## 1748 6.2 [Timestamp] Property

1749 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`  
 1750 header. If the value is 'true', the timestamp element **MUST** be present and **MUST** be integrity protected  
 1751 either by transport or message level security. If the value is 'false', the timestamp element **MUST NOT** be  
 1752 present. The default value for this property is 'false'.

## 1753 6.3 [Protection Order] Property

1754 This property indicates the order in which integrity and confidentiality are applied to the message, in  
 1755 cases where both integrity and confidentiality are **REQUIRED**:

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

1756 The default value for this property is 'SignBeforeEncrypting'.

## 1757 6.4 [Signature Protection] Property

1758 This boolean property specifies whether the signature **MUST** be encrypted. If the value is 'true', the  
 1759 primary signature **MUST** be encrypted and any signature confirmation elements **MUST** also be encrypted.  
 1760 The primary signature element is **NOT REQUIRED** to be encrypted if the value is 'true' when there is  
 1761 nothing in the message that is covered by this signature that is encrypted. If the value is 'false', the  
 1762 primary signature **MUST NOT** be encrypted and any signature confirmation elements **MUST NOT** be  
 1763 encrypted. The default value for this property is 'false'.

## 1764 6.5 [Token Protection] Property

1765 This boolean property specifies whether signatures **MUST** cover the token used to generate that  
 1766 signature. If the value is 'true', then each token used to generate a signature **MUST** be covered by that  
 1767 signature. If the value is 'false', then the token **MUST NOT** be covered by the signature. Note that in  
 1768 cases where derived keys are used the 'main' token, and **NOT** the derived key token, is covered by the  
 1769 signature. It is **RECOMMENDED** that assertions that define values for this property apply to [Endpoint  
 1770 Policy Subject]. The default value for this property is 'false'.

1771 **6.6 [Entire Header and Body Signatures] Property**

1772 This boolean property specifies whether signature digests over the SOAP body and SOAP headers  
1773 MUST only cover the entire body and entire header elements. If the value is 'true', then each digest over  
1774 the SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In  
1775 addition each digest over a SOAP header MUST be over an actual header element and not a descendant  
1776 of a header element. This restriction does not specifically apply to the wsse:Security header. However  
1777 signature digests over child elements of the wsse:Security header MUST be over the entire child element  
1778 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a  
1779 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to  
1780 'true' mitigates against some possible re-writing attacks. It is RECOMENDDED that assertions that define  
1781 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

1782 **6.7 [Security Header Layout] Property**

1783 This property indicates which layout rules to apply when adding items to the security header. The  
1784 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.

1785

1786 **6.7.1 Strict Layout Rules for WSS 1.0**

- 1787 1. Tokens that are included in the message MUST be declared before use. For example:
- 1788 a. A local signing token MUST occur before the signature that uses it.
  - 1789 b. A local token serving as the source token for a derived key token MUST occur before that  
1790 derived key token.
  - 1791 c. A local encryption token MUST occur before the reference list that points to  
1792 xenc:EncryptedData elements that use it.
  - 1793 d. If the same token is used for both signing and encryption, then it SHOULD appear before  
1794 the ds:Signature and xenc:ReferenceList elements in the security header that are  
1795 generated using the token.
- 1796 2. Signed elements inside the security header MUST occur before the signature that signs them.  
1797 For example:
- 1798 a. A timestamp MUST occur before the signature that signs it.

- 1799            b. A Username token (usually in encrypted form) MUST occur before the signature that  
1800            signs it.
- 1801            c. A primary signature MUST occur before the supporting token signature that signs the  
1802            primary signature's signature value element.
- 1803            3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1804            has the same order requirements as the source plain text element, unless requirement 4  
1805            indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1806            supporting token signature per 2.c above and an encrypted token has the same ordering  
1807            requirements as the unencrypted token.

1808            If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top  
1809            level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the  
1810            security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any  
1811            xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict  
1812            Layout Rules for WSS 1.1

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

---

## 1842 7 Security Binding Assertions

1843 The appropriate representation of the different facets of security mechanisms requires distilling the  
1844 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy  
1845 scope of assertions defined in this section is the policy scope of their containing element.

### 1846 7.1 AlgorithmSuite Assertion

1847 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]  
1848 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

#### 1849 Syntax

```
1850 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1851   <wsp:Policy xmlns:wsp="...">  
1852     (<sp:Basic256 ... /> |  
1853     <sp:Basic192 ... /> |  
1854     <sp:Basic128 ... /> |  
1855     <sp:TripleDes ... /> |  
1856     <sp:Basic256Rsa15 ... /> |  
1857     <sp:Basic192Rsa15 ... /> |  
1858     <sp:Basic128Rsa15 ... /> |  
1859     <sp:TripleDesRsa15 ... /> |  
1860     <sp:Basic256Sha256 ... /> |  
1861     <sp:Basic192Sha256 ... /> |  
1862     <sp:Basic128Sha256 ... /> |  
1863     <sp:TripleDesSha256 ... /> |  
1864     <sp:Basic256Sha256Rsa15 ... /> |  
1865     <sp:Basic192Sha256Rsa15 ... /> |  
1866     <sp:Basic128Sha256Rsa15 ... /> |  
1867     <sp:TripleDesSha256Rsa15 ... /> |  
1868     ...)  
1869     <sp:InclusiveC14N ... /> ?  
1870     <sp:InclusiveC14N11 ... /> ?  
1871     <sp:SOAPNormalization10 ... /> ?  
1872     <sp:STRTransform10 ... /> ?  
1873     (<sp:XPath10 ... /> |  
1874     <sp:XPathFilter20 ... /> |  
1875     <sp:AbsXPath ... /> |  
1876     ...)?  
1877     ...  
1878   </wsp:Policy>  
1879   ...  
1880 </sp:AlgorithmSuite>
```

1881  
1882 The following describes the attributes and elements listed in the schema outlined above:

1883 /sp:AlgorithmSuite

1884     This identifies an AlgorithmSuite assertion.

1885 /sp:AlgorithmSuite/wsp:Policy

1886     This REQUIRED element contains one or more policy assertions that indicate the specific  
1887     algorithm suite to use.

1888 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1889     This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1890     set to 'Basic256'.

- 1891 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192
- 1892 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1893 set to 'Basic192'.
- 1894 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128
- 1895 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1896 set to 'Basic128'.
- 1897 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes
- 1898 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1899 set to 'TripleDes'.
- 1900 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15
- 1901 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1902 set to 'Basic256Rsa15'.
- 1903 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15
- 1904 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1905 set to 'Basic192Rsa15'.
- 1906 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15
- 1907 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1908 set to 'Basic128Rsa15'.
- 1909 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15
- 1910 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1911 set to 'TripleDesRsa15'.
- 1912 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256
- 1913 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1914 set to 'Basic256Sha256'.
- 1915 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256
- 1916 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1917 set to 'Basic192Sha256'.
- 1918 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256
- 1919 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1920 set to 'Basic128Sha256'.
- 1921 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256
- 1922 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1923 set to 'TripleDesSha256'.
- 1924 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15
- 1925 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1926 set to 'Basic256Sha256Rsa15'.
- 1927 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15
- 1928 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1929 set to 'Basic192Sha256Rsa15'.
- 1930 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15
- 1931 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1932 set to 'Basic128Sha256Rsa15'.
- 1933 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15



1934 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1935 set to 'TripleDesSha256Rsa15'.

1936 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1937 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an  
1938 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]  
1939 property is 'ExC14N'.

1940 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N11

1941  
1942 This optional element is a policy assertion that indicates that the  
1943 [C14N] property of an algorithm suite is set to 'C14N11'. Note: as  
1944 indicated in Section 6.1 the default value of the [C14N] property is  
1945 'ExC14N'.

1946 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1947 This OPTIONAL element is a policy assertion that indicates that the [SOAP Norm] property is set  
1948 to 'SNT'.

1949 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1950 This OPTIONAL element is a policy assertion that indicates that the [STR Transform] property is  
1951 set to 'STRT10'.

1952 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1953 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1954 'XPath'.

1955 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1956 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1957 'XPath20'.

1958 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1959 This OPTIONAL element is a policy assertion that indicates that the [XPath] property is set to  
1960 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1961

## 1962 7.2 Layout Assertion

1963 This assertion indicates a requirement for a particular security header layout as defined under the  
1964 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its  
1965 containing assertion.

### 1966 Syntax

```
1967 <sp:Layout xmlns:sp="..." ... >  
1968   <wsp:Policy xmlns:wsp="...">  
1969     <sp:Strict ... /> |  
1970     <sp:Lax ... /> |  
1971     <sp:LaxTsFirst ... /> |  
1972     <sp:LaxTsLast ... /> |  
1973     ...  
1974   </wsp:Policy>  
1975   ...  
1976 </sp:Layout>
```

1977

1978 The following describes the attributes and elements listed in the schema outlined above:

1979 /sp:Layout



- 1980 This identifies a Layout assertion.
- 1981 /sp:Layout/wsp:Policy
- 1982 This REQUIRED element contains one or more policy assertions that indicate the specific security  
1983 header layout to use.
- 1984 /sp:Layout/wsp:Policy/sp:Strict
- 1985 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1986 property is set to 'Strict'.
- 1987 /sp:Layout/wsp:Policy/sp:Lax
- 1988 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1989 property is set to 'Lax'.
- 1990 /sp:Layout/wsp:Policy/sp:LaxTsFirst
- 1991 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1992 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to  
1993 'true' by the presence of an sp:IncludeTimestamp assertion.
- 1994 /sp:Layout/wsp:Policy/sp:LaxTsLast
- 1995 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1996 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to  
1997 'true' by the presence of an sp:IncludeTimestamp assertion.

### 1998 7.3 TransportBinding Assertion

- 1999 The TransportBinding assertion is used in scenarios in which message protection and security correlation  
2000 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like  
2001 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by  
2002 the transport. This binding has one binding specific token property; [Transport Token]. This assertion  
2003 MUST apply to [Endpoint Policy Subject].

#### 2004 Syntax

```

2005 <sp:TransportBinding xmlns:sp="..." ... >
2006   <wsp:Policy xmlns:wsp="...">
2007     <sp:TransportToken ... >
2008       <wsp:Policy> ... </wsp:Policy>
2009       ...
2010     </sp:TransportToken>
2011     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2012     <sp:Layout ... > ... </sp:Layout> ?
2013     <sp:IncludeTimestamp ... /> ?
2014     ...
2015   </wsp:Policy>
2016   ...
2017 </sp:TransportBinding>

```

- 2018
- 2019 The following describes the attributes and elements listed in the schema outlined above:

- 2020 /sp:TransportBinding
- 2021 This identifies a TransportBinding assertion.
- 2022 /sp:TransportBinding/wsp:Policy
- 2023 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding  
2024 assertion.
- 2025 /sp:TransportBinding/wsp:Policy/sp:TransportToken

2026 This REQUIRED element is a policy assertion that indicates a requirement for a Transport Token.  
2027 The specified token populates the [Transport Token] property and indicates how the transport is  
2028 secured.

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

2030 This indicates a nested policy that identifies the type of Transport Token to use.

2031 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

2032 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2033 Suite] property. See Section 6.1 for more details.

2034 /sp:TransportBinding/wsp:Policy/sp:Layout

2035 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2036 Header Layout] property. See Section 6.7 for more details.

2037 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

2038 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2039 to 'true'.

## 2040 7.4 SymmetricBinding Assertion

2041 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means  
2042 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;  
2043 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this  
2044 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to  
2045 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to  
2046 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token  
2047 properties and is used as the basis for both encryption and signature in both directions. This assertion  
2048 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

### 2049 Syntax

```
2050 <sp:SymmetricBinding xmlns:sp="..." ... >  
2051   <wsp:Policy xmlns:wsp="...">  
2052     (  
2053       <sp:EncryptionToken ... >  
2054         <wsp:Policy> ... </wsp:Policy>  
2055       </sp:EncryptionToken>  
2056       <sp:SignatureToken ... >  
2057         <wsp:Policy> ... </wsp:Policy>  
2058       </sp:SignatureToken>  
2059     ) | (  
2060       <sp:ProtectionToken ... >  
2061         <wsp:Policy> ... </wsp:Policy>  
2062       </sp:ProtectionToken>  
2063     )  
2064     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>  
2065     <sp:Layout ... > ... </sp:Layout> ?  
2066     <sp:IncludeTimestamp ... /> ?  
2067     <sp:EncryptBeforeSigning ... /> ?  
2068     <sp:EncryptSignature ... /> ?  
2069     <sp:ProtectTokens ... /> ?  
2070     <sp:OnlySignEntireHeadersAndBody ... /> ?  
2071     ...  
2072   </wsp:Policy>  
2073   ...  
2074 </sp:SymmetricBinding>
```

2075

2076 The following describes the attributes and elements listed in the schema outlined above:

- 2077 /sp:SymmetricBinding  
2078 This identifies a SymmetricBinding assertion.
- 2079 /sp:SymmetricBinding/wsp:Policy  
2080 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding  
2081 assertion.
- 2082 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken  
2083 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption  
2084 Token. The specified token populates the [Encryption Token] property and is used for encryption.  
2085 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.
- 2086 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy  
2087 The policy contained here MUST identify exactly one token to use for encryption.
- 2088 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken  
2089 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.  
2090 The specified token populates the [Signature Token] property and is used for the message  
2091 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be  
2092 specified.
- 2093 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy  
2094 The policy contained here MUST identify exactly one token to use for signatures.
- 2095 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken  
2096 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.  
2097 The specified token populates the [Encryption Token] and [Signature Token properties] and is  
2098 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken  
2099 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be  
2100 specified.
- 2101 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy  
2102 The policy contained here MUST identify exactly one token to use for protection.
- 2103 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite  
2104 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2105 Suite] property. See Section 6.1 for more details.
- 2106 /sp:SymmetricBinding/wsp:Policy/sp:Layout  
2107 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2108 Header Layout] property. See Section 6.7 for more details.
- 2109 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
2110 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2111 to 'true'.
- 2112 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning  
2113 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
2114 set to 'EncryptBeforeSigning'.
- 2115 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature  
2116 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
2117 property is set to 'true'.
- 2118 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens  
2119 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
2120 set to 'true'.

2121 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
2122 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
2123 Signatures] property is set to 'true'.

## 2124 7.5 AsymmetricBinding Assertion

2125 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means  
2126 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly  
2127 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and  
2128 signature. However it is also common practice to use distinct keys for encryption and signature, because  
2129 of their different lifecycles.

2130  
2131 This binding enables either of these practices by means of four binding specific token properties: [Initiator  
2132 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption  
2133 Token].

2134  
2135 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator  
2136 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient  
2137 Encryption Token] will both refer to the same token.

2138  
2139 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator  
2140 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient  
2141 Encryption Token] will refer to different tokens.

2142  
2143 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the  
2144 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response  
2145 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the  
2146 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for  
2147 the message encryption from initiator to the recipient. Note that in each case, the token is associated with  
2148 the party (initiator or recipient) who knows the secret.

2149 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy  
2150 Subject].

### 2151 Syntax

```
2152 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2153   <wsp:Policy xmlns:wsp="...">  
2154     (  
2155       <sp:InitiatorToken>  
2156         <wsp:Policy> ... </wsp:Policy>  
2157       </sp:InitiatorToken>  
2158     ) | (  
2159       <sp:InitiatorSignatureToken>  
2160         <wsp:Policy> ... </wsp:Policy>  
2161       </sp:InitiatorSignatureToken>  
2162       <sp:InitiatorEncryptionToken>  
2163         <wsp:Policy> ... </wsp:Policy>  
2164       </sp:InitiatorEncryptionToken>  
2165     )  
2166     (  
2167       <sp:RecipientToken>  
2168         <wsp:Policy> ... </wsp:Policy>  
2169       </sp:RecipientToken>  
2170     ) | (  
2171
```

```

2171     <sp:RecipientSignatureToken>
2172         <wsp:Policy> ... </wsp:Policy>
2173     </sp:RecipientSignatureToken>
2174     <sp:RecipientEncryptionToken>
2175         <wsp:Policy> ... </wsp:Policy>
2176     </sp:RecipientEncryptionToken>
2177 )
2178 <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2179 <sp:Layout ... > ... </sp:Layout> ?
2180 <sp:IncludeTimestamp ... /> ?
2181 <sp:EncryptBeforeSigning ... /> ?
2182 <sp:EncryptSignature ... /> ?
2183 <sp:ProtectTokens ... /> ?
2184 <sp:OnlySignEntireHeadersAndBody ... /> ?
2185 ...
2186 </wsp:Policy>
2187 ...
2188 </sp:AsymmetricBinding>

```

2189  
2190 The following describes the attributes and elements listed in the schema outlined above:

2191 /sp:AsymmetricBinding

2192       This identifies a AsymmetricBinding assertion.

2193 /sp:AsymmetricBinding/wsp:Policy

2194       This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding  
2195       assertion.

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

2197       This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.  
2198       The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]  
2199       properties and is used for the message signature from initiator to recipient, and encryption from  
2200       recipient to initiator.

2201 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2202       The policy contained here MUST identify one or more token assertions.

2203 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2204       This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2205       Signature Token. The specified token populates the [Initiator Signature Token] property and is  
2206       used for the message signature from initiator to recipient.

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

2208       The policy contained here MUST identify one or more token assertions.

2209 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2210       This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2211       Encryption Token. The specified token populates the [Initiator Encryption Token] property and is  
2212       used for the message encryption from recipient to initiator.

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

2214       The policy contained here MUST identify one or more token assertions.

2215 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2216       This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.  
2217       The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]  
2218       property and is used for encryption from initiator to recipient, and for the message signature from  
2219       recipient to initiator.

- 2220 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy  
2221 The policy contained here MUST identify one or more token assertions.
- 2222 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken  
2223 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
2224 Signature Token. The specified token populates the [Recipient Signature Token] property and is  
2225 used for the message signature from recipient to initiator.
- 2226 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy  
2227 The policy contained here MUST identify one or more token assertions.
- 2228 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken  
2229 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
2230 Encryption Token. The specified token populates the [Recipient Encryption Token] property and  
2231 is used for the message encryption from initiator to recipient.
- 2232 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy  
2233 The policy contained here MUST identify one or more token assertions.
- 2234 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite  
2235 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2236 Suite] property. See Section 6.1 for more details.
- 2237 /sp:AsymmetricBinding/wsp:Policy/sp:Layout  
2238 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2239 Header Layout] property. See Section 6.7 for more details.
- 2240 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
2241 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2242 to 'true'.
- 2243 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning  
2244 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
2245 set to 'EncryptBeforeSigning'.
- 2246 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature  
2247 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
2248 property is set to 'true'.
- 2249 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens  
2250 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
2251 set to 'true'.
- 2252 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
2253 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
2254 Signatures] property is set to 'true'.

2255

## 8 Supporting Tokens

2256 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to  
2257 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore  
2258 be referred to as the “message signature”. In case of Transport Binding the message is signed outside of  
2259 the message XML by the underlying transport protocol and the signature itself is not part of the message.  
2260 Additional tokens MAY be specified to augment the claims provided by the token associated with the  
2261 “message signature” provided by the Security Binding. This section defines seven properties related to  
2262 supporting token requirements which MAY be referenced by a Security Binding: [Supporting Tokens],  
2263 [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens],  
2264 [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing  
2265 Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties:  
2266 SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,  
2267 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,  
2268 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These  
2269 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy  
2270 Subject] or [Operation Policy Subject].

2271

2272 Supporting tokens MAY be specified at a different scope than the binding assertion which provides  
2273 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while  
2274 the supporting tokens might be defined at the scope of a message. When assertions that populate this  
2275 property are defined in overlapping scopes, the sender SHOULD merge the requirements by including all  
2276 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

2277

2278 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the  
2279 tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements  
2280 (tokens, signatures, reference lists etc.) in the security header would be used to determine which order  
2281 signature and encryptions occurred in.

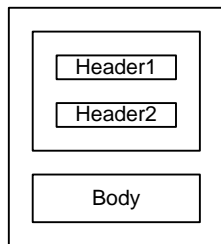
2282

2283 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional  
2284 constraints defined by the supporting token assertion. For example, if the supporting token assertion  
2285 specifies message parts that need to be encrypted, the specified tokens need to be capable of  
2286 encryption.

2287

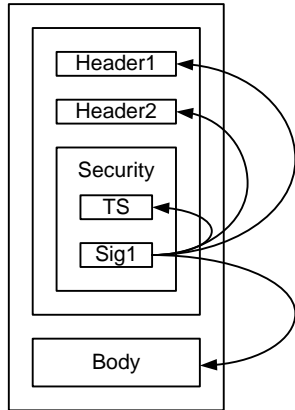
2288 To illustrate the different ways that supporting tokens MAY be bound to the message, let’s consider a  
2289 message with three components: Header1, Header2, and Body.

2290



2291 2.

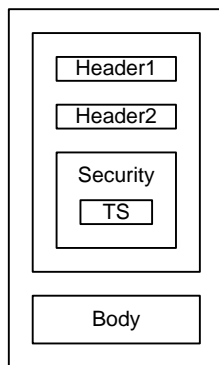
2292 Even before any supporting tokens are added, each binding requires that the message is signed using a  
 2293 token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important  
 2294 parts of the message including the message timestamp (TS) facilitate replay detection. The signature is  
 2295 then included as part of the Security header as illustrated below:  
 2296



2297 3.

2298 Note: if REQUIRED, the initiator may also include in the Security header the token used as the basis for  
 2299 the message signature (Sig1), not shown in the diagram.

2300 If transport security is used, only the message timestamp (TS) is included in the Security header as  
 2301 illustrated below. The “message signature” is provided by the underlying transport protocol and is not part  
 2302 of the message XML.



2303 4.

## 2304 8.1 SupportingTokens Assertion

2305 Supporting tokens are included in the security header and MAY OPTIONALLY include additional  
 2306 message parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and  
 2307 do not require any protection (signature or encryption) to be applied to the message before they are  
 2308 added. More specifically there is no requirement on “message signature” being present before the  
 2309 supporting tokens are added. However it is RECOMMENDED to employ underlying protection  
 2310 mechanism to ensure that the supporting tokens are cryptographically bound to the message during the  
 2311 transmission.

### 2312 Syntax

```
2313 <sp:SupportingTokens xmlns:sp="..." ... >
2314   <wsp:Policy xmlns:wsp="...">
2315     [Token Assertion]+
2316     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2317     (
2318       <sp:SignedParts ... > ... </sp:SignedParts> |
```



```

2319     <sp:SignedElements ... > ... </sp:SignedElements> |
2320     <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2321     <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2322     <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2323   ) *
2324   ...
2325 </wsp:Policy>
2326   ...
2327 </sp:SupportingTokens>

```

2328

2329 The following describes the attributes and elements listed in the schema outlined above:

2330 /sp:SupportingTokens

2331 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting  
2332 Tokens] property.

2333 /sp:SupportingTokens/wsp:Policy

2334 This describes additional requirements for satisfying the SupportingTokens assertion.

2335 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2336 The policy MUST identify one or more token assertions.

2337 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2338 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2339 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2340 by this policy assertion.

2341 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2342 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2343 and describes additional message parts that MUST be included in the signature generated with  
2344 the token identified by this policy assertion.

2345 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2346 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2347 and describes additional message elements that MUST be included in the signature generated  
2348 with the token identified by this policy assertion.

2349 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2350 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2351 and describes additional message parts that MUST be encrypted using the token identified by  
2352 this policy assertion.

2353 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2354 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2355 and describes additional message elements that MUST be encrypted using the token identified  
2356 by this policy assertion.

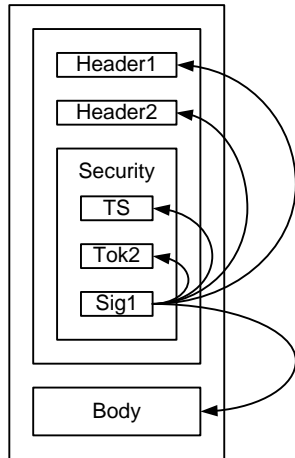
2357 /sp:SupportingTokens/wsp:Policy/sp:ContentEncryptedElements

2358 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2359 and describes additional message elements whose content MUST be encrypted using the token  
2360 identified by this policy assertion.

2361

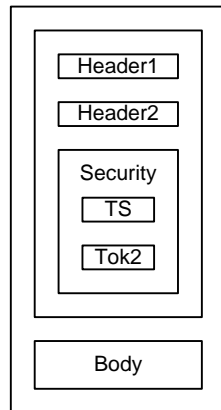
2362 **8.2 SignedSupportingTokens Assertion**

2363 Signed tokens are included in the “message signature” as defined above and MAY OPTIONALLY include  
2364 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token  
2365 (Tok2) is signed by the message signature (Sig1):  
2366



2367 5.

2368 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:  
2369



2370 6.

2371 **Syntax**

```
2372 <sp:SignedSupportingTokens xmlns:sp="..." ... >  
2373   <wsp:Policy xmlns:wsp="...">  
2374     [Token Assertion]+  
2375     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?  
2376     (  
2377       <sp:SignedParts ... > ... </sp:SignedParts> |  
2378       <sp:SignedElements ... > ... </sp:SignedElements> |  
2379       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |  
2380       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |  
2381       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>  
2382     ) *  
2383     ...  
2384   </wsp:Policy>  
2385   ...  
2386 </sp:SignedSupportingTokens>
```

2387

2388 The following describes the attributes and elements listed in the schema outlined above:

2389 /sp:SignedSupportingTokens

2390 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed  
2391 Supporting Tokens] property.

2392 /sp:SignedSupportingTokens/wsp:Policy

2393 This describes additional requirements for satisfying the SignedSupportingTokens assertion.

2394 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]

2395 The policy MUST identify one or more token assertions.

2396 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2397 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2398 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2399 by this policy assertion.

2400 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts

2401 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2402 and describes additional message parts that MUST be included in the signature generated with  
2403 the token identified by this policy assertion.

2404 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements

2405 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2406 and describes additional message elements that MUST be included in the signature generated  
2407 with the token identified by this policy assertion.

2408 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts

2409 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2410 and describes additional message parts that MUST be encrypted using the token identified by  
2411 this policy assertion.

2412 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements

2413 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2414 and describes additional message elements that MUST be encrypted using the token identified  
2415 by this policy assertion.

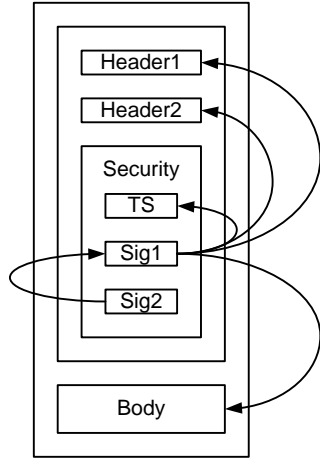
2416 /sp:SignedSupportingTokens/wsp:Policy/sp:ContentEncryptedElements

2417 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2418 and describes additional message elements whose content MUST be encrypted using the token  
2419 identified by this policy assertion.

### 2420 **8.3 EndorsingSupportingTokens Assertion**

2421 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element  
2422 produced from the message signature and MAY OPTIONALLY include additional message parts to sign  
2423 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message  
2424 signature (Sig1):

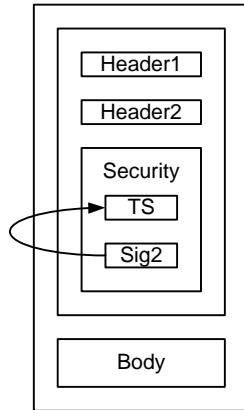
2425



2426 7.

2427 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated  
 2428 below:

2429



2430 8.

2431 **Syntax**

```

2432 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2433   <wsp:Policy xmlns:wsp="...">
2434     [Token Assertion]+
2435     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2436     (
2437       <sp:SignedParts ... > ... </sp:SignedParts> |
2438       <sp:SignedElements ... > ... </sp:SignedElements> |
2439       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2440       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2441       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2442     ) *
2443     ...
2444   </wsp:Policy>
2445   ...
2446 </sp:EndorsingSupportingTokens>
  
```

2447

2448 The following describes the attributes and elements listed in the schema outlined above:

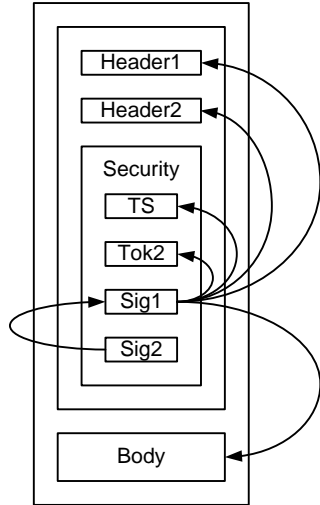
2449 /sp:EndorsingSupportingTokens

2450 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the  
 2451 [Endorsing Supporting Tokens] property.

2452 /sp:EndorsingSupportingTokens/wsp:Policy  
2453 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.  
2454 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]  
2455 The policy MUST identify one or more token assertions.  
2456 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite  
2457 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2458 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2459 by this policy assertion.  
2460 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts  
2461 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2462 and describes additional message parts that MUST be included in the signature generated with  
2463 the token identified by this policy assertion.  
2464 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements  
2465 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2466 and describes additional message elements that MUST be included in the signature generated  
2467 with the token identified by this policy assertion.  
2468 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts  
2469 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2470 and describes additional message parts that MUST be encrypted using the token identified by  
2471 this policy assertion.  
2472 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements  
2473 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2474 and describes additional message elements that MUST be encrypted using the token identified  
2475 by this policy assertion.  
2476 /sp:EndorsingSupportingTokens/wsp:Policy/sp:ContentEncryptedElements  
2477 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2478 and describes additional message elements whose content MUST be encrypted using the token  
2479 identified by this policy assertion.

## 2480 **8.4 SignedEndorsingSupportingTokens Assertion**

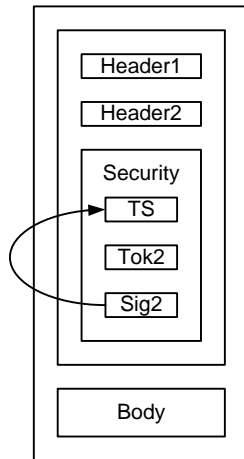
2481 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature  
2482 and are themselves signed by that message signature, that is both tokens (the token used for the  
2483 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY  
2484 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed  
2485 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the  
2486 message signature (Sig1):  
2487



2488 9.

2489 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)  
 2490 SHOULD cover the message timestamp as illustrated below:

2491



2492 10.

2493 **Syntax**

```

2494 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2495   <wsp:Policy xmlns:wsp="...">
2496     [Token Assertion]+
2497     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2498     (
2499       <sp:SignedParts ... > ... </sp:SignedParts> |
2500       <sp:SignedElements ... > ... </sp:SignedElements> |
2501       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2502       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2503       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2504     ) *
2505     ...
2506   </wsp:Policy>
2507   ...
2508 </sp:SignedEndorsingSupportingTokens>
  
```

2509

2510 The following describes the attributes and elements listed in the schema outlined above:

2511 /sp:SignedEndorsingSupportingTokens

2512 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the  
2513 [Signed Endorsing Supporting Tokens] property.

2514 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2515 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2516 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2517 The policy MUST identify one or more token assertions.

2518 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2519 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2520 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2521 by this policy assertion.

2522 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2523 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2524 and describes additional message parts that MUST be included in the signature generated with  
2525 the token identified by this policy assertion.

2526 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2527 This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional  
2528 message elements that MUST be included in the signature generated with the token identified by  
2529 this policy assertion.

2530 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2531 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2532 and describes additional message parts that MUST be encrypted using the token identified by  
2533 this policy assertion.

2534 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2535 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2536 and describes additional message elements that MUST be encrypted using the token identified  
2537 by this policy assertion.

2538 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:ContentEncryptedElements

2539 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2540 and describes additional message elements whose content MUST be encrypted using the token  
2541 identified by this policy assertion.

## 2542 **8.5 SignedEncryptedSupportingTokens Assertion**

2543 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also  
2544 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2545 encrypting the supporting tokens.

2546 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of  
2547 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2548 sp:SignedSupportingTokens assertion.

## 2549 **8.6 EncryptedSupportingTokens Assertion**

2550 Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in the security  
2551 header and MUST be encrypted when they appear in the security header. Element encryption SHOULD  
2552 be used for encrypting these tokens. The encrypted supporting tokens can be added to any SOAP  
2553 message and do not require the “message signature” being present before the encrypted supporting  
2554 tokens are added.

2555 The syntax for the sp:EncryptedSupportingTokens differs from the syntax of sp:SupportingTokens only in  
2556 the name of the assertion itself. All nested policy is as per the sp:SupportingTokens assertion.

2557 The encrypted supporting tokens SHOULD be used only when the sender cannot provide the “message  
2558 signature” and it is RECOMMENDED that the receiver employs some security mechanisms external to  
2559 the message to prevent the spoofing attacks. In all other cases it is RECOMMENDED to use signed  
2560 encrypted supporting tokens instead to ensure that the encrypted tokens are cryptographically bound to  
2561 the message (See section 8.5).

## 2562 **8.7 EndorsingEncryptedSupportingTokens Assertion**

2563 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also  
2564 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2565 encrypting the supporting tokens.

2566 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of  
2567 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2568 sp:EndorsingSupportingTokens assertion.

## 2569 **8.8 SignedEndorsingEncryptedSupportingTokens Assertion**

2570 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section  
2571 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD  
2572 be used for encrypting the supporting tokens.

2573 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of  
2574 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per  
2575 the sp:SignedEndorsingSupportingTokens assertion.

## 2576 **8.9 Interaction between [Token Protection] property and supporting 2577 token assertions**

2578 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that  
2579 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2580 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or  
2581 the [Signature Token] in the Symmetric binding case, covers that token.
- 2582 • Endorsing signatures cover the main signature and the endorsing token.
- 2583 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the  
2584 message signature and once by the endorsing signature.

2585 In addition, signed supporting tokens are covered by the message signature, although this is independent  
2586 of [Token Protection].

## 2587 **8.10 Example**

2588 Example policy containing supporting token assertions:

2589 

```
<!-- Example Endpoint Policy -->
```



```

2590 <wsp:Policy xmlns:wsp="...">
2591   <sp:SymmetricBinding xmlns:sp="...">
2592     <wsp:Policy>
2593       <sp:ProtectionToken>
2594         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2595           <sp:Issuer>...</sp:Issuer>
2596           <sp:RequestSecurityTokenTemplate>
2597             ...
2598           </sp:RequestSecurityTokenTemplate>
2599         </sp:IssuedToken>
2600       </sp:ProtectionToken>
2601       <sp:AlgorithmSuite>
2602         <wsp:Policy>
2603           <sp:Basic256 />
2604         </wsp:Policy>
2605       </sp:AlgorithmSuite>
2606       ...
2607     </wsp:Policy>
2608   </sp:SymmetricBinding>
2609   ...
2610   <sp:SignedSupportingTokens>
2611     <wsp:Policy>
2612       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2613     </wsp:Policy>
2614   </sp:SignedSupportingTokens>
2615   <sp:SignedEndorsingSupportingTokens>
2616     <wsp:Policy>
2617       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2618         <wsp:Policy>
2619           <sp:WssX509v3Token10 />
2620         </wsp:Policy>
2621       </sp:X509Token>
2622     </wsp:Policy>
2623   </sp:SignedEndorsingSupportingTokens>
2624   ...
2625 </wsp:Policy>

```

2626 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be  
2627 included in the security header and covered by the message signature. The  
2628 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the  
2629 security header and covered by the message signature. In addition, a signature over the message  
2630 signature based on the key material associated with the X509 certificate must be included in the security  
2631 header.

2632

---

## 9 WSS: SOAP Message Security Options

2633 There are several OPTIONAL aspects to the WSS: SOAP Message Security specification that are  
2634 independent of the trust and token taxonomies. This section describes another class of properties and  
2635 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The  
2636 assertions defined here MUST apply to [Endpoint Policy Subject].

2637 The properties and assertions dealing with token references defined in this section indicate whether the  
2638 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and  
2639 recipient MAY send a fault if such references are encountered.

2640

2641 Note: This approach is chosen because:

2642 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used  
2643 in a single reference.

2644 B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending  
2645 on which of a series of messages is being secured.

2646

2647 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a  
2648 `wsse:InvalidSecurity` fault.

2649

### 2650 **WSS: SOAP Message Security 1.0 Properties**

#### 2651 **[Direct References]**

2652 This property indicates whether the initiator and recipient MUST be able to process direct token  
2653 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations  
2654 MUST be able to process such references.

2655

#### 2656 **[Key Identifier References]**

2657 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific  
2658 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2659 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST  
2660 NOT generate such references and that the initiator and recipient MAY send a fault if such references are  
2661 encountered. This property has a default value of 'false'.

2662

#### 2663 **[Issuer Serial References]**

2664 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2665 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST  
2666 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT  
2667 generate such references and that the initiator and recipient MAY send a fault if such references are  
2668 encountered. This property has a default value of 'false'.

2669

#### 2670 **[External URI References]**

2671 This boolean property indicates whether the initiator and recipient MUST be able to process references to  
2672 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST  
2673 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2674 generate such references and that the initiator and recipient MAY send a fault if such references are  
2675 encountered. This property has a default value of 'false'.

2676 **[Embedded Token References]**

2677 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2678 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to  
2679 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2680 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2681 This property has a default value of 'false'.

2682

2683 **WSS: SOAP Message Security 1.1 Properties**

2684 **[Thumbprint References]**

2685 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2686 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to  
2687 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2688 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2689 This property has a default value of 'false'.

2690

2691 **[EncryptedKey References]**

2692 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2693 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2694 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2695 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2696 This property has a default value of 'false'.

2697

2698 **[Signature Confirmation]**

2699 This boolean property specifies whether `wss11:SignatureConfirmation` elements SHOULD be  
2700 used as defined in WSS: Soap Message Security 1.1. If the value is 'true',  
2701 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If  
2702 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property  
2703 applies to all signatures that are included in the security header. This property has a default value of  
2704 'false'. This value of this property does not affect the message parts protected by the message signature  
2705 (see the `sp:SignedParts` and `sp:SignedElements` assertions)

2706 **9.1 Wss10 Assertion**

2707 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are  
2708 supported.

2709 **Syntax**

```
2710 <sp:Wss10 xmlns:sp="..." ... >  
2711   <wsp:Policy xmlns:wsp="...">  
2712     <sp:MustSupportRefKeyIdentifier ... /> ?  
2713     <sp:MustSupportRefIssuerSerial ... /> ?  
2714     <sp:MustSupportRefExternalURI ... /> ?  
2715     <sp:MustSupportRefEmbeddedToken ... /> ?  
2716     ...  
2717   </wsp:Policy>  
2718   ...  
2719 </sp:Wss10>
```

2720

2721 The following describes the attributes and elements listed in the schema outlined above:  
 2722 /sp:Wss10  
 2723 This identifies a WSS10 assertion.  
 2724 /sp:Wss10/wsp:Policy  
 2725 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.  
 2726 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2727 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2728 property is set to 'true'.  
 2729 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2730 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2731 property is set to 'true'.  
 2732 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI  
 2733 This OPTIONAL element is a policy assertion indicates that the [External URI References]  
 2734 property is set to 'true'.  
 2735 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken  
 2736 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
 2737 property is set to 'true'.

## 2738 9.2 Wss11 Assertion

2739 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are  
 2740 supported.

### 2741 Syntax

```

2742 <sp:Wss11 xmlns:sp="..." ... >
2743   <wsp:Policy xmlns:wsp="...">
2744     <sp:MustSupportRefKeyIdentifier ... /> ?
2745     <sp:MustSupportRefIssuerSerial ... /> ?
2746     <sp:MustSupportRefExternalURI ... /> ?
2747     <sp:MustSupportRefEmbeddedToken ... /> ?
2748     <sp:MustSupportRefThumbprint ... /> ?
2749     <sp:MustSupportRefEncryptedKey ... /> ?
2750     <sp:RequireSignatureConfirmation ... /> ?
2751     ...
2752   </wsp:Policy>
2753 </sp:Wss11>
  
```

2754  
 2755 The following describes the attributes and elements listed in the schema outlined above:  
 2756 /sp:Wss11  
 2757 This identifies an WSS11 assertion.  
 2758 /sp:Wss11/wsp:Policy  
 2759 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.  
 2760 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2761 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2762 property is set to 'true'.  
 2763 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2764 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2765 property is set to 'true'.

- 2766 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI  
2767 This OPTIONAL element is a policy assertion indicates that the [External URI References]  
2768 property is set to 'true'.
- 2769 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken  
2770 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
2771 property is set to 'true'.
- 2772 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint  
2773 This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property  
2774 is set to 'true'.
- 2775 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey  
2776 This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]  
2777 property is set to 'true'.
- 2778 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation  
2779 This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property  
2780 is set to 'true'.

2781

## 10 WS-Trust Options

2782 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically  
2783 with client and server challenges and entropy behaviors. These assertions relate to interactions with a  
2784 Security Token Service and MAY augment the behaviors defined by the Binding Property Assertions  
2785 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2786

### 2787 **WS-Trust Properties**

#### 2788 **[Client Challenge]**

2789 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a  
2790 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of  
2791 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of  
2792 messages exchanged by the client and service in satisfying the RST. This property has a default value of  
2793 'false'.

2794

#### 2795 **[Server Challenge]**

2796 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a  
2797 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of  
2798 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server MAY  
2799 increase the number of messages exchanged by the client and service in order to accommodate the  
2800 `wst:SignChallengeResponse` element sent by the client to the server in response to the  
2801 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the  
2802 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2803

#### 2804 **[Client Entropy]**

2805 This boolean property indicates whether client entropy is REQUIRED to be used as key material for a  
2806 requested proof token. A value of 'true' indicates that client entropy is REQUIRED. A value of 'false'  
2807 indicates that client entropy is NOT REQUIRED. This property has a default value of 'false'.

2808

#### 2809 **[Server Entropy]**

2810 This boolean property indicates whether server entropy is REQUIRED to be used as key material for a  
2811 requested proof token. A value of 'true' indicates that server entropy is REQUIRED. A value of 'false'  
2812 indicates that server entropy is NOT REQUIRED. This property has a default value of 'false'.

2813 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy  
2814 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm  
2815 Suite] property.

2816

#### 2817 **[Issued Tokens]**

2818 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in  
2819 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'  
2820 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of  
2821 'false'.

#### 2822 **[Collection]**

2823 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A  
2824 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and  
2825 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that  
2826 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default  
2827 value of 'false'.

2828

### 2829 **[Scope Policy 1.5]**

2830 This boolean property indicates whether the wsp:AppliesTo element in the [WS-Policy] 1.5 namespace is  
2831 supported as described in [WS-Trust]. A value of 'true' indicates that the wsp:AppliesTo element in the  
2832 [WS-Policy] 1.5 namespace is supported. A value of 'false' indicates that the wsp:AppliesTo element in  
2833 the [WS-Policy] 1.5 namespace is not supported, the [WS-Policy] 1.2 namespace is used instead in this  
2834 case. This property has a default value of 'false'.

2835

### 2836 **[Interactive Challenge]**

2837 This boolean property indicates whether interactive challenges are supported. A value of 'true' indicates  
2838 that a wst14:InteractiveChallenge element is supported inside of an RSTR sent by the server to the client.  
2839 A value of 'false' indicates that wst14:InteractiveChallenge is not supported. A challenge issued by the  
2840 server may increase the number of messages exchanged by the client and service in order to  
2841 accommodate the wst14:InteractiveChallengeResponse element sent by the client to the server in  
2842 response to the wst14:InteractiveChallenge element. There is an optimization in which a client MAY send  
2843 the wst14:InteractiveChallengeResponse element in an initial RST to the server. A final RSTR containing  
2844 the issued token will follow subsequent to the server receiving the wst14:InteractiveChallengeResponse  
2845 element. This property has a default value of 'false'.

2846

## 2847 **10.1 Trust13 Assertion**

2848 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.

### 2849 **Syntax**

```
2850 <sp:Trust13 xmlns:sp="..." ... >  
2851   <wsp:Policy xmlns:wsp="...">  
2852     <sp:MustSupportClientChallenge ... />?  
2853     <sp:MustSupportServerChallenge ... />?  
2854     <sp:RequireClientEntropy ... />?  
2855     <sp:RequireServerEntropy ... />?  
2856     <sp:MustSupportIssuedTokens ... />?  
2857     <sp:RequireRequestSecurityTokenCollection />?  
2858     <sp:RequireAppliesTo />?  
2859     <sp13:ScopePolicy15 />?  
2860     <sp13:MustSupportInteractiveChallenge />?  
2861     ...  
2862   </wsp:Policy>  
2863   ...  
2864 </sp:Trust13 ... >
```

2865

2866 The following describes the attributes and elements listed in the schema outlined above:

2867 /sp:Trust13

2868         This identifies a Trust13 assertion.

2869 /sp:Trust13/wsp:Policy

2870         This indicates a policy that controls WS-Trust 1.3 options.

2871 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge  
2872 This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set  
2873 to 'true'.

2874 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge  
2875 This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set  
2876 to 'true'.

2877 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy  
2878 This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to  
2879 'true'.

2880 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy  
2881 This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to  
2882 'true'.

2883 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens  
2884 This OPTIONAL element is a policy assertion indicates that the [Issued Tokens] property is set to  
2885 'true'.

2886 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection  
2887 This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to  
2888 'true'.

2889 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo  
2890 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor  
2891 to specify the scope for the issued token using wsp:AppliesTo in the RST.

2892 /sp:Trust13/wsp:Policy/sp13:ScopePolicy15  
2893 This OPTIONAL element is a policy assertion that indicates that the [Scope Policy 1.5] property is  
2894 set to 'true'.

2895 /sp:Trust13/wsp:Policy/sp13:MustSupportInteractiveChallenge  
2896 This optional element is a policy assertion indicates that the [Interactive Challenge] property is set  
2897 to 'true'.



---

## 2898 11 Guidance on creating new assertions and assertion 2899 extensibility

2900 This non-normative appendix provides guidance for designers of new assertions intended for use with this  
2901 specification.

### 2902 11.1 General Design Points

- 2903 • Prefer Distinct Qnames
- 2904 • Parameterize using nested policy where possible.
- 2905 • Parameterize using attributes and/or child elements where necessary.

### 2906 11.2 Detailed Design Guidance

2907 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per  
2908 WS-Policy is performed by matching element Qnames. Matching does not take into account attributes  
2909 that are present on the assertion element. Nor does it take into account child elements except for  
2910 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions  
2911 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2912  
2913 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken  
2914 into account. In general, multiple assertions with distinct Qnames are preferably to a single assertion that  
2915 uses attributes and/or content to distinguish different cases. For example, given two possible assertion  
2916 designs;

2917

2918

2919

2920

2921

2922

2923

2924

2925

2926

2927

2928

2929

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

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

2930 then design 1. Would generally be preferred because it allows the policy matching logic to provide more  
2931 accurate matches between policies.

2932

2933 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct  
2934 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These  
2935 distinct token assertions make policy matching much more useful as less false positives are generated  
2936 when performing policy matching.

2937

2938 There are cases where using attributes or child elements as parameters in assertion design is  
2939 reasonable. Examples include cases when implementations are expected to understand all the values for  
2940 a given parameter and when encoding the parameter information into the assertion QName would result  
2941 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2942 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken  
2943 attribute and implementations are expected to understand the meaning of all 5 values. If this information  
2944 was encoded into the assertion Qnames, each existing token assertion would require five variants, one  
2945 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2946

2947 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For  
2948 example, the token version assertions defined in Section 5 use such an approach. The overall token type  
2949 assertion is parameterized by the nested token version assertions. Policy matching can use these  
2950 parameters to find matches between policies where the broad token type is support by both parties but  
2951 they might not support the same specific versions.

2952

2953 Note, when designing assertions for new token types such assertions SHOULD allow the  
2954 sp:IncludeToken attribute and SHOULD allow nested policy.

2955

2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967  
2968  
2969  
2970  
2971  
2972  
2973

---

## 12 Security Considerations

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

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

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

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

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

2974

---

## 13 Conformance

2975 An implementation conforms to this specification if it satisfies all of the MUST or REQUIRED level  
2976 requirements defined within this specification. A SOAP Node MUST NOT use the XML namespace  
2977 identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is compliant with this  
2978 specification.

2979

2980 This specification references a number of other specifications (see the table above). In order to comply  
2981 with this specification, an implementation MUST implement the portions of referenced specifications  
2982 necessary to comply with the required provisions of this specification. Additionally, the implementation of  
2983 the portions of the referenced specifications that are specifically cited in this specification MUST comply  
2984 with the rules for those portions as established in the referenced specification.

2985 Additionally normative text within this specification takes precedence over normative outlines (as  
2986 described in section 1.4.1), which in turn take precedence over the XML Schema [XML Schema Part 1,  
2987 Part 2] and WSDL [WSDL 1.1] descriptions. That is, the normative text in this specification further  
2988 constrains the schemas and/or WSDL that are part of this specification; and this specification contains  
2989 further constraints on the elements defined in referenced schemas.

2990 This specification defines a number of extensions; compliant services are NOT REQUIRED to implement  
2991 OPTIONAL features defined in this specification. However, if a service implements an aspect of the  
2992 specification, it MUST comply with the requirements specified (e.g. related "MUST" statements). If an  
2993 OPTIONAL message is not supported, then the implementation SHOULD Fault just as it would for any  
2994 other unrecognized/unsupported message. If an OPTIONAL message is supported, then the  
2995 implementation MUST satisfy all of the MUST and REQUIRED sections of the message.

2996

---

## 2997 **Appendix A. Assertions and WS-PolicyAttachment**

2998 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per  
2999 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical  
3000 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any  
3001 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy  
3002 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

### 3003 **A.1 Endpoint Policy Subject Assertions**

#### 3004 **A.1.1 Security Binding Assertions**

- 3005 [TransportBinding Assertion](#) (Section 7.3)
- 3006 [SymmetricBinding Assertion](#) (Section 7.4)
- 3007 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 3008 **A.1.2 Token Assertions**

- 3009 [SupportingTokens Assertion](#) (Section 8.1)
- 3010 [SignedSupportingTokens Assertion](#) (Section 8.2)
- 3011 [EndorsingSupportingTokens Assertion](#) (Section 8.3)
- 3012 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)
- 3013 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)
- 3014 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)
- 3015 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

#### 3016 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

- 3017 [Wss10 Assertion](#) (Section 9.1)

#### 3018 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

- 3019 [Wss11 Assertion](#) (Section 9.2)

#### 3020 **A.1.5 Trust 1.0 Assertions**

- 3021 [Trust13 Assertion](#) (Section 10.1)

### 3022 **A.2 Operation Policy Subject Assertions**

#### 3023 **A.2.1 Security Binding Assertions**

- 3024 [SymmetricBinding Assertion](#) (Section 7.4)
- 3025 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 3026 **A.2.2 Supporting Token Assertions**

- 3027 [SupportingTokens Assertion](#) (Section 8.1)
- 3028 [SignedSupportingTokens Assertion](#) (Section 8.2)

3029	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
3030	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
3031	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
3032	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
3033	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

## 3034 **A.3 Message Policy Subject Assertions**

### 3035 **A.3.1 Supporting Token Assertions**

3036	<a href="#">SupportingTokens Assertion</a>	(Section 8.1)
3037	<a href="#">SignedSupportingTokens Assertion</a>	(Section 8.2)
3038	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
3039	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
3040	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
3041	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
3042	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

### 3043 **A.3.2 Protection Assertions**

3044	<a href="#">SignedParts Assertion</a>	(Section 4.1.1)
3045	<a href="#">SignedElements Assertion</a>	(Section 4.1.2)
3046	<a href="#">EncryptedParts Assertion</a>	(Section 4.2.1)
3047	<a href="#">EncryptedElements Assertion</a>	(Section 4.2.2)
3048	<a href="#">ContentEncryptedElements Assertion</a>	(Section 4.2.3)
3049	<a href="#">RequiredElements Assertion</a>	(Section 4.3.1)
3050	<a href="#">RequiredParts Assertion</a>	(Section 4.3.2)

## 3051 **A.4 Assertions With Undefined Policy Subject**

3052 The assertions listed in this section do not have a defined policy subject because they appear nested  
 3053 inside some other assertion which does have a defined policy subject. This list is derived from nested  
 3054 assertions in the specification that have independent sections. It is not a complete list of nested  
 3055 assertions. Many of the assertions previously listed in this appendix as well as the ones below have  
 3056 additional nested assertions.

### 3057 **A.4.1 General Assertions**

3058	<a href="#">AlgorithmSuite Assertion</a>	(Section 7.1)
3059	<a href="#">Layout Assertion</a>	(Section 7.2)

### 3060 **A.4.2 Token Usage Assertions**

3061 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)  
 3062 assertions.

### 3063 **A.4.3 Token Assertions**

3064	<a href="#">UsernameToken Assertion</a>	(Section 5.3.1)
------	---	-----------------

3065	<a href="#">IssuedToken Assertion</a>	(Section 5.3.2)
3066	<a href="#">X509Token Assertion</a>	(Section 5.3.3)
3067	<a href="#">KerberosToken Assertion</a>	(Section 5.3.4)
3068	<a href="#">SpnegoContextToken Assertion</a>	(Section 5.3.5)
3069	<a href="#">SecurityContextToken Assertion</a>	(Section 5.3.6)
3070	<a href="#">SecureConversationToken Assertion</a>	(Section 5.3.7)
3071	<a href="#">SamlToken Assertion</a>	(Section 5.3.8)
3072	<a href="#">RelToken Assertion</a>	(Section 5.3.9)
3073	<a href="#">HttpsToken Assertion</a>	(Section 5.3.10)

## Appendix B. Issued Token Policy

3074

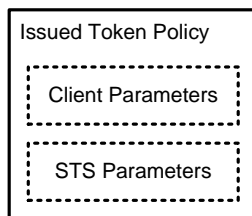
3075 The section provides further detail about behavior associated with the IssuedToken assertion in section  
3076 5.3.2.

3077

3078 The issued token security model involves a three-party setup. There's a target Server, a Client, and a  
3079 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from  
3080 STS to Client. Policy MAY be embedded inside an Issued Token assertion, or acquired out-of-band.  
3081 There MAY be an explicit trust relationship between the Server and the STS. There MUST be a trust  
3082 relationship between the Client and the STS.

3083

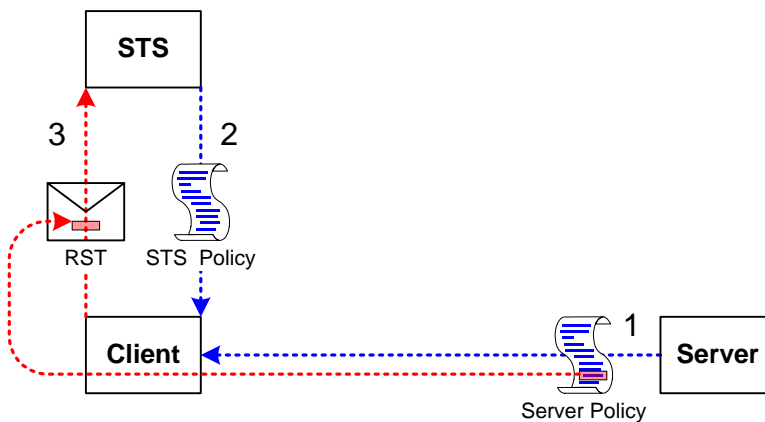
3084 The Issued Token policy assertion includes two parts: 1) client-specific parameters that MUST be  
3085 understood and processed by the client and 2) STS specific parameters which are to be processed by the  
3086 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



3087 11.

3088 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server  
3089 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are  
3090 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the  
3091 RST request sent by the Client to the STS as illustrated in the figure below.

3092



3093 12.

3094 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to  
3095 formulate the RST request and will include any security-specific requirements of the STS.

3096

3097 The Client MAY augment or replace the contents of the RST made to the STS based on the Client-  
3098 specific parameters received from the Issued Token policy assertion contained in the Server policy, from  
3099 policy it received for the STS, or any other local parameters.

3100



3101 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The  
3102 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along  
3103 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST  
3104 request sent by the Client to the STS following the protocol defined in WS-Trust.

3105

3106 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)  
3107 [Trust\]](#). All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship  
3108 which specifies some or all of the conditions and constraints for issued tokens.

---

## 3109 Appendix C. Strict Security Header Layout Examples

3110 The following sections describe the security header layout for specific bindings when applying the 'Strict'  
3111 layout rules defined in Section 6.7.

### 3112 C.1 Transport Binding

3113 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

#### 3114 C.1.1 Policy

3115 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport  
3116 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username  
3117 token attached to the message, and finally an X509 token attached to the message and endorsing the  
3118 message signature. No message protection requirements are described since the transport covers all  
3119 message parts.

```
3120 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3121   <sp:TransportBinding>
3122     <wsp:Policy>
3123       <sp:TransportToken>
3124         <wsp:Policy>
3125           <sp:HttpsToken />
3126         </wsp:Policy>
3127       </sp:TransportToken>
3128       <sp:AlgorithmSuite>
3129         <wsp:Policy>
3130           <sp:Basic256 />
3131         </wsp:Policy>
3132       </sp:AlgorithmSuite>
3133       <sp:Layout>
3134         <wsp:Policy>
3135           <sp:Strict />
3136         </wsp:Policy>
3137       </sp:Layout>
3138       <sp:IncludeTimestamp />
3139     </wsp:Policy>
3140   </sp:TransportBinding>
3141   <sp:SignedSupportingTokens>
3142     <wsp:Policy>
3143       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3144     </wsp:Policy>
3145   </sp:SignedSupportingTokens>
3146   <sp:SignedEndorsingSupportingTokens>
3147     <wsp:Policy>
3148       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3149         <wsp:Policy>
3150           <sp:WssX509v3Token10 />
3151         </wsp:Policy>
3152       </sp:X509Token>
3153     </wsp:Policy>
3154   </sp:SignedEndorsingSupportingTokens>
3155   <sp:Wss11>
3156     <sp:RequireSignatureConfirmation />
3157   </sp:Wss11>
3158 </wsp:Policy>
```

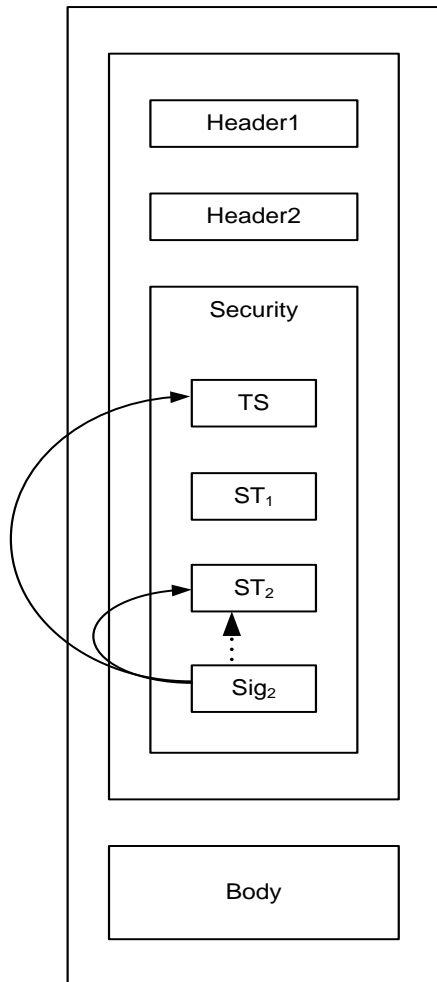
3159 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3160 header layout for this binding.

3161 **C.1.2 Initiator to Recipient Messages**

3162 Messages sent from initiator to recipient have the following layout for the security header:

- 3163 1. A `wsu:Timestamp` element.
- 3164 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 3165 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the  
3166 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1  
3167 above and **SHOULD** cover any other unique identifier for the message in order to prevent  
3168 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If  
3169 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a  
3170 Derived Key Token, based on the supporting token, appears between the supporting token and  
3171 the signature.
- 3172 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each  
3173 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least  
3174 some other unique identifier for the message in order to prevent replays. If [Token Protection] is  
3175 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the  
3176 supporting token is associated with a symmetric key, then a Derived Key Token, based on the  
3177 supporting token, appears before the signature.

3178 The following diagram illustrates the security header layout for the initiator to recipient message:



3179 13.

3180 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The  
3181 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>,  
3182 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST<sub>2</sub>.  
3183 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering  
3184 of the items in the security header follows the most optimal layout for a receiver to process its contents.

3185 *Example:*

3186 Initiator to recipient message

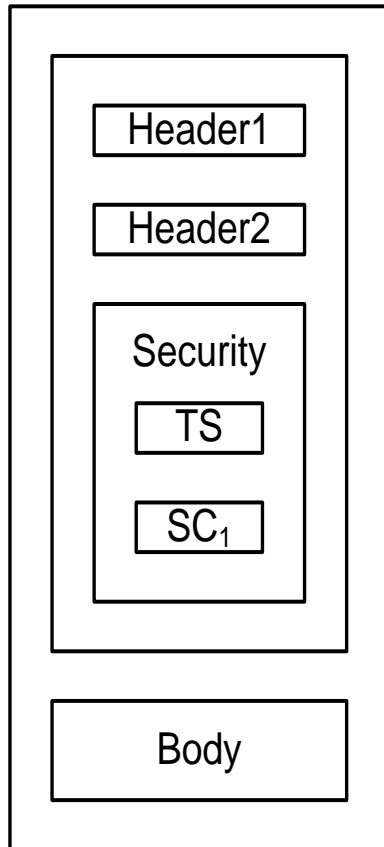
```
3187 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">  
3188   <S:Header>  
3189     ...  
3190     <wsse:Security>  
3191       <wsu:Timestamp wsu:Id="timestamp">  
3192         <wsu:Created>[datetime]</wsu:Created>  
3193         <wsu:Expires>[datetime]</wsu:Expires>  
3194       </wsu:Timestamp>  
3195       <wsse:UsernameToken wsu:Id='SomeSignedToken' >  
3196         ...  
3197       </wsse:UsernameToken>  
3198       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >  
3199         ...  
3200       </wsse:BinarySecurityToken>  
3201       <ds:Signature>  
3202         <ds:SignedInfo>  
3203           <ds:References>  
3204             <ds:Reference URI="#timestamp" />  
3205             <ds:Reference URI="#SomeSignedEndorsingToken" />  
3206           </ds:References>  
3207         </ds:SignedInfo>  
3208         <ds:SignatureValue>...</ds:SignatureValue>  
3209         <ds:KeyInfo>  
3210           <wsse:SecurityTokenReference>  
3211             <wsse:Reference URI="#SomeSignedEndorsingToken" />  
3212           </wsse:SecurityTokenReference>  
3213         </ds:KeyInfo>  
3214       </ds:Signature>  
3215     ...  
3216   </wsse:Security>  
3217   ...  
3218 </S:Header>  
3219 <S:Body>  
3220   ...  
3221 </S:Body>  
3222 </S:Envelope>
```

### 3223 **C.1.3 Recipient to Initiator Messages**

3224 Messages sent from recipient to initiator have the following layout for the security header:

- 3225 1. A `wsu:Timestamp` element.
- 3226 2. If the [Signature Confirmation] property has a value of 'true', then a  
3227 `wsse11:SignatureConfirmation` element for each signature in the corresponding message  
3228 sent from initiator to recipient. If there are no signatures in the corresponding message from the  
3229 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no `Value`  
3230 attribute.

3231 The following diagram illustrates the security header layout for the recipient to initiator message:



3232 14.

3233 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One  
 3234 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial  
 3235 message illustrated previously is included. In general, the ordering of the items in the security header  
 3236 follows the most optimal layout for a receiver to process its contents.

3237 *Example:*

3238 Recipient to initiator message

```

3239 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3240   <S:Header>
3241     ...
3242     <wsse:Security>
3243       <wsu:Timestamp wsu:Id="timestamp">
3244         <wsu:Created>[datetime]</wsu:Created>
3245         <wsu:Expires>[datetime]</wsu:Expires>
3246       </wsu:Timestamp>
3247       <wsse11:SignatureConfirmation Value="..." />
3248     ...
3249   </wsse:Security>
3250   ...
3251 </S:Header>
3252 <S:Body>
3253   ...
3254 </S:Body>
3255 </S:Envelope>
  
```

## 3256 C.2 Symmetric Binding

3257 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

## 3258 C.2.1 Policy

3259 The following example shows a policy indicating a Symmetric Binding, a symmetric key based  
3260 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message  
3261 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in  
3262 the message signature and the supporting signatures, a username token attached to the message, and  
3263 finally an X509 token attached to the message and endorsing the message signature. Minimum message  
3264 protection requirements are described as well.

```
3265 <!-- Example Endpoint Policy -->
3266 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3267   <sp:SymmetricBinding>
3268     <wsp:Policy>
3269       <sp:ProtectionToken>
3270         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3271           <sp:Issuer>...</sp:Issuer>
3272           <sp:RequestSecurityTokenTemplate>
3273             ...
3274           </sp:RequestSecurityTokenTemplate>
3275         </sp:IssuedToken>
3276       </sp:ProtectionToken>
3277       <sp:AlgorithmSuite>
3278         <wsp:Policy>
3279           <sp:Basic256 />
3280         </wsp:Policy>
3281       </sp:AlgorithmSuite>
3282       <sp:Layout>
3283         <wsp:Policy>
3284           <sp:Strict />
3285         </wsp:Policy>
3286       </sp:Layout>
3287       <sp:IncludeTimestamp />
3288       <sp:EncryptBeforeSigning />
3289       <sp:EncryptSignature />
3290       <sp:ProtectTokens />
3291     </wsp:Policy>
3292   </sp:SymmetricBinding>
3293   <sp:SignedSupportingTokens>
3294     <wsp:Policy>
3295       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3296     </wsp:Policy>
3297   </sp:SignedSupportingTokens>
3298   <sp:SignedEndorsingSupportingTokens>
3299     <wsp:Policy>
3300       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3301         <wsp:Policy>
3302           <sp:WssX509v3Token10 />
3303         </wsp:Policy>
3304       </sp:X509Token>
3305     </wsp:Policy>
3306   </sp:SignedEndorsingSupportingTokens>
3307   <sp:Wss11>
3308     <wsp:Policy>
3309       <sp:RequireSignatureConfirmation />
3310     </wsp:Policy>
3311   </sp:Wss11>
3312 </wsp:Policy>
3313
```

```

3314 <!-- Example Message Policy -->
3315 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3316   <sp:SignedParts>
3317     <sp:Header Name="Header1" Namespace="..." />
3318     <sp:Header Name="Header2" Namespace="..." />
3319     <sp:Body/>
3320   </sp:SignedParts>
3321   <sp:EncryptedParts>
3322     <sp:Header Name="Header2" Namespace="..." />
3323     <sp:Body/>
3324   </sp:EncryptedParts>
3325 </wsp:Policy>

```

3327 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3328 header layout for this binding.

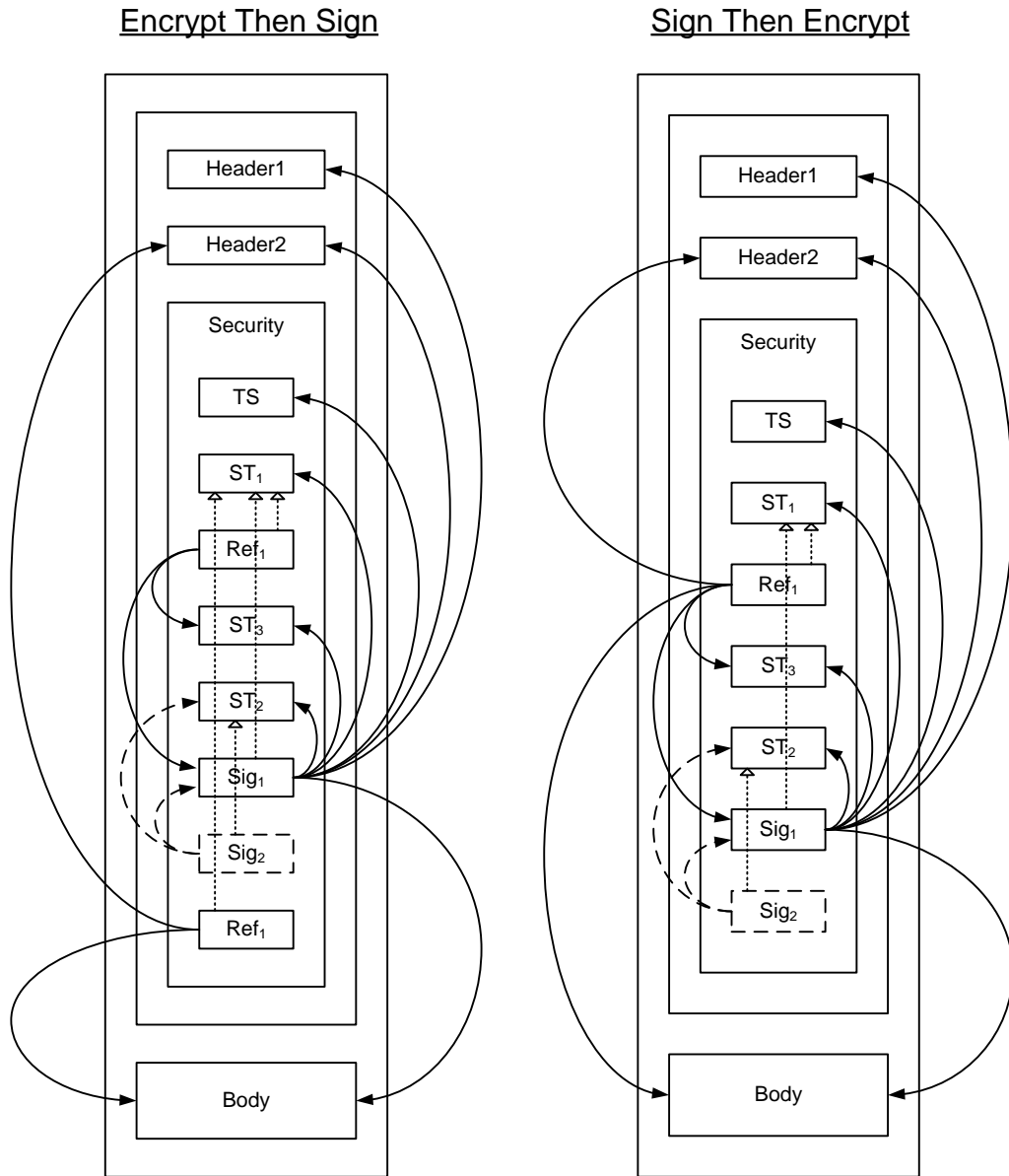
## 3329 C.2.2 Initiator to Recipient Messages

3330 Messages sent from initiator to recipient have the following layout for the security header:

- 3331 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3332 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or  
3333 `.../IncludeToken/Always`, then the [Encryption Token].
- 3334 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3335 Derived Key Token is used for encryption.
- 3336 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3337 reference list MUST include a reference to the message signature. If [Protection Order] is  
3338 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3339 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3340 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3341 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]  
3342 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or  
3343 `.../IncludeToken/Always`.
- 3344 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3345 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the  
3346 [Signature Token].
- 3347 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This  
3348 Derived Key Token is used for signature.
- 3349 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of  
3350 whether they are included in the message, and any message parts specified in SignedParts  
3351 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature  
3352 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in  
3353 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3354 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing  
3355 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]  
3356 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the  
3357 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the  
3358 endorsing token, appears before the signature.
- 3359 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message  
3360 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
3361 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3362 above.

3363

3364 The following diagram illustrates the security header layout for the initiator to recipient message:



3365 15.

3366 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3367 The dashed arrows on the left from the box labeled Sig<sub>2</sub> indicate the parts signed by the supporting token  
 3368 labeled ST<sub>2</sub>, namely the message signature labeled Sig<sub>1</sub> and the token used as the basis for the  
 3369 signature labeled ST<sub>2</sub>. The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts  
 3370 encrypted using a key based on the Shared Secret Token labeled ST<sub>1</sub>. The dotted arrows inside the box  
 3371 labeled Security indicate the token that was used as the basis for each cryptographic operation. In  
 3372 general, the ordering of the items in the security header follows the most optimal layout for a receiver to  
 3373 process its contents.

3374 *Example:*

3375 Initiator to recipient message using EncryptBeforeSigning:



```
3376 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3377     xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3378     xmlns:xenc="..." xmlns:ds="...">
3379 <S:Header>
3380   <x:Header1 wsu:Id="Header1" >
3381     ...
3382   </x:Header1>
3383
```

```

3384 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3385   <!-- Plaintext Header2
3386   <x:Header2 wsu:Id="Header2" >
3387     ...
3388   </x:Header2>
3389   -->
3390   ...
3391 </wsse1:EncryptedHeader>
3392 ...
3393 <wsse:Security>
3394   <wsu:Timestamp wsu:Id="Timestamp">
3395     <wsu:Created>...</wsu:Created>
3396     <wsu:Expires>...</wsu:Expires>
3397   </wsu:Timestamp>
3398   <saml:Assertion AssertionId="_SharedSecretToken" ...>
3399     ...
3400   </saml:Assertion>
3401   <xenc:ReferenceList>
3402     <xenc:DataReference URI="#enc_Signature" />
3403     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3404     ...
3405   </xenc:ReferenceList>
3406   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3407     <!-- Plaintext UsernameToken
3408     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3409       ...
3410     </wsse:UsernameToken>
3411     -->
3412     ...
3413     <ds:KeyInfo>
3414       <wsse:SecurityTokenReference>
3415         <wsse:Reference URI="#_SharedSecretToken" />
3416       </wsse:SecurityTokenReference>
3417     </ds:KeyInfo>
3418   </xenc:EncryptedData>
3419   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3420     ...
3421   </wsse:BinarySecurityToken>
3422   <xenc:EncryptedData ID="enc_Signature">
3423     <!-- Plaintext Signature
3424     <ds:Signature Id="Signature">
3425       <ds:SignedInfo>
3426         <ds:References>
3427           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3428           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3429           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3430           <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3431           <ds:Reference URI="#Header1" >...</ds:Reference>
3432           <ds:Reference URI="#Header2" >...</ds:Reference>
3433           <ds:Reference URI="#Body" >...</ds:Reference>
3434         </ds:References>
3435       </ds:SignedInfo>
3436     <ds:SignatureValue>...</ds:SignatureValue>
3437     <ds:KeyInfo>
3438       <wsse:SecurityTokenReference>
3439         <wsse:Reference URI="#_SharedSecretToken" />
3440       </wsse:SecurityTokenReference>
3441     </ds:KeyInfo>
3442   </ds:Signature>
3443   -->
3444   ...
3445   <ds:KeyInfo>
3446     <wsse:SecurityTokenReference>
3447       <wsse:Reference URI="#_SharedSecretToken" />

```

```

3448     </wsse:SecurityTokenReference>
3449   </ds:KeyInfo>
3450 </xenc:EncryptedData>
3451 <ds:Signature>
3452   <ds:SignedInfo>
3453     <ds:References>
3454       <ds:Reference URI="#Signature" >...</ds:Reference>
3455       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3456     </ds:References>
3457   </ds:SignedInfo>
3458 <ds:SignatureValue>...</ds:SignatureValue>
3459 <ds:KeyInfo>
3460   <wsse:SecurityTokenReference>
3461     <wsse:Reference URI="#SomeSupportingToken" />
3462   </wsse:SecurityTokenReference>
3463 </ds:KeyInfo>
3464 </ds:Signature>
3465 <xenc:ReferenceList>
3466   <xenc:DataReference URI="#enc_Body" />
3467   <xenc:DataReference URI="#enc_Header2" />
3468   ...
3469 </xenc:ReferenceList>
3470 </wsse:Security>
3471 </S:Header>
3472 <S:Body wsu:Id="Body">
3473   <xenc:EncryptedData Id="enc_Body">
3474     ...
3475     <ds:KeyInfo>
3476       <wsse:SecurityTokenReference>
3477         <wsse:Reference URI="#_SharedSecretToken" />
3478       </wsse:SecurityTokenReference>
3479     </ds:KeyInfo>
3480   </xenc:EncryptedData>
3481 </S:Body>
3482 </S:Envelope>

```

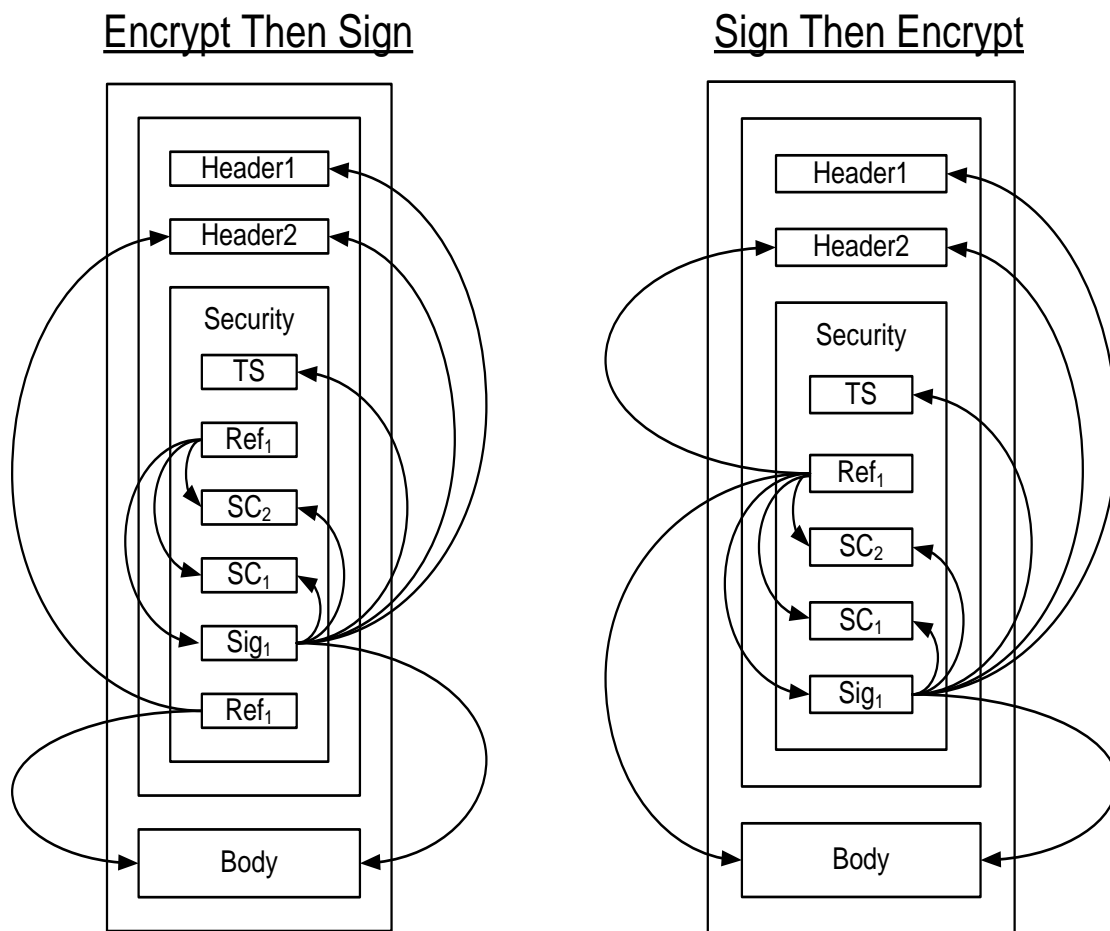
### 3483 C.2.3 Recipient to Initiator Messages

3484 Messages send from recipient to initiator have the following layout for the security header:

- 3485 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3486 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the  
3487 [Encryption Token].
- 3488 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3489 Derived Key Token is used for encryption.
- 3490 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3491 reference list MUST include a reference to the message signature from 6 below, and the  
3492 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is  
3493 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3494 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3495 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3496 above.
- 3497 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each  
3498 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3499 in the corresponding message from the initiator to the recipient, then a  
3500 `wss11:SignatureConfirmation` element with no Value attribute.
- 3501 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3502 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].

- 3503 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This  
 3504 Derived Key Token is used for signature.
- 3505 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation  
 3506 elements from 5 above, and all the message parts specified in SignedParts assertions in the  
 3507 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]  
 3508 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token  
 3509 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 3510 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message  
 3511 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
 3512 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption  
 3513 Token].

3514 The following diagram illustrates the security header layout for the recipient to initiator message:



3515 16.  
 3516

3517 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3518 The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts encrypted using a key based  
 3519 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two  
 3520 wssell:SignatureConfirmation elements labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures  
 3521 in the initial message illustrated previously is included. In general, the ordering of the items in the security  
 3522 header follows the most optimal layout for a receiver to process its contents. The rules used to determine  
 3523 this ordering are described in Appendix C.

3524 *Example:*

3525 Recipient to initiator message using EncryptBeforeSigning:

```
3526 <S:Envelope>
3527   <S:Header>
3528     <x:Header1 wsu:Id="Header1" >
3529       ...
3530     </x:Header1>
3531     <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3532       <!-- Plaintext Header2
3533       <x:Header2 wsu:Id="Header2" >
3534         ...
3535       </x:Header2>
3536       -->
3537       ...
3538     </wsse11:EncryptedHeader>
3539     ...
3540   <wsse:Security>
3541     <wsu:Timestamp wsu:Id="Timestamp">
3542       <wsu:Created>...</wsu:Created>
3543       <wsu:Expires>...</wsu:Expires>
3544     </wsu:Timestamp>
3545     <xenc:ReferenceList>
3546       <xenc:DataReference URI="#enc_Signature" />
3547       <xenc:DataReference URI="#enc_SigConf1" />
3548       <xenc:DataReference URI="#enc_SigConf2" />
3549       ...
3550     </xenc:ReferenceList>
3551     <xenc:EncryptedData ID="enc_SigConf1" >
3552       <!-- Plaintext SignatureConfirmation
3553       <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3554         ...
3555       </wsse11:SignatureConfirmation>
3556       -->
3557       ...
3558     </xenc:EncryptedData>
3559     <xenc:EncryptedData ID="enc_SigConf2" >
3560       <!-- Plaintext SignatureConfirmation
3561       <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3562         ...
3563       </wsse11:SignatureConfirmation>
3564       -->
3565       ...
3566     </xenc:EncryptedData>
```

```

3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
<xenc:EncryptedData Id="enc_Signature">
  <!-- Plaintext Signature
  <ds:Signature Id="Signature">
    <ds:SignedInfo>
      <ds:References>
        <ds:Reference URI="#Timestamp" >...</ds:Reference>
        <ds:Reference URI="#SigConf1" >...</ds:Reference>
        <ds:Reference URI="#SigConf2" >...</ds:Reference>
        <ds:Reference URI="#Header1" >...</ds:Reference>
        <ds:Reference URI="#Header2" >...</ds:Reference>
        <ds:Reference URI="#Body" >...</ds:Reference>
      </ds:References>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
-->
</xenc:EncryptedData>
...
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#_SomeIssuedToken" />
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:EncryptedData>
<xenc:ReferenceList>
  <xenc:DataReference URI="#enc_Body" />
  <xenc:DataReference URI="#enc_Header2" />
  ...
</xenc:ReferenceList>
</xenc:EncryptedData>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="Body">
  <xenc:EncryptedData Id="enc_Body">
    ...
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SomeIssuedToken" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </xenc:EncryptedData>
</S:Body>
</S:Envelope>

```

## 3616 C.3 Asymmetric Binding

3617 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

### 3618 C.3.1 Policy

3619 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator  
3620 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the  
3621 message parts before signing, a requirement to encrypt the message signature, a requirement to include  
3622 tokens in the message signature and the supporting signatures, a requirement to include  
3623 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3624 an X509 token attached to the message and endorsing the message signature. Minimum message  
3625 protection requirements are described as well.

```
3626 <!-- Example Endpoint Policy -->  
3627 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
3628   <sp:AsymmetricBinding>  
3629     <wsp:Policy>  
3630       <sp:RecipientToken>  
3631         <wsp:Policy>  
3632           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3633         </wsp:Policy>  
3634       </sp:RecipientToken>  
3635       <sp:InitiatorToken>  
3636         <wsp:Policy>  
3637           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3638         </wsp:Policy>  
3639       </sp:InitiatorToken>  
3640     <sp:AlgorithmSuite>  
3641       <wsp:Policy>  
3642         <sp:Basic256 />  
3643       </wsp:Policy>  
3644     </sp:AlgorithmSuite>  
3645     <sp:Layout>  
3646       <wsp:Policy>  
3647         <sp:Strict />  
3648       </wsp:Policy>  
3649     </sp:Layout>  
3650     <sp:IncludeTimestamp />  
3651     <sp:EncryptBeforeSigning />  
3652     <sp:EncryptSignature />  
3653     <sp:ProtectTokens />  
3654   </wsp:Policy>  
3655 </sp:AsymmetricBinding>  
3656 <sp:SignedEncryptedSupportingTokens>  
3657   <wsp:Policy>  
3658     <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />  
3659   </wsp:Policy>  
3660 </sp:SignedEncryptedSupportingTokens>  
3661 <sp:SignedEndorsingSupportingTokens>  
3662   <wsp:Policy>  
3663     <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">  
3664       <wsp:Policy>  
3665         <sp:WssX509v3Token10 />  
3666       </wsp:Policy>  
3667     </sp:X509Token>  
3668   </wsp:Policy>  
3669 </sp:SignedEndorsingSupportingTokens>  
3670 <sp:Wss11>  
3671   <wsp:Policy>  
3672     <sp:RequireSignatureConfirmation />  
3673   </wsp:Policy>  
3674 </sp:Wss11>  
3675 </wsp:Policy>  
3676
```

3677

```
3678 <!-- Example Message Policy -->
3679 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3680   <sp:SignedParts>
3681     <sp:Header Name="Header1" Namespace="..." />
3682     <sp:Header Name="Header2" Namespace="..." />
3683     <sp:Body/>
3684   </sp:SignedParts>
3685   <sp:EncryptedParts>
3686     <sp:Header Name="Header2" Namespace="..." />
3687     <sp:Body/>
3688   </sp:EncryptedParts>
3689 </wsp:All>
```

3690  
3691 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3692 header layout for this binding.

### 3693 **C.3.2 Initiator to Recipient Messages**

3694 Messages sent from initiator to recipient have the following layout:

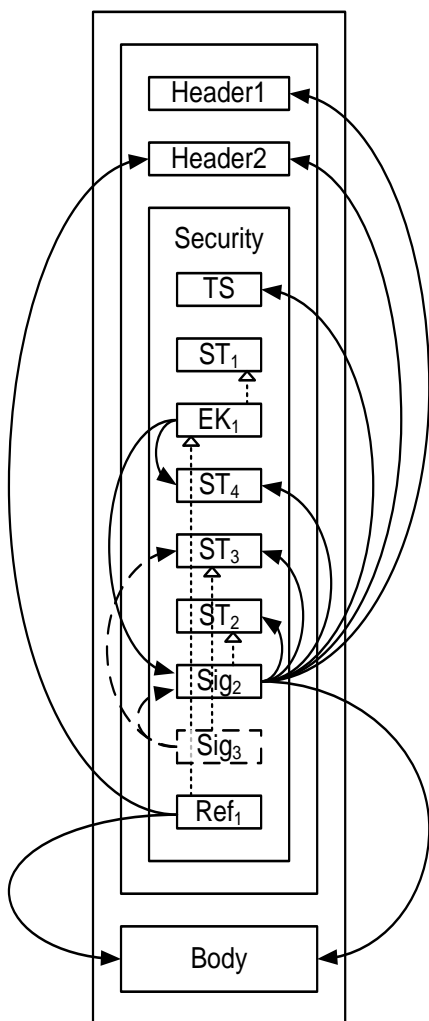
- 3695 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3696 2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3697 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
- 3698 3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3699 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3700 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3701 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If  
3702 [Signature Protection] is 'true' then the reference list MUST contain a reference to the message  
3703 signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message  
3704 signature.
- 3705 4. Any tokens from the supporting tokens properties (as defined in section 8) whose  
3706 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3707 5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3708 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
- 3709 6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from  
3710 1 above, any tokens from 4 above regardless of whether they are included in the message, and  
3711 any message parts specified in `SignedParts` assertions in the policy. If [Token Protection] is 'true',  
3712 the signature MUST also cover the [Initiator Token] regardless of whether it is included in the  
3713 message.
- 3714 7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting  
3715 Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a  
3716 symmetric key, then a Derived Key Token, based on the supporting token, appears before the  
3717 signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token  
3718 regardless of whether it is included in the message.
- 3719 8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
3720 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
3721 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
3722 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions  
3723 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
3724 element from 3 above.

3725

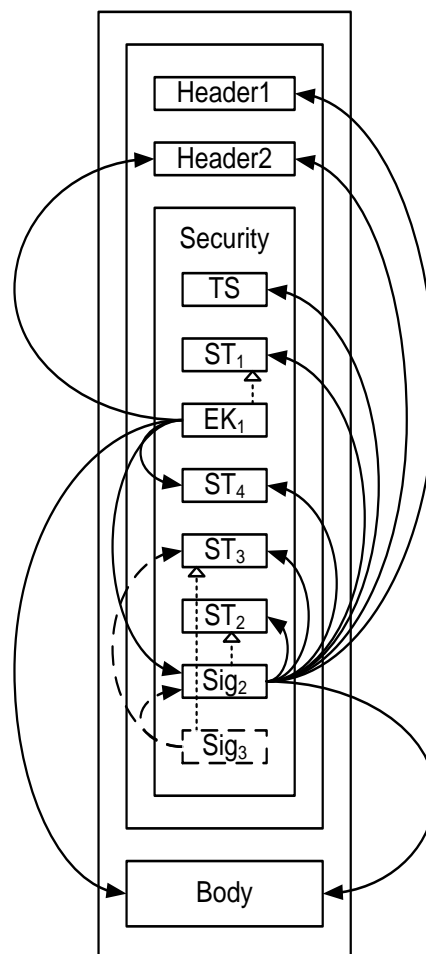


3726 The following diagram illustrates the security header layout for the initiator to recipient messages:

### Encrypt Then Sign



### Sign Then Encrypt



3727 17.  
3728

3729 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3730 using the [Initiator Token] labeled ST<sub>2</sub>. The dashed arrows on the left from the box labeled Sig<sub>3</sub> indicate  
 3731 the parts signed by the supporting token ST<sub>3</sub>, namely the message signature Sig<sub>2</sub> and the token used as  
 3732 the basis for the signature labeled ST<sub>3</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate references  
 3733 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on the left  
 3734 from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained in the  
 3735 encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token used as  
 3736 the basis for each cryptographic operation. In general, the ordering of the items in the security header  
 3737 follows the most optimal layout for a receiver to process its contents. The rules used to determine this  
 3738 ordering are described in Appendix C.

3739

3740 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted  
 3741 key contains an external reference to the token containing the encryption key. The diagram illustrates  
 3742 how one might attach a security token related to the encrypted key for completeness. One possible use-

3743 case for this approach might be a stack which does not support the STR Dereferencing Transform, but  
3744 wishes to include the encryption token in the message signature.

3745 Initiator to recipient message *Example*

3746 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3747   xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3748   <S:Header>
3749     <x:Header1 wsu:Id="Header1" >
3750     ...
3751   </x:Header1>
3752   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3753     <!-- Plaintext Header2
3754     <x:Header2 wsu:Id="Header2" >
3755     ...
3756   </x:Header2>
3757   -->
3758   ...
3759 </wssell1:EncryptedHeader>
3760 ...
3761 <wsse:Security>
3762   <wsu:Timestamp wsu:Id="Timestamp">
3763     <wsu:Created>...</wsu:Created>
3764     <wsu:Expires>...</wsu:Expires>
3765   </wsu:Timestamp>
3766   <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3767   ...
3768 </wsse:BinarySecurityToken>
3769   <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3770   ...
3771   <xenc:ReferenceList>
3772     <xenc:DataReference URI="#enc_Signature" />
3773     <xenc:DataReference URI="#enc_SomeUsernameToken" />
3774     ...
3775   </xenc:ReferenceList>
3776 </xenc:EncryptedKey>
3777   <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3778     <!-- Plaintext UsernameToken
3779     <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3780     ...
3781   </wsse:UsernameToken>
3782   -->
3783   ...
3784 </xenc:EncryptedData>
3785   <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3786   ...
3787 </wsse:BinarySecurityToken>
3788   <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3789   ...
3790 </wsse:BinarySecurityToken>
3791   <xenc:EncryptedData ID="enc_Signature">
3792     <!-- Plaintext Signature
3793     <ds:Signature Id="Signature">
3794       <ds:SignedInfo>
3795         <ds:References>
3796           <ds:Reference URI="#Timestamp" >...</ds:Reference>
3797           <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3798           <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3799           <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3800           <ds:Reference URI="#Header1" >...</ds:Reference>
3801           <ds:Reference URI="#Header2" >...</ds:Reference>
3802           <ds:Reference URI="#Body" >...</ds:Reference>
3803         </ds:References>
3804       </ds:SignedInfo>
3805     <ds:SignatureValue>...</ds:SignatureValue>
3806     <ds:KeyInfo>
3807       <wsse:SecurityTokenReference>
3808         <wsse:Reference URI="#InitiatorToken" />
3809       </wsse:SecurityTokenReference>
3810     </ds:KeyInfo>

```

```

3811     </ds:Signature>
3812     -->
3813     ...
3814 </xenc:EncryptedData>
3815 <ds:Signature>
3816   <ds:SignedInfo>
3817     <ds:References>
3818       <ds:Reference URI="#Signature" >...</ds:Reference>
3819       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3820     </ds:References>
3821   </ds:SignedInfo>
3822   <ds:SignatureValue>...</ds:SignatureValue>
3823   <ds:KeyInfo>
3824     <wsse:SecurityTokenReference>
3825       <wsse:Reference URI="#SomeSupportingToken" />
3826     </wsse:SecurityTokenReference>
3827   </ds:KeyInfo>
3828 </ds:Signature>
3829 <xenc:ReferenceList>
3830   <xenc:DataReference URI="#enc_Body" />
3831   <xenc:DataReference URI="#enc_Header2" />
3832   ...
3833 </xenc:ReferenceList>
3834 </wsse:Security>
3835 </S:Header>
3836 <S:Body wsu:Id="Body">
3837   <xenc:EncryptedData Id="enc_Body">
3838     ...
3839     <ds:KeyInfo>
3840       <wsse:SecurityTokenReference>
3841         <wsse:Reference URI="#RecipientEncryptedKey" />
3842       </wsse:SecurityTokenReference>
3843     </ds:KeyInfo>
3844   </xenc:EncryptedData>
3845 </S:Body>
3846 </S:Envelope>

```

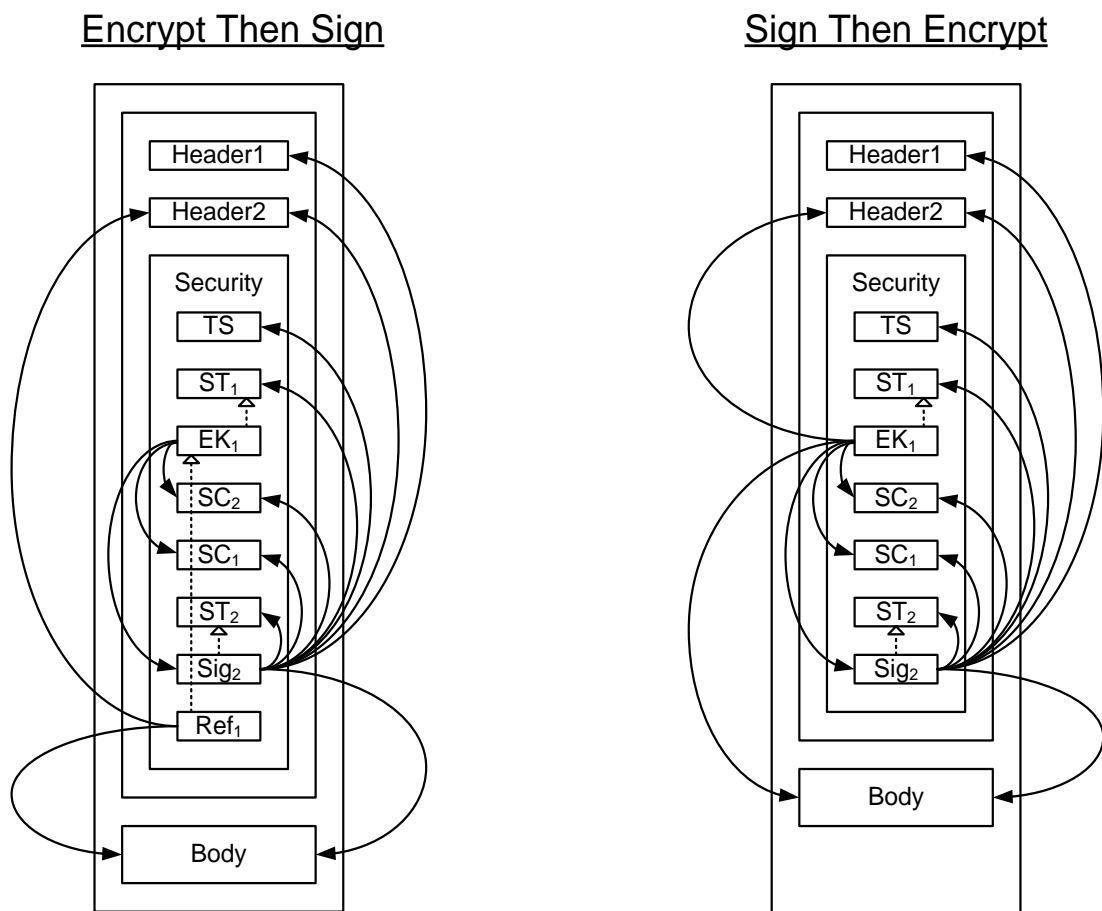
### 3847 C.3.3 Recipient to Initiator Messages

3848 Messages sent from recipient to initiator have the following layout:

- 3849 1. A `wsu:Timestamp` element if `[Timestamp]` is 'true'.
- 3850 2. If an `[Initiator Token]` is specified, and the associated `sp:IncludeToken` attribute is  
3851 `.../IncludeToken/Always`, then the `[Initiator Token]`.
- 3852 3. If an `[Initiator Token]` is specified and `[Protection Order]` is 'SignBeforeEncrypting' or  
3853 `[SignatureProtection]` is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3854 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3855 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If  
3856 `[Signature Protection]` is 'true' then the reference list MUST also contain a reference to the  
3857 message signature from 6 below, if any and references to the  
3858 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3859 4. If `[Signature Confirmation]` is 'true', then a `wss11:SignatureConfirmation` element for each  
3860 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3861 in the corresponding message from the initiator to the recipient, then a  
3862 `wss11:SignatureConfirmation` element with no `Value` attribute.
- 3863 5. If a `[Recipient Token]` is specified, and the associated `sp:IncludeToken` attribute is  
3864 `.../IncludeToken/Always`, then the `[Recipient Token]`.

- 3865 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],  
 3866 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements  
 3867 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token  
 3868 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3869 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
 3870 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
 3871 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
 3872 reference list includes a reference to all the message parts specified in EncryptedParts assertions  
 3873 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
 3874 element from 3 above.

3875  
 3876 The following diagram illustrates the security header layout for the recipient to initiator messages:



3877 18.  
 3878

3879 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3880 using the [Recipient Token] labeled ST<sub>2</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate  
 3881 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on  
 3882 the left from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained  
 3883 in the encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token  
 3884 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements  
 3885 labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures in the initial message illustrated previously is  
 3886 included. In general, the ordering of the items in the security header follows the most optimal layout for a  
 3887 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

3888 Recipient to initiator message *Example:*

```
3889 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3890   xmlns:wssell="..." xmlns:wsse="..."
3891   xmlns:xenc="..." xmlns:ds="...">
3892   <S:Header>
3893     <x:Header1 wsu:Id="Header1" >
3894       ...
3895     </x:Header1>
3896     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3897       <!-- Plaintext Header2
3898       <x:Header2 wsu:Id="Header2" >
3899         ...
3900       </x:Header2>
3901       -->
3902     ...
3903   </wssell:EncryptedHeader>
3904   ...
3905   <wsse:Security>
3906     <wsu:Timestamp wsu:Id="Timestamp">
3907       <wsu:Created>...</wsu:Created>
3908       <wsu:Expires>...</wsu:Expires>
3909     </wsu:Timestamp>
3910     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3911       ...
3912     </wsse:BinarySecurityToken>
3913     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3914       ...
3915       <xenc:ReferenceList>
3916         <xenc:DataReference URI="#enc_Signature" />
3917         <xenc:DataReference URI="#enc_SigConf1" />
3918         <xenc:DataReference URI="#enc_SigConf2" />
3919         ...
3920       </xenc:ReferenceList>
3921     </xenc:EncryptedKey>
3922     <xenc:EncryptedData ID="enc_SigConf2" >
3923       <!-- Plaintext SignatureConfirmation
3924       <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3925         ...
3926       </wssell:SignatureConfirmation>
3927       -->
3928     ...
3929   </xenc:EncryptedData>
3930   <xenc:EncryptedData ID="enc_SigConf1" >
3931     <!-- Plaintext SignatureConfirmation
3932     <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3933       ...
3934     </wssell:SignatureConfirmation>
3935     -->
3936   ...
3937 </xenc:EncryptedData>
3938 <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3939   ...
3940 </wsse:BinarySecurityToken>
3941
```

```

3942 <xenc:EncryptedData ID="enc_Signature">
3943   <!-- Plaintext Signature
3944   <ds:Signature Id="Signature">
3945     <ds:SignedInfo>
3946       <ds:References>
3947         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3948         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3949         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3950         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3951         <ds:Reference URI="#Header1" >...</ds:Reference>
3952         <ds:Reference URI="#Header2" >...</ds:Reference>
3953         <ds:Reference URI="#Body" >...</ds:Reference>
3954       </ds:References>
3955     </ds:SignedInfo>
3956     <ds:SignatureValue>...</ds:SignatureValue>
3957     <ds:KeyInfo>
3958       <wsse:SecurityTokenReference>
3959         <wsse:Reference URI="#RecipientToken" />
3960       </wsse:SecurityTokenReference>
3961     </ds:KeyInfo>
3962   </ds:Signature>
3963   -->
3964   ...
3965 </xenc:EncryptedData>
3966 <xenc:ReferenceList>
3967   <xenc:DataReference URI="#enc_Body" />
3968   <xenc:DataReference URI="#enc_Header2" />
3969   ...
3970 </xenc:ReferenceList>
3971 </wsse:Security>
3972 </S:Header>
3973 <S:Body wsu:Id="Body">
3974   <xenc:EncryptedData Id="enc_Body">
3975     ...
3976     <ds:KeyInfo>
3977       <wsse:SecurityTokenReference>
3978         <wsse:Reference URI="#InitiatorEncryptedKey" />
3979       </wsse:SecurityTokenReference>
3980     </ds:KeyInfo>
3981   </xenc:EncryptedData>
3982 </S:Body>
3983 </S:Envelope>

```

3984  
3985

---

## Appendix D. Signed and Encrypted Elements in the Security Header

3986  
3987  
3988  
3989  
3990

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.

3991

### D.1 Elements signed by the message signature

3992  
3993  
3994  
3995  
3996  
3997  
3998

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).

3999

### D.2 Elements signed by all endorsing signatures

4000  
4001

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).

4002

### D.3 Elements signed by a specific endorsing signature

4003  
4004

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

4005

### D.4 Elements that are encrypted

4006  
4007  
4008  
4009  
4010  
4011  
4012

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).



---

4013 **Appendix E. Acknowledgements**

4014 The following individuals have participated in the creation of this specification and are gratefully  
4015 acknowledged:

4016 **Original Authors of the initial contribution:**

4017 Giovanni Della-Libera, Microsoft  
4018 Martin Gudgin, Microsoft  
4019 Phillip Hallam-Baker, VeriSign  
4020 Maryann Hondo, IBM  
4021 Hans Granqvist, Verisign  
4022 Chris Kaler, Microsoft (editor)  
4023 Hiroshi Maruyama, IBM  
4024 Michael McIntosh, IBM  
4025 Anthony Nadalin, IBM (editor)  
4026 Nataraj Nagaratnam, IBM  
4027 Rob Philpott, RSA Security  
4028 Hemma Prafullchandra, VeriSign  
4029 John Shewchuk, Microsoft  
4030 Doug Walter, Microsoft  
4031 Riaz Zolfonoon, RSA Security

4032  
4033 **Original Acknowledgements of the initial contribution:**

4034 Vaithialingam B. Balayoghan, Microsoft  
4035 Francisco Curbera, IBM  
4036 Christopher Ferris, IBM  
4037 Cédric Fournet, Microsoft  
4038 Andy Gordon, Microsoft  
4039 Tomasz Janczuk, Microsoft  
4040 David Melgar, IBM  
4041 Mike Perks, IBM  
4042 Bruce Rich, IBM  
4043 Jeffrey Schlimmer, Microsoft  
4044 Chris Sharp, IBM  
4045 Kent Tamura, IBM  
4046 T.R. Vishwanath, Microsoft  
4047 Elliot Waingold, Microsoft

4048  
4049 **TC Members during the development of this specification:**

4050 Don Adams, Tibco Software Inc.  
4051 Jan Alexander, Microsoft Corporation  
4052 Steve Anderson, BMC Software  
4053 Donal Arundel, IONA Technologies  
4054 Howard Bae, Oracle Corporation  
4055 Abbie Barbir, Nortel Networks Limited  
4056 Charlton Barreto, Adobe Systems  
4057 Mighael Botha, Software AG, Inc.  
4058 Toufic Boubez, Layer 7 Technologies Inc.  
4059 Norman Brickman, Mitre Corporation  
4060 Melissa Brumfield, Booz Allen Hamilton

4061 Geoff Bullen, Microsoft Corporation  
4062 Lloyd Burch, Novell  
4063 Scott Cantor, Internet2  
4064 Greg Carpenter, Microsoft Corporation  
4065 Steve Carter, Novell  
4066 Symon Chang, Oracle Corporation Ching-Yun (C.Y.) Chao, IBM  
4067 Martin Chapman, Oracle Corporation  
4068 Kate Cherry, Lockheed Martin  
4069 Henry (Hyenvui) Chung, IBM  
4070 Luc Clement, Systinet Corp.  
4071 Paul Cotton, Microsoft Corporation  
4072 Glen Daniels, Sonic Software Corp.  
4073 Peter Davis, Neustar, Inc.  
4074 Duane DeCouteau, Veterans Health Administration  
4075 Martijn de Boer, SAP AG  
4076 Werner Dittmann, Siemens AG  
4077 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
4078 Fred Dushin, IONA Technologies  
4079 Petr Dvorak, Systinet Corp.  
4080 Colleen Evans, Microsoft Corporation  
4081 Ruchith Fernando, WSO2  
4082 Mark Fussell, Microsoft Corporation  
4083 Vijay Gajjala, Microsoft Corporation  
4084 Marc Goodner, Microsoft Corporation  
4085 Hans Granqvist, VeriSign  
4086 Martin Gudgin, Microsoft Corporation  
4087 Tony Gullotta, SOA Software Inc.  
4088 Jiandong Guo, Sun Microsystems  
4089 Phillip Hallam-Baker, VeriSign  
4090 Patrick Harding, Ping Identity Corporation  
4091 Heather Hinton, IBM  
4092 Frederick Hirsch, Nokia Corporation  
4093 Jeff Hodges, Neustar, Inc.  
4094 Will Hopkins, Oracle Corporation  
4095 Alex Hristov, Otecia Incorporated  
4096 John Hughes, PA Consulting  
4097 Diane Jordan, IBM  
4098 Venugopal K, Sun Microsystems  
4099 Chris Kaler, Microsoft Corporation  
4100 Dana Kaufman, Forum Systems, Inc.  
4101 Paul Knight, Nortel Networks Limited  
4102 Ramanathan Krishnamurthy, IONA Technologies  
4103 Christopher Kurt, Microsoft Corporation  
4104 Kelvin Lawrence, IBM  
4105 Hubert Le Van Gong, Sun Microsystems  
4106 Jong Lee, Oracle Corporation  
4107 Rich Levinson, Oracle Corporation  
4108 Tommy Lindberg, Dajeil Ltd.  
4109 Mark Little, JBoss Inc.  
4110 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.  
4111 Eve Maler, Sun Microsystems  
4112 Ashok Malhotra, Oracle Corporation  
4113 Anand Mani, CrimsonLogic Pte Ltd  
4114 Jonathan Marsh, Microsoft Corporation  
4115 Robin Martherus, Oracle Corporation  
4116 Miko Matsumura, Infravio, Inc.  
4117 Gary McAfee, IBM

4118 Michael McIntosh, IBM  
4119 John Merrells, Sxip Networks SRL  
4120 Jeff Mischkinsky, Oracle Corporation  
4121 Prateek Mishra, Oracle Corporation  
4122 Bob Morgan, Internet2  
4123 Vamsi Motukuru, Oracle Corporation  
4124 Raajmohan Na, EDS  
4125 Anthony Nadalin, IBM  
4126 Andrew Nash, Reactivity, Inc.  
4127 Eric Newcomer, IONA Technologies  
4128 Duane Nickull, Adobe Systems  
4129 Toshihiro Nishimura, Fujitsu Limited  
4130 Rob Philpott, RSA Security  
4131 Denis Pilipchuk, Oracle Corporation.  
4132 Darren Platt, Ping Identity Corporation  
4133 Martin Raepple, SAP AG  
4134 Nick Ragouzis, Enosis Group LLC  
4135 Prakash Reddy, CA  
4136 Alain Regnier, Ricoh Company, Ltd.  
4137 Irving Reid, Hewlett-Packard  
4138 Bruce Rich, IBM  
4139 Tom Rutt, Fujitsu Limited  
4140 Maneesh Sahu, Actional Corporation  
4141 Frank Siebenlist, Argonne National Laboratory  
4142 Joe Smith, Apani Networks  
4143 Davanum Srinivas, WSO2  
4144 David Staggs, Veterans Health Administration  
4145 Yakov Sverdlov, CA  
4146 Gene Thurston, AmberPoint  
4147 Victor Valle, IBM  
4148 Asir Vedamuthu, Microsoft Corporation  
4149 Greg Whitehead, Hewlett-Packard  
4150 Ron Williams, IBM  
4151 Corinna Witt, Oracle Corporation  
4152 Kyle Young, Microsoft Corporation