

## WS-SecurityPolicy 1.2

### OASIS Standard incorporating Approved Errata 01

25 April 2012

#### Specification URIs

##### This version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/ws-securitypolicy-1.2-errata01-os-complete.doc> (Authoritative)  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/ws-securitypolicy-1.2-errata01-os-complete.html>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/ws-securitypolicy-1.2-errata01-os-complete.pdf>

##### Previous version:

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

##### Latest version:

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/ws-securitypolicy-1.2-errata01-complete.doc> (Authoritative)  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/ws-securitypolicy-1.2-errata01-complete.html>  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/ws-securitypolicy-1.2-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 1.2 Errata 01*. 25 April 2012. OASIS Approved Errata. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/ws-securitypolicy-1.2-errata01-os.html>.
- XML schema: <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/schemas/ws-securitypolicy-1.2.xsd>

**Related work:**

This specification is related to:

- *WS-SecurityPolicy 1.2*. 1 July 2007. OASIS Standard.  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html>.

**Declared XML namespace:**

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

**Abstract:**

This document indicates the policy assertions for use with [WS-Policy] which apply to WSS: SOAP Message Security [WSS10, WSS11], [WS-Trust] and [WS-SecureConversation]. 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-1.2]**

*WS-SecurityPolicy 1.2*. 25 April 2012. OASIS Standard incorporating Approved Errata.  
<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/errata01/os/ws-securitypolicy-1.2-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.....	22
4.3	Required Elements Assertion.....	23
4.3.1	RequiredElements Assertion.....	23
4.3.2	RequiredParts Assertion.....	24
5	Token Assertions.....	25
5.1	Token Inclusion.....	25
5.1.1	Token Inclusion Values.....	25
5.1.2	Token Inclusion and Token References.....	26
5.2	Token Issuer and Required Claims.....	26
5.2.1	Token Issuer.....	26
5.2.2	Token Issuer Name.....	26
5.2.3	Required Claims.....	26
5.2.4	Processing Rules and Token Matching.....	27
5.3	Token Properties.....	27
5.3.1	[Derived Keys] Property.....	27
5.3.2	[Explicit Derived Keys] Property.....	27
5.3.3	[Implied Derived Keys] Property.....	27
5.4	Token Assertion Types.....	27
5.4.1	UsernameToken Assertion.....	27
5.4.2	IssuedToken Assertion.....	29
5.4.3	X509Token Assertion.....	31

5.4.4	KerberosToken Assertion .....	33
5.4.5	SpnegoContextToken Assertion .....	34
5.4.6	SecurityContextToken Assertion .....	36
5.4.7	SecureConversationToken Assertion .....	37
5.4.8	SamlToken Assertion .....	40
5.4.9	RelToken Assertion .....	42
5.4.10	HttpsToken Assertion .....	43
5.4.11	KeyValueToken Assertion .....	44
6	Security Binding Properties .....	47
6.1	[Algorithm Suite] Property .....	47
6.2	[Timestamp] Property .....	49
6.3	[Protection Order] Property .....	49
6.4	[Signature Protection] Property .....	49
6.5	[Token Protection] Property .....	49
6.6	[Entire Header and Body Signatures] Property .....	50
6.7	[Security Header Layout] Property .....	50
6.7.1	Strict Layout Rules for WSS 1.0 .....	50
7	Security Binding Assertions .....	52
7.1	AlgorithmSuite Assertion .....	52
7.2	Layout Assertion .....	54
7.3	TransportBinding Assertion .....	55
7.4	SymmetricBinding Assertion .....	56
7.5	AsymmetricBinding Assertion .....	58
8	Supporting Tokens .....	61
8.1	SupportingTokens Assertion .....	62
8.2	SignedSupportingTokens Assertion .....	64
8.3	EndorsingSupportingTokens Assertion .....	65
8.4	SignedEndorsingSupportingTokens Assertion .....	67
8.5	SignedEncryptedSupportingTokens Assertion .....	69
8.6	EncryptedSupportingTokens Assertion .....	69
8.7	EndorsingEncryptedSupportingTokens Assertion .....	70
8.8	SignedEndorsingEncryptedSupportingTokens Assertion .....	70
8.9	Interaction between [Token Protection] property and supporting token assertions .....	70
8.10	Example .....	70
9	WSS: SOAP Message Security Options .....	72
9.1	Wss10 Assertion .....	73
9.2	Wss11 Assertion .....	74
10	WS-Trust Options .....	76
10.1	Trust13 Assertion .....	77
11	Guidance on creating new assertions and assertion extensibility .....	79
11.1	General Design Points .....	79
11.2	Detailed Design Guidance .....	79
12	Security Considerations .....	81
Appendix A.	Assertions and WS-PolicyAttachment .....	82
A.1	Endpoint Policy Subject Assertions .....	82

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

---

# 1 Introduction

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

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

## 1.1 Example

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

Table 1: Example security policy.

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

```

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

```

49

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

## 62 1.2 Namespaces

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

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

65

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

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

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



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

## 69 1.3 Schema Files

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

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

## 73 1.4 Terminology

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

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

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

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

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

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

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

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

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

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

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

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

### 90 1.4.1 Notational Conventions

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

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

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

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

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

112

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

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

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

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

127

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

## 133 1.5 Normative References

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

153

154 [RFC2068] IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January  
155 1997  
156 <http://www.ietf.org/rfc/rfc2068.txt>  
157

158 [RFC2246] IETF Standard, "The TLS Protocol", January 1999.  
159 <http://www.ietf.org/rfc/rfc2246.txt>  
160

161 [SwA] W3C Note, "SOAP Messages with Attachments", 11 December 2000  
162 <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211>  
163

164 [WS-Addressing] W3C Recommendation, "Web Services Addressing (WS-Addressing)",  
165 9 May 2006.  
166 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>  
167

168 [WS-Policy] W3C Member Submission "Web Services Policy 1.2 - Framework", 25  
169 April 2006.  
170 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>  
171 W3C Candidate Recommendation "Web Services Policy 1.5 –  
172 Framework", 28 February 2007  
173 <http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/>  
174

175 [WS-PolicyAttachment] W3C Member Submission "Web Services Policy 1.2 - Attachment", 25  
176 April 2006.  
177 <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/>  
178  
179 W3C Candidate Recommendation "Web Services Policy 1.5 –  
180 Attachment", 28 February 2007  
181 <http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/>  
182

183 [WS-Trust] OASIS Committee Draft, "WS-Trust 1.3", September 2006  
184 <http://docs.oasis-open.org/ws-sx/ws-trust/200512>  
185

186 [WS-SecureConversation] OASIS Committee Draft, "WS-SecureConversation 1.3", September  
187 2006  
188 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512>  
189

190 [WSS10] OASIS Standard, "OASIS Web Services Security: SOAP Message  
191 Security 1.0 (WS-Security 2004)", March 2004.  
192 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-  
193 message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)  
194

195 [WSS11] OASIS Standard, "OASIS Web Services Security: SOAP Message  
196 Security 1.1 (WS-Security 2004)", February 2006.  
197 [http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-  
198 spec-os-SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)

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

245 <http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf>  
246  
247  
248 [XML-Encrypt] W3C Recommendation, "XML Encryption Syntax and Processing", 10  
249 December 2002.  
250 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>  
251  
252 [XML-Signature] W3C Recommendation, "XML-Signature Syntax and Processing", 12  
253 February 2002.  
254 <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>  
255 W3C Recommendation, D. Eastlake et al. XML Signature Syntax and  
256 Processing (Second Edition). 10 June 2008.  
257 <http://www.w3.org/TR/2008/REC-xmlsig-core-20080610/>  
258  
259 [XPath] W3C Recommendation "XML Path Language (XPath) Version 1.0", 16  
260 November 1999.  
261 <http://www.w3.org/TR/1999/REC-xpath-19991116>  
262  
263 [XML-Schema1] W3C Recommendation, "XML Schema Part 1: Structures Second  
264 Edition", 28 October 2004.  
265 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>  
266  
267 [XML-Schema2] W3C Recommendation, "XML Schema Part 2: Datatypes Second  
268 Edition", 28 October 2004.  
269 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>  
270

## 271 **1.6 Non-Normative References**

272 None.  
273

---

## 274 **2 Security Policy Model**

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

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

284  
285 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for  
286 selecting policy alternatives and the attachment mechanism for associating policy assertions with web  
287 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters  
288 or attributes. This enables first-level, QName based assertion matching without security domain-specific  
289 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed  
290 set of policy alternatives that are shared by the two parties attempting to establish a secure  
291 communication path. Parameters defined by this specification represent additional information for  
292 engaging behaviors that do not need to participate in matching. When multiple security policy assertions  
293 of the same type with parameters present occur in the same policy alternative the parameters should be  
294 treated as a union. Note that a service may choose to accept messages that do not match its policy.

295  
296 In general, assertions defined in this specification allow additional attributes, based on schemas, to be  
297 added on to the assertion element as an extensibility mechanism but the WS-Policy framework will not  
298 match based on these attributes. Attributes specified on the assertion element that are not defined in this  
299 specification or in WS-Policy are to be treated as informational properties.

### 300 **2.1 Security Assertion Model**

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

308  
309 To indicate the scope of protection, assertions identify message parts that are to be protected in a  
310 specific way, such as integrity or confidentiality protection, and are referred to as protection assertions.

311  
312 The general aspects of security includes the relationships between or characteristics of the environment  
313 in which security is being applied, such as the tokens being used, which are for integrity or confidentiality  
314 protection and which are supporting, the applicable algorithms to use, etc.

315

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

321  
322 The intent of representing characteristics as assertions is so that QName matching will be sufficient to  
323 find common alternatives and so that many aspects of security can be factored out and re-used. For  
324 example, it may be common that the mechanism is constant for an endpoint, but that the parts protected  
325 vary by message action.

## 326 **2.2 Nested Policy Assertions**

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

## 333 **2.3 Security Binding Abstraction**

334 As previously indicated, individual assertions are designed to be used in multiple combinations. The  
335 binding represents common usage patterns for security mechanisms. These Security Binding assertions  
336 are used to determine how the security is performed and what to expect in the `wsse:Security` header.  
337 Bindings are described textually and enforced programmatically. This specification defines several  
338 bindings but others can be defined and agreed to for interoperability if participating parties support it.

339  
340 A binding defines the following security characteristics:

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

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

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

- 360 • TransportBinding (Section 7.3)
- 361 • SymmetricBinding (Section 7.4)
- 362 • AsymmetricBinding (Section 7.5)



---

## 363 3 Policy Considerations

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

### 366 3.1 Nested Policy

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

369

### 370 3.2 Policy Subjects

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

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

#### 375 [Message Policy Subject]

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

378

379 wsdl:message

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

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

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

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

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

#### 388 [Operation Policy Subject]

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

390 wsdl:portType/wsdl:operation

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

393 wsdl:binding/wsdl:operation

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

396

397

#### 398 [Endpoint Policy Subject]

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

401 wsdl:portType

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

404 wsdl:binding

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

407 wsdl:port

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

---

## 410 4 Protection Assertions

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

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

### 418 4.1 Integrity Assertions

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

#### 422 4.1.1 SignedParts Assertion

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

428

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

#### 433 Syntax

```
434 <sp:SignedParts xmlns:sp="..." ... >  
435   <sp:Body />?  
436   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />?  
437   <sp:Attachments> ... </sp:Attachments>?  
438   ...  
439 </sp:SignedParts>
```

440

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

442 /sp:SignedParts

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

446 /sp:SignedParts/sp:Body

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

449 /sp:SignedParts/sp:Header

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

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

460 /sp:SignedParts/sp:Header/@Name

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

464 /sp:SignedParts/sp:Header/@Namespace

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

467 /sp:SignedParts/sp:Attachments

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

## 472 4.1.2 SignedElements Assertion

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

478

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

### 482 Syntax

```
483 <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >  
484   <sp:XPath>xs:string</sp:XPath>+  
485   ...  
486 </sp:SignedElements>
```

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

488 /sp:SignedElements

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

490 /sp:SignedElements/@XPathVersion

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

493 /sp:SignedElements/sp:XPath

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

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

## 498 4.2 Confidentiality Assertions

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

### 502 4.2.1 EncryptedParts Assertion

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

508

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

#### 513 Syntax

```
514 <sp:EncryptedParts xmlns:sp="..." ... >  
515   <sp:Body/>?  
516   <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*  
517   <sp:Attachments />?  
518   ...  
519 </sp:EncryptedParts>
```

520

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

522 /sp:EncryptedParts

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

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

526 /sp:EncryptedParts/sp:Body

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

531 /sp:EncryptedParts/sp:Header

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

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

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

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

## 554 4.2.2 EncryptedElements Assertion

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

560

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

### 564 Syntax

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

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

570 /sp:EncryptedElements

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

573 /sp:EncryptedElements/@XPathVersion

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

576 /sp:EncryptedElements/sp:XPath

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

## 581 4.2.3 ContentEncryptedElements Assertion

582 The ContentEncryptedElements assertion is used to specify arbitrary elements in the message that  
 583 require confidentiality protection of their content. This assertion can be satisfied using WSS: SOAP  
 584 Message Security mechanisms or by mechanisms out of scope of SOAP message security, for example

585 by sending the message over a secure transport protocol like HTTPS. The binding specific token  
586 properties detail the exact mechanism by which the protection is provided.

587

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

#### 591 **Syntax**

```
592 <sp:ContentEncryptedElements XPathVersion="xs:anyURI"? ...>  
593   <sp:XPath>xs:string</sp:XPath>+  
594   ...  
595 </sp:ContentEncryptedElements>
```

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

597 /sp:ContentEncryptedElements

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

600 /sp:ContentEncryptedElements/@XPathVersion

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

603 /sp:ContentEncryptedElements/sp:XPath

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

### 608 **4.3 Required Elements Assertion**

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

611

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

#### 615 **4.3.1 RequiredElements Assertion**

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

618

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

#### 622 **Syntax**

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

627

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

629 /sp:RequiredElements

630 This assertion specifies the headers elements that MUST appear in a message.  
631 /sp:RequiredElements/@XPathVersion  
632 This OPTIONAL attribute contains a URI which indicates the version of XPath to use. If no  
633 attribute is provided, then XPath 1.0 is assumed.  
634 /sp:RequiredElements/sp:XPath  
635 This element contains a string specifying an XPath expression that identifies the header elements  
636 that a message MUST contain. The XPath expression is evaluated against the  
637 S:Envelope/S:Header element node of the message. Multiple instances of this element MAY  
638 appear within this assertion and SHOULD be treated as a combined XPath expression.

### 639 4.3.2 RequiredParts Assertion

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

643  
644 There MAY be multiple RequiredParts assertions present. Multiple RequiredParts assertions present  
645 within a policy alternative are equivalent to a single RequiredParts assertion containing the union of all  
646 specified Header elements.

#### 647 Syntax

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

651  
652 The following describes the attributes and elements listed in the schema outlined above:  
653 /sp:RequiredParts/sp:Header  
654 This assertion specifies the headers elements that MUST be present in the message.  
655 /sp:RequiredParts/sp:Header/@Name  
656 This REQUIRED attribute indicates the local name of the SOAPHeader that needs to be present  
657 in the message.  
658 /sp:RequiredParts/sp:Header/@Namespace  
659 This REQUIRED attribute indicates the namespace of the SOAP header that needs to be present  
660 in the message.



## 661 5 Token Assertions

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

### 666 5.1 Token Inclusion

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

#### 672 5.1.1 Token Inclusion Values

673 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.

674  
675 Note: In examples, the namespace URI is replaced with "...". For example,  
676 `.../IncludeToken/Never` is actually `http://docs.oasis-open.org/ws-sx/ws-`  
677 `securitypolicy/200702/IncludeToken/Never`. Other token inclusion URI values MAY be defined but are out-  
678 of-scope of this specification.

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

## 681 **5.1.2 Token Inclusion and Token References**

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

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

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

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

## 698 **5.2 Token Issuer and Required Claims**

### 699 **5.2.1 Token Issuer**

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

### 704 **5.2.2 Token Issuer Name**

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

709  
710 It is out of scope of this specification how the relationship between the issuer's logical name and the  
711 physical manifestation of the issuer in the security token is defined.

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

### 714 **5.2.3 Required Claims**

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

718  
719 This element indicates the REQUIRED claims that the security token MUST contain in order to satisfy the  
720 requirements of the token assertion.

721  
722 Individual token assertions MAY further limit what claims MAY be specified for that specific token  
723 assertion.

## 724 5.2.4 Processing Rules and Token Matching

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

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

- 732 1. Two tokens, T1 and T2. T1 is issued by issuer A and contains claims C1 and C2 and T2 is issued  
733 by issuer B and contains claims C3 and C4.
- 734 2. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claim C1, T2 is also issued  
735 by issuer A and contains claim C2 and T3 is issued by issuer B and contains claims C3 and C4.
- 736 3. Three tokens, T1, T2 and T3. T1 is issued by issuer A and contains claims C1 and C2, T2 is  
737 issued by issuer B and contains claim C3 and T3 is also issued by issuer B and contains claim  
738 C4.
- 739 4. Four tokens, T1, T2, T3 and T4. T1 is issued by issuer A and contains claim C1, T2 is also issued  
740 by issuer A and contains claim C2, T3 is issued by issuer B and contains claim C3 and T4 is also  
741 issued by issuer B and contains claim C4.

## 743 5.3 Token Properties

### 744 5.3.1 [Derived Keys] Property

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

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

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

### 753 5.3.2 [Explicit Derived Keys] Property

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

### 757 5.3.3 [Implied Derived Keys] Property

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

## 761 5.4 Token Assertion Types

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

### 763 5.4.1 UsernameToken Assertion

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

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

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

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

773

774 **Syntax**

```

775 <sp:UsernameToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
776 (
777   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
778   <sp:IssuerName>xs:anyURI</sp:IssuerName>
779 ) ?
780 <wst:Claims Dialect="..."> ... </wst:Claims> ?
781 <wsp:Policy xmlns:wsp="...">
782 (
783   <sp:NoPassword ... /> |
784   <sp:HashPassword ... />
785 ) ?
786 (
787   <sp:RequireDerivedKeys /> |
788   <sp:RequireImpliedDerivedKeys ... /> |
789   <sp:RequireExplicitDerivedKeys ... />
790 ) ?
791 (
792   <sp:WssUsernameToken10 ... /> |
793   <sp:WssUsernameToken11 ... />
794 ) ?
795 ...
796 </wsp:Policy>
797 ...
798 </sp:UsernameToken>

```

799

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

801 /sp:UsernameToken

802 This identifies a UsernameToken assertion.

803 /sp:UsernameToken/@sp:IncludeToken

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

805 /sp:UsernameToken/sp:Issuer

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

808 /sp:UsernameToken/sp:IssuerName

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

811 /sp:UsernameToken/wst:Claims

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

814 /sp:UsernameToken/wsp:Policy

815 This REQUIRED element identifies additional requirements for use of the sp:UsernameToken  
816 assertion.

817 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

820 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

825 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

828 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

832 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

836 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

839 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

## 842 5.4.2 IssuedToken Assertion

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

### 847 Syntax

```
848 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
849 (   
850 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
851 <sp:IssuerName>xs:anyURI</sp:IssuerName>   
852 ) ?
```

```

853 <wst:Claims Dialect="..."> ... </wst:Claims> ?
854 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
855   ...
856 </sp:RequestSecurityTokenTemplate>
857 <wsp:Policy xmlns:wsp="...">
858   (
859     <sp:RequireDerivedKeys ... /> |
860     <sp:RequireImpliedDerivedKeys ... /> |
861     <sp:RequireExplicitDerivedKeys ... />
862   ) ?
863   <sp:RequireExternalReference ... /> ?
864   <sp:RequireInternalReference ... /> ?
865   ...
866 </wsp:Policy>
867   ...
868 </sp:IssuedToken>

```

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

870 /sp:IssuedToken

871 This identifies an IssuedToken assertion.

872 /sp:IssuedToken/@sp:IncludeToken

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

874 /sp:IssuedToken/sp:Issuer

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

877 /sp:IssuedToken/sp:IssuerName

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

880 /sp:IssuedToken/wst:Claims

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

883 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

888 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

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

891 /sp:IssuedToken/wsp:Policy

892 This REQUIRED element identifies additional requirements for use of the sp:IssuedToken  
893 assertion.

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

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

897 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

901 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
902 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
903 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
904 'false'.

905 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference  
906 This OPTIONAL element is a policy assertion that indicates whether an internal reference is  
907 REQUIRED when referencing this token.  
908 Note: This reference will be supplied by the issuer of the token.

909 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference  
910 This OPTIONAL element is a policy assertion that indicates whether an external reference is  
911 REQUIRED when referencing this token.  
912 Note: This reference will be supplied by the issuer of the token.

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

### 918 5.4.3 X509Token Assertion

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

#### 920 Syntax

```
921 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
922   (  
923     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
924     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
925   ) ?  
926   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
927   <wsp:Policy xmlns:wsp="...">  
928     (  
929       <sp:RequireDerivedKeys ... /> |  
930       <sp:RequireExplicitDerivedKeys ... /> |  
931       <sp:RequireImpliedDerivedKeys ... />  
932     ) ?  
933     <sp:RequireKeyIdentifierReference ... /> ?  
934     <sp:RequireIssuerSerialReference ... /> ?  
935     <sp:RequireEmbeddedTokenReference ... /> ?  
936     <sp:RequireThumbprintReference ... /> ?  
937     (  
938       <sp:WssX509V3Token10 ... /> |  
939       <sp:WssX509Pkcs7Token10 ... /> |  
940       <sp:WssX509PkiPathV1Token10 ... /> |  
941       <sp:WssX509V1Token11 ... /> |  
942       <sp:WssX509V3Token11 ... /> |  
943       <sp:WssX509Pkcs7Token11 ... /> |  
944       <sp:WssX509PkiPathV1Token11 ... />  
945     ) ?  
946     ...  
947   </wsp:Policy>  
948   ...  
949 </sp:X509Token>
```

950  
951 The following describes the attributes and elements listed in the schema outlined above:

952 /sp:X509Token

953 This identifies an X509Token assertion.

954 /sp:X509Token/@sp:IncludeToken

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

956 /sp:X509Token/sp:Issuer

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

959 /sp:X509Token/sp:IssuerName

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

962 /sp:X509Token/wst:Claims

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

965 /sp:X509Token/wsp:Policy

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

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

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

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

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

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

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

979 /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

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

982 /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

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

985 /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

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

988 /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

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

991 /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

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

994 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10



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

997 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

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

1000 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1001 This OPTIONAL element is a policy assertion that indicates that an X509 Version 1 token should  
1002 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1003 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1004 This OPTIONAL element is a policy assertion that indicates that an X509 Version 3 token should  
1005 be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1006 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1007 This OPTIONAL element is a policy assertion that indicates that an X509 PKCS7 token should be  
1008 used as defined in [\[WSS:X509TokenProfile1.1\]](#).

1009 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1010 This OPTIONAL element is a policy assertion that indicates that an X509 PKI Path Version 1  
1011 token should be used as defined in [\[WSS:X509TokenProfile1.1\]](#).

## 1012 5.4.4 KerberosToken Assertion

1013 This element represents a requirement for a Kerberos token [\[WSS:KerberosToken1.1\]](#).

### 1014 Syntax

```
1015 <sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1016 (   
1017   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
1018   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1019 ) ?  
1020 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1021 <wsp:Policy xmlns:wsp="...">  
1022 (   
1023   <sp:RequireDerivedKeys ... /> |   
1024   <sp:RequireImpliedDerivedKeys ... /> |   
1025   <sp:RequireExplicitDerivedKeys ... />  
1026 ) ?  
1027 <sp:RequireKeyIdentifierReference ... /> ?  
1028 (   
1029   <sp:WssKerberosV5ApReqToken11 ... /> |   
1030   <sp:WssGssKerberosV5ApReqToken11 ... />  
1031 ) ?  
1032 ...  
1033 </wsp:Policy>  
1034 ...  
1035 </sp:KerberosToken>
```

1037

1038 The following describes the attributes and elements listed in the schema outlined above:

1039 /sp:KerberosToken

1040 This identifies a KerberosV5ApReqToken assertion.

1041 /sp:KerberosToken/@sp:IncludeToken

1042 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1043 /sp:KerberosToken/sp:Issuer  
 1044 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
 1045 of the sp:KerberosToken.

1046 /sp:KerberosToken/sp:IssuerName  
 1047 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:KerberosToken  
 1048 issuer.

1049 /sp:KerberosToken/wst:Claims  
 1050 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
 1051 order to satisfy the token assertion requirements.

1052 /sp:KerberosToken/wsp:Policy  
 1053 This REQUIRED element identifies additional requirements for use of the sp:KerberosToken  
 1054 assertion.

1055 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys  
 1056 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1057 and [Implied Derived Keys] properties for this token to 'true'.

1058 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
 1059 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1060 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1061 'false'.

1062 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1063 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1064 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1065 'false'.

1066 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference  
 1067 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
 1068 REQUIRED when referencing this token.

1069 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11  
 1070 This OPTIONAL element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ  
 1071 token should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

1072 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11  
 1073 This OPTIONAL element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-  
 1074 REQ token should be used as defined in [[WSS:KerberosTokenProfile1.1](#)].

## 1075 5.4.5 SpnegoContextToken Assertion

1076 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg  
 1077 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

### 1078 Syntax

```
1079 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1080 (
1081   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1082   <sp:IssuerName>xs:anyURI</sp:IssuerName>
```

```

1083 ) ?
1084 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1085 <wsp:Policy xmlns:wsp="...">
1086   (
1087     <sp:RequireDerivedKeys ... /> |
1088     <sp:RequireImpliedDerivedKeys ... /> |
1089     <sp:RequireExplicitDerivedKeys ... />
1090   ) ?
1091   <sp:MustNotSendCancel ... /> ?
1092   <sp:MustNotSendAmend ... /> ?
1093   <sp:MustNotSendRenew ... /> ?
1094   ...
1095 </wsp:Policy>
1096 ...
1097 </sp:SpnegoContextToken>

```

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

1100 /sp:SpnegoContextToken

1101 This identifies a SpnegoContextToken assertion.

1102 /sp:SpnegoContextToken/@sp:IncludeToken

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

1104 /sp:SpnegoContextToken/sp:Issuer

1105 This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
1106 for the Spnego Context Token.

1107 /sp:SpnegoContextToken/sp:IssuerName

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

1110 /sp:SpnegoContextToken/wst:Claims

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

1113 /sp:SpnegoContextToken/wsp:Policy

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

1116 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

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

1119 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1123 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

1127 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1128 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
1129 token does not support SCT/Cancel RST messages. If this assertion is missing it means that  
1130 SCT/Cancel RST messages are supported by the STS.

1131 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend  
1132 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
1133 token does not support SCT/Amend RST messages. If this assertion is missing it means that  
1134 SCT/Amend RST messages are supported by the STS.

1135 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew  
1136 This OPTIONAL element is a policy assertion that indicates that the STS issuing the SP/Nego  
1137 token does not support SCT/Renew RST messages. If this assertion is missing it means that  
1138 SCT/Renew RST messages are supported by the STS.

## 1139 5.4.6 SecurityContextToken Assertion

1140 This element represents a requirement for a SecurityContextToken token.

### 1141 Syntax

```
1142 <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1143 (  
1144   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
1145   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1146 ) ?  
1147 <wst:Claims Dialect="..."> ... </wst:Claims> ?  
1148 <wsp:Policy xmlns:wsp="...">  
1149   (  
1150     <sp:RequireDerivedKeys ... /> |  
1151     <sp:RequireImpliedDerivedKeys ... /> |  
1152     <sp:RequireExplicitDerivedKeys ... />  
1153   ) ?  
1154   <sp:RequireExternalUriReference ... /> ?  
1155   <sp:SC13SecurityContextToken... /> ?  
1156   ...  
1157 </wsp:Policy>  
1158   ...  
1159 </sp:SecurityContextToken>
```

1160  
1161 The following describes the attributes and elements listed in the schema outlined above:

1162 /sp:SecurityContextToken

1163 This identifies a SecurityContextToken assertion.

1164 /sp:SecurityContextToken/@sp:IncludeToken

1165 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1166 /sp:SecurityContextToken/sp:Issuer

1167 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1168 of the sp:SecurityContextToken.

1169 /sp:SecurityContextToken/sp:IssuerName

1170 This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1171 sp:SecurityContextToken issuer.

1172 /sp:SecurityContextToken/wst:Claims

1173 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1174 order to satisfy the token assertion requirements.

1175 /sp:SecurityContextToken/wsp:Policy

1176 This REQUIRED element identifies additional requirements for use of the  
1177 sp:SecurityContextToken assertion.

1178 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys  
 1179 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
 1180 and [Implied Derived Keys] properties for this token to 'true'.

1181 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
 1182 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1183 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1184 'false'.

1185 /sp:SecurityContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1186 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1187 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1188 'false'.

1189 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference  
 1190 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
 1191 REQUIRED when referencing this token.

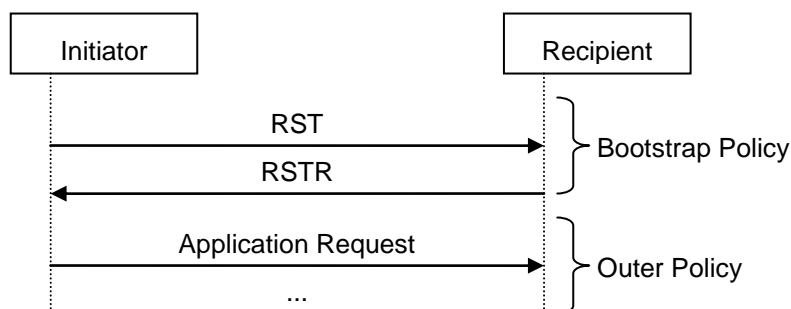
1192 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken  
 1193 This OPTIONAL element is a policy assertion that indicates that a Security Context Token  
 1194 SHOULD be used as defined in [[WS-SecureConversation](#)].

1195  
 1196 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that  
 1197 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If  
 1198 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the  
 1199 sp:SecureConversationToken or the sp:IssuedToken assertion SHOULD be used instead.

## 5.4.7 SecureConversationToken Assertion

1201 This element represents a requirement for a Security Context Token retrieved from the indicated issuer  
 1202 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the  
 1203 service endpoint address.

1205 Note: This assertion describes the token accepted by the target service. Because this token is issued by  
 1206 the target service and MAY NOT have a separate port (with separate policy), this assertion SHOULD  
 1207 contain a bootstrap policy indicating the security binding and policy that is used when requesting this  
 1208 token from the target service. That is, the bootstrap policy is used to obtain the token and then the  
 1209 current (outer) policy is used when making requests with the token. This is illustrated in the diagram  
 1210 below.



### Syntax

```

1213 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1214 (
  
```

```

1215 <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1216 <sp:IssuerName>xs:anyURI</sp:IssuerName>
1217 ) ?
1218 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1219 <wsp:Policy xmlns:wsp="...">
1220 (
1221   <sp:RequireDerivedKeys ... /> |
1222   <sp:RequireImpliedDerivedKeys ... /> |
1223   <sp:RequireExplicitDerivedKeys ... />
1224 ) ?
1225 <sp:RequireExternalUriReference ... /> ?
1226 <sp:SC13SecurityContextToken ... /> ?
1227 <sp:MustNotSendCancel ... /> ?
1228 <sp:MustNotSendAmend ... /> ?
1229 <sp:MustNotSendRenew ... /> ?
1230 <sp:BootstrapPolicy ... >
1231   <wsp:Policy> ... </wsp:Policy>
1232 </sp:BootstrapPolicy> ?
1233 </wsp:Policy>
1234 ...
1235 </sp:SecureConversationToken>

```

1236

1237 The following describes the attributes and elements listed in the schema outlined above:

1238 /sp:SecureConversationToken

1239       This identifies a SecureConversationToken assertion.

1240 /sp:SecureConversationToken/@sp:IncludeToken

1241       This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1242 /sp:SecureConversationToken/sp:Issuer

1243       This OPTIONAL element, of type wsa:EndpointReferenceType, contains a reference to the issuer  
1244       for the Security Context Token.

1245 /sp:SecureConversationToken/sp:IssuerName

1246       This OPTIONAL element, of type xs:anyURI, contains the logical name of the  
1247       sp:SecureConversationToken issuer.

1248 /sp:SpnegoContextToken/wst:Claims

1249       This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1250       order to satisfy the token assertion requirements.

1251 /sp:SecureConversationToken/wsp:Policy

1252       This REQUIRED element identifies additional requirements for use of the  
1253       sp:SecureConversationToken assertion.

1254 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1255       This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1256       and [Implied Derived Keys] properties for this token to 'true'.

1257 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1258       This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1259       Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1260       'false'.

1261 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1262 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1263 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1264 'false'.

1265 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1266 This OPTIONAL element is a policy assertion that indicates that an external URI reference is  
 1267 REQUIRED when referencing this token.

1268 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken

1269 This OPTIONAL element is a policy assertion that indicates that a Security Context Token should  
 1270 be used as obtained using the protocol defined in [WS-SecureConversation].

1271 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

1272 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1273 conversation token does not support SCT/Cancel RST messages. If this assertion is missing it  
 1274 means that SCT/Cancel RST messages are supported by the STS.

1275 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

1276 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1277 conversation token does not support SCT/Amend RST messages. If this assertion is missing it  
 1278 means that SCT/Amend RST messages are supported by the STS.

1279 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1280 This OPTIONAL element is a policy assertion that indicates that the STS issuing the secure  
 1281 conversation token does not support SCT/Renew RST messages. If this assertion is missing it  
 1282 means that SCT/Renew RST messages are supported by the STS.

1283 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1284 This OPTIONAL element is a policy assertion that contains the policy indicating the requirements  
 1285 for obtaining the Security Context Token.

1286 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1287 This element contains the security binding requirements for obtaining the Security Context Token.  
 1288 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with  
 1289 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that  
 1290 are to be protected.

1291 **Example**

```

1292 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1293   <sp:SymmetricBinding>
1294     <wsp:Policy>
1295       <sp:ProtectionToken>
1296         <wsp:Policy>
1297           <sp:SecureConversationToken>
1298             <sp:Issuer>
1299               <wsa:Address>http://example.org/sts</wsa:Address>
1300             </sp:Issuer>
1301           </wsp:Policy>

```

```

1302         <sp:SC13SecurityContextToken />
1303         <sp:BootstrapPolicy>
1304             <wsp:Policy>
1305                 <sp:AsymmetricBinding>
1306                     <wsp:Policy>
1307                         <sp:InitiatorToken>
1308                             ...
1309                         </sp:InitiatorToken>
1310                         <sp:RecipientToken>
1311                             ...
1312                         </sp:RecipientToken>
1313                     </wsp:Policy>
1314                 </sp:AsymmetricBinding>
1315                 <sp:SignedParts>
1316                     ...
1317                 </sp:SignedParts>
1318                 ...
1319             </wsp:Policy>
1320         </sp:BootstrapPolicy>
1321     </wsp:Policy>
1322 </sp:SecureConversationToken>
1323 </wsp:Policy>
1324 </sp:ProtectionToken>
1325     ...
1326 </wsp:Policy>
1327 </sp:SymmetricBinding>
1328 <sp:SignedParts>
1329     ...
1330 </sp:SignedParts>
1331     ...
1332 </wsp:Policy>

```

### 1333 5.4.8 SamlToken Assertion

1334 This element represents a requirement for a SAML token.

#### 1335 Syntax

```

1336 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1337 (
1338     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1339     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1340 ) ?
1341 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1342 <wsp:Policy xmlns:wsp="...">
1343 (
1344     <sp:RequireDerivedKeys ... /> |
1345     <sp:RequireImpliedDerivedKeys ... /> |
1346     <sp:RequireExplicitDerivedKeys ... />
1347 ) ?
1348 <sp:RequireKeyIdentifierReference ... /> ?
1349 (
1350     <sp:WssSamlV11Token10 ... /> |
1351     <sp:WssSamlV11Token11 ... /> |
1352     <sp:WssSamlV20Token11 ... />
1353 ) ?
1354     ...
1355 </wsp:Policy>
1356     ...
1357 </sp:SamlToken>

```

1358

1359 The following describes the attributes and elements listed in the schema outlined above:



- 1360 /sp:SamIToken  
1361 This identifies a SamIToken assertion.
- 1362 /sp:SamIToken/@sp:IncludeToken  
1363 This OPTIONAL attribute identifies the token inclusion value for this token assertion.
- 1364 /sp:SamIToken/sp:Issuer  
1365 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1366 of the sp:SamIToken.
- 1367 /sp:SamIToken/sp:IssuerName  
1368 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:SamIToken  
1369 issuer.
- 1370 /sp:SamIToken/wst:Claims  
1371 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1372 order to satisfy the token assertion requirements.
- 1373 /sp:SamIToken/wsp:Policy  
1374 This REQUIRED element identifies additional requirements for use of the sp:SamIToken  
1375 assertion.
- 1376 /sp:SamIToken/wsp:Policy/sp:RequireDerivedKeys  
1377 This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1378 and [Implied Derived Keys] properties for this token to 'true'.
- 1379 /sp:SamIToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
1380 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
1381 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
1382 'false'.
- 1383 /sp:SamIToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
1384 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
1385 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
1386 'false'.
- 1387 /sp:SamIToken/wsp:Policy/sp:RequireKeyIdentifierReference  
1388 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
1389 REQUIRED when referencing this token.
- 1390 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10  
1391 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1392 should be used as defined in [[WSS:SAMLSecurityProfile1.0](#)].
- 1393 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11  
1394 This OPTIONAL element is a policy assertion that identifies that a SAML Version 1.1 token  
1395 should be used as defined in [[WSS:SAMLSecurityProfile1.1](#)].
- 1396 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11  
1397 This OPTIONAL element is a policy assertion that identifies that a SAML Version 2.0 token  
1398 should be used as defined in [[WSS:SAMLSecurityProfile1.1](#)].
- 1399
- 1400 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties  
1401 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition

1402 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion  
1403 SHOULD be used instead.

## 1404 5.4.9 RelToken Assertion

1405 This element represents a requirement for a REL token.

### 1406 Syntax

```
1407 <sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
1408 (   
1409   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |   
1410   <sp:IssuerName>xs:anyURI</sp:IssuerName>  
1411 ) ?   
1412 <wst:Claims Dialect="..."> ... </wst:Claims> ?   
1413 <wsp:Policy xmlns:wsp="...">  
1414   (   
1415     <sp:RequireDerivedKeys ... /> |   
1416     <sp:RequireImpliedDerivedKeys ... /> |   
1417     <sp:RequireExplicitDerivedKeys ... />  
1418   ) ?   
1419   <sp:RequireKeyIdentifierReference ... /> ?   
1420   (   
1421     <sp:WssRelV10Token10 ... /> |   
1422     <sp:WssRelV20Token10 ... /> |   
1423     <sp:WssRelV10Token11 ... /> |   
1424     <sp:WssRelV20Token11 ... />  
1425   ) ?   
1426   ...   
1427 </wsp:Policy>  
1428   ...   
1429 </sp:RelToken>
```

1430

1431 The following describes the attributes and elements listed in the schema outlined above:

1432 /sp:RelToken

1433       This identifies a RelToken assertion.

1434 /sp:RelToken/@sp:IncludeToken

1435       This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1436 /sp:RelToken/sp:Issuer

1437       This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1438       of the sp:RelToken.

1439 /sp:RelToken/sp:IssuerName

1440       This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:RelToken  
1441       issuer.

1442 /sp:RelToken/wst:Claims

1443       This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1444       order to satisfy the token assertion requirements.

1445 /sp:RelToken/wsp:Policy

1446       This REQUIRED element identifies additional requirements for use of the sp:RelToken assertion.

1447 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1448       This OPTIONAL element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]  
1449       and [Implied Derived Keys] property for this token to 'true'.

1450 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys  
 1451 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Explicit Derived  
 1452 Keys] properties for this token to 'true' and the [Implied Derived Keys] property for this token to  
 1453 'false'.

1454 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys  
 1455 This OPTIONAL element is a policy assertion that sets the [Derived Keys] and [Implied Derived  
 1456 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to  
 1457 'false'.

1458 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference  
 1459 This OPTIONAL element is a policy assertion that indicates that a key identifier reference is  
 1460 REQUIRED when referencing this token.

1461 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10  
 1462 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
 1463 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1464 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10  
 1465 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
 1466 be used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1467 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11  
 1468 This OPTIONAL element is a policy assertion that identifies that a REL Version 1.0 token should  
 1469 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1470 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11  
 1471 This OPTIONAL element is a policy assertion that identifies that a REL Version 2.0 token should  
 1472 be used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1473

1474 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties  
 1475 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition  
 1476 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion  
 1477 SHOULD be used instead.

## 1478 5.4.10 HttpsToken Assertion

1479 This element represents a requirement for a transport binding to support the use of HTTPS.

### 1480 Syntax

```

1481 <sp:HttpsToken xmlns:sp="..." ... >
1482   (
1483     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1484     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1485   ) ?
1486   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1487   <wsp:Policy xmlns:wsp="...">
1488     (
1489       <sp:HttpBasicAuthentication /> |
1490       <sp:HttpDigestAuthentication /> |
1491       <sp:RequireClientCertificate /> |
1492       ...
1493     ) ?
1494     ...
1495   </wsp:Policy>
1496   ...
1497 </sp:HttpsToken>
  
```

1498 The following describes the attributes and elements listed in the schema outlined above:

1499 /sp:HttpsToken

1500 This identifies an Https assertion stating that use of the HTTPS protocol specification is  
1501 supported.

1502 /sp:HttpsToken/sp:Issuer

1503 This OPTIONAL element, of type wsa:EndpointReferenceType, contains reference to the issuer  
1504 of the sp:HttpsToken.

1505 /sp:HttpsToken/sp:IssuerName

1506 This OPTIONAL element, of type xs:anyURI, contains the logical name of the sp:HttpsToken  
1507 issuer.

1508 /sp:HttpsToken/wst:Claims

1509 This OPTIONAL element identifies the REQUIRED claims that a security token must contain in  
1510 order to satisfy the token assertion requirements.

1511 /sp:HttpsToken/wsp:Policy

1512 This REQUIRED element identifies additional requirements for use of the sp:HttpsToken  
1513 assertion.

1514 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1515 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP Basic  
1516 Authentication [RFC2068] to authenticate to the service.

1517 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1518 This OPTIONAL element is a policy assertion that indicates that the client MUST use HTTP  
1519 Digest Authentication [RFC2068] to authenticate to the service.

1520 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1521 This OPTIONAL element is a policy assertion that indicates that the client MUST provide a  
1522 certificate when negotiating the HTTPS session.

## 1523 5.4.11 KeyValueToken Assertion

1524 This element represents a requirement for a KeyValue token. The next section defines the KeyValue  
1525 security token abstraction for purposes of this token assertion.

1526 This document defines requirements for KeyValue token when used in combination with RSA  
1527 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by  
1528 introducing new nested assertions besides *sp:RsaKeyValue*.  
1529

### 1530 Syntax

```
1531 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1532   <wsp:Policy xmlns:wsp="...">
1533     <sp:RsaKeyValue ... /> ?
1534     ...
1535   </wsp:Policy>
1536   ...
1537 </sp:KeyValueToken>
```

1538 The following describes the attributes listed in the schema outlined above:

1539 /sp:KeyValueToken

1540 This identifies a RsaToken assertion.

1541 /sp:KeyValueToken/@sp:IncludeToken

1542 This OPTIONAL attribute identifies the token inclusion value for this token assertion.

1543 /sp:KeyValueToken/wsp:Policy  
1544 This REQUIRED element identifies additional requirements for use of the sp:KeyValueToken  
1545 assertion.

1546 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue  
1547 This OPTIONAL element is a policy assertion that indicates that the ds:RSAKeyValue element  
1548 must be present in the KeyValue token. This indicates that an RSA key pair must be used.

### 1549 5.4.11.1 KeyValue Token

1550 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key  
1551 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this  
1552 section is to define the KeyValue token abstraction that represents such key pair referencing mechanism.

1553  
1554 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be  
1555 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*  
1556 element in combination with RSA cryptographic algorithm.

1557  
1558 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the  
1559 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

```
1560 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1561   <ds:KeyValue>  
1562     <ds:RSAKeyValue>  
1563       <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1564       <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1565     </ds:RSAKeyValue>  
1566   </ds:KeyValue>  
1567 </ds:KeyInfo>
```

1568  
1569 When the KeyValue token is used the corresponding public key value appears directly in the signature or  
1570 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValue token  
1571 manifestation outside the *ds:KeyInfo* element.

```
1572 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1573   <SignedInfo>  
1574     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1575 c14n#" />  
1576     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1577     <Reference URI="#_1">  
1578       <Transforms>  
1579         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1580       </Transforms>  
1581       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
1582       <DigestValue>...</DigestValue>  
1583     </Reference>  
1584   </SignedInfo>  
1585   <SignatureValue>...</SignatureValue>  
1586   <KeyInfo>  
1587     <KeyValue>  
1588       <RSAKeyValue>  
1589         <Modulus>...</Modulus>  
1590         <Exponent>...</Exponent>  
1591       </RSAKeyValue>  
1592     </KeyValue>  
1593   </KeyInfo>  
1594 </Signature>
```

1595  
1596 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no  
1597 identifier can be associated with the token, the KeyValue token cannot be referenced by using

1598 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the *KeyValue*  
1599 token can be used whenever a security token can be used as illustrated on the following example:

```
1600 <t:RequestSecurityToken xmlns:t="...">  
1601   <t:RequestType>...</t:RequestType>  
1602   ...  
1603   <t:UseKey>  
1604     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1605       <KeyValue>  
1606         <RSAKeyValue>  
1607           <Modulus>...</Modulus>  
1608           <Exponent>...</Exponent>  
1609         </RSAKeyValue>  
1610       </KeyValue>  
1611     </KeyInfo>  
1612   </t:UseKey>  
1613 </t:RequestSecurityToken>
```

---

## 6 Security Binding Properties

1614

1615 This section defines the various properties or conditions of a security binding, their semantics, values and  
1616 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are  
1617 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that  
1618 populates a value of a property appears in a policy, that property is set to the value indicated by the  
1619 assertion. The security binding then uses the value of the property to control its behavior. The properties  
1620 listed here are common to the various security bindings described in Section 7. Assertions that define  
1621 values for these properties are defined in Section 7. The following properties are used by the security  
1622 binding assertions.

### 6.1 [Algorithm Suite] Property

1623

1624 This property specifies the algorithm suite REQUIRED for performing cryptographic operations with  
1625 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and  
1626 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This  
1627 property defines the set of available algorithms. The value of this property is typically referenced by a  
1628 security binding and is used to specify the algorithms used for all message level cryptographic operations  
1629 performed under the security binding.

1630 Note: In some cases, this property MAY be referenced under a context other than a security binding and  
1631 used to control the algorithms used under that context. For example, supporting token assertions define  
1632 such a context. In such contexts, the specified algorithms still apply to message level cryptographic  
1633 operations.

1634 An algorithm suite defines values for each of the following operations and properties:

- 1635 • [Sym Sig] Symmetric Key Signature
- 1636 • [Asym Sig] Signature with an asymmetric key
- 1637 • [Dig] Digest
- 1638 • [Enc] Encryption
- 1639 • [Sym KW] Symmetric Key Wrap
- 1640 • [Asym KW] Asymmetric Key Wrap
- 1641 • [Comp Key] Computed key
- 1642 • [Enc KD] Encryption key derivation
- 1643 • [Sig KD] Signature key derivation
- 1644 • [Min SKL] Minimum symmetric key length
- 1645 • [Max SKL] Maximum symmetric key length
- 1646 • [Min AKL] Minimum asymmetric key length
- 1647 • [Max AKL] Maximum asymmetric key length

1648

1649 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/xmlenc#sha256">http://www.w3.org/2001/04/xmlenc#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>

1650

1651 The tables below show all the base algorithm suites defined by this specification. This table defines  
 1652 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

1653 This table defines additional properties whose values can be specified along with the default value for that  
 1654 property.

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

1655 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

## 1656 6.2 [Timestamp] Property

1657 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`  
1658 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected  
1659 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be  
1660 present. The default value for this property is 'false'.

## 1661 6.3 [Protection Order] Property

1662 This property indicates the order in which integrity and confidentiality are applied to the message, in  
1663 cases where both integrity and confidentiality are REQUIRED:

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

1664 The default value for this property is 'SignBeforeEncrypting'.

## 1665 6.4 [Signature Protection] Property

1666 This boolean property specifies whether the signature MUST be encrypted. If the value is 'true', the  
1667 primary signature MUST be encrypted and any signature confirmation elements MUST also be encrypted.  
1668 The primary signature element is NOT REQUIRED to be encrypted if the value is 'true' when there is  
1669 nothing in the message that is covered by this signature that is encrypted. If the value is 'false', the  
1670 primary signature MUST NOT be encrypted and any signature confirmation elements MUST NOT be  
1671 encrypted. The default value for this property is 'false'.

## 1672 6.5 [Token Protection] Property

1673 This boolean property specifies whether signatures MUST cover the token used to generate that  
1674 signature. If the value is 'true', then each token used to generate a signature MUST be covered by that  
1675 signature. If the value is 'false', then the token MUST NOT be covered by the signature. Note that in  
1676 cases where derived keys are used the 'main' token, and NOT the derived key token, is covered by the  
1677 signature. It is RECOMMENDED that assertions that define values for this property apply to [Endpoint  
1678 Policy Subject]. The default value for this property is 'false'.

## 1679 6.6 [Entire Header and Body Signatures] Property

1680 This boolean property specifies whether signature digests over the SOAP body and SOAP headers  
1681 MUST only cover the entire body and entire header elements. If the value is 'true', then each digest over  
1682 the SOAP body MUST be over the entire SOAP body element and not a descendant of that element. In  
1683 addition each digest over a SOAP header MUST be over an actual header element and not a descendant  
1684 of a header element. This restriction does not specifically apply to the wsse:Security header. However  
1685 signature digests over child elements of the wsse:Security header MUST be over the entire child element  
1686 and not a descendent of that element. If the value is 'false', then signature digests MAY be over a  
1687 descendant of the SOAP Body or a descendant of a header element. Setting the value of this property to  
1688 'true' mitigates against some possible re-writing attacks. It is RECOMENDED that assertions that define  
1689 values for this property apply to [Endpoint Policy Subject]. The default value for this property is 'false'.

## 1690 6.7 [Security Header Layout] Property

1691 This property indicates which layout rules to apply when adding items to the security header. The  
1692 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.

1693

### 1694 6.7.1 Strict Layout Rules for WSS 1.0

- 1695 1. Tokens that are included in the message MUST be declared before use. For example:
- 1696 a. A local signing token MUST occur before the signature that uses it.
- 1697 b. A local token serving as the source token for a derived key token MUST occur before that
- 1698 derived key token.
- 1699 c. A local encryption token MUST occur before the reference list that points to
- 1700 xenc:EncryptedData elements that use it.
- 1701 d. If the same token is used for both signing and encryption, then it SHOULD appear before
- 1702 the ds:Signature and xenc:ReferenceList elements in the security header that are
- 1703 generated using the token.
- 1704 2. Signed elements inside the security header MUST occur before the signature that signs them.
- 1705 For example:
- 1706 a. A timestamp MUST occur before the signature that signs it.

- 1707           b. A Username token (usually in encrypted form) MUST occur before the signature that  
1708           signs it.
- 1709           c. A primary signature MUST occur before the supporting token signature that signs the  
1710           primary signature's signature value element.
- 1711       3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element  
1712       has the same order requirements as the source plain text element, unless requirement 4  
1713       indicates otherwise. For example, an encrypted primary signature MUST occur before any  
1714       supporting token signature per 2.c above and an encrypted token has the same ordering  
1715       requirements as the unencrypted token.

1716       If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top  
1717       level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the  
1718       security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any  
1719       xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict  
1720       Layout Rules for WSS 1.1

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

---

## 7 Security Binding Assertions

1750

1751 The appropriate representation of the different facets of security mechanisms requires distilling the  
1752 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy  
1753 scope of assertions defined in this section is the policy scope of their containing element.

### 7.1 AlgorithmSuite Assertion

1754

1755 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]  
1756 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

#### Syntax

1757

```
1758 <sp:AlgorithmSuite xmlns:sp="..." ... >  
1759   <wsp:Policy xmlns:wsp="...">  
1760     (<sp:Basic256 ... /> |  
1761     <sp:Basic192 ... /> |  
1762     <sp:Basic128 ... /> |  
1763     <sp:TripleDes ... /> |  
1764     <sp:Basic256Rsa15 ... /> |  
1765     <sp:Basic192Rsa15 ... /> |  
1766     <sp:Basic128Rsa15 ... /> |  
1767     <sp:TripleDesRsa15 ... /> |  
1768     <sp:Basic256Sha256 ... /> |  
1769     <sp:Basic192Sha256 ... /> |  
1770     <sp:Basic128Sha256 ... /> |  
1771     <sp:TripleDesSha256 ... /> |  
1772     <sp:Basic256Sha256Rsa15 ... /> |  
1773     <sp:Basic192Sha256Rsa15 ... /> |  
1774     <sp:Basic128Sha256Rsa15 ... /> |  
1775     <sp:TripleDesSha256Rsa15 ... /> |  
1776     ...)  
1777     <sp:InclusiveC14N ... /> ?  
1778     <sp:InclusiveC14N11 ... /> ?  
1779     <sp:SOAPNormalization10 ... /> ?  
1780     <sp:STRTransform10 ... /> ?  
1781     (<sp:XPath10 ... /> |  
1782     <sp:XPathFilter20 ... /> |  
1783     <sp:AbsXPath ... /> |  
1784     ...)?  
1785     ...  
1786   </wsp:Policy>  
1787   ...  
1788 </sp:AlgorithmSuite>
```

1789

1790 The following describes the attributes and elements listed in the schema outlined above:

1791 /sp:AlgorithmSuite

1792     This identifies an AlgorithmSuite assertion.

1793 /sp:AlgorithmSuite/wsp:Policy

1794     This REQUIRED element contains one or more policy assertions that indicate the specific  
1795     algorithm suite to use.

1796 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1797     This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1798     set to 'Basic256'.

- 1799 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192
- 1800 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1801 set to 'Basic192'.
- 1802 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128
- 1803 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1804 set to 'Basic128'.
- 1805 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes
- 1806 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1807 set to 'TripleDes'.
- 1808 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15
- 1809 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1810 set to 'Basic256Rsa15'.
- 1811 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15
- 1812 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1813 set to 'Basic192Rsa15'.
- 1814 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15
- 1815 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1816 set to 'Basic128Rsa15'.
- 1817 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15
- 1818 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1819 set to 'TripleDesRsa15'.
- 1820 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256
- 1821 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1822 set to 'Basic256Sha256'.
- 1823 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256
- 1824 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1825 set to 'Basic192Sha256'.
- 1826 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256
- 1827 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1828 set to 'Basic128Sha256'.
- 1829 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256
- 1830 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1831 set to 'TripleDesSha256'.
- 1832 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15
- 1833 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1834 set to 'Basic256Sha256Rsa15'.
- 1835 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15
- 1836 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1837 set to 'Basic192Sha256Rsa15'.
- 1838 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15
- 1839 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1840 set to 'Basic128Sha256Rsa15'.
- 1841 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1842 This OPTIONAL element is a policy assertion that indicates that the [Algorithm Suite] property is  
1843 set to 'TripleDesSha256Rsa15'.

1844 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1845 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an  
1846 algorithm suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N]  
1847 property is 'ExC14N'.

1848 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N11

1849 This OPTIONAL element is a policy assertion that indicates that the [C14N] property of an  
1850 algorithm suite is set to 'C14N11'. Note: as indicated in Section 6.1 the default value of the  
1851 [C14N] property is 'ExC14N'.

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

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

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

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

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

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

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

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

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

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

1867

## 1868 7.2 Layout Assertion

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

### 1872 Syntax

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

1883

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

1885 /sp:Layout

1886 This identifies a Layout assertion.

1887 /sp:Layout/wsp:Policy

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

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

1891 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1892 property is set to 'Strict'.

1893 /sp:Layout/wsp:Policy/sp:Lax

1894 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1895 property is set to 'Lax'.

1896 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1897 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1898 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to  
1899 'true' by the presence of an sp:IncludeTimestamp assertion.

1900 /sp:Layout/wsp:Policy/sp:LaxTsLast

1901 This OPTIONAL element is a policy assertion that indicates that the [Security Header Layout]  
1902 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to  
1903 'true' by the presence of an sp:IncludeTimestamp assertion.

## 1904 7.3 TransportBinding Assertion

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

### 1910 Syntax

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

1924

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

1926 /sp:TransportBinding

1927 This identifies a TransportBinding assertion.

1928 /sp:TransportBinding/wsp:Policy

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

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



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

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

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

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

1938 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
1939 Suite] property. See Section 6.1 for more details.

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

1941 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
1942 Header Layout] property. See Section 6.7 for more details.

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

1944 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
1945 to 'true'.

## 1946 7.4 SymmetricBinding Assertion

1947 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means  
1948 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;  
1949 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this  
1950 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to  
1951 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to  
1952 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token  
1953 properties and is used as the basis for both encryption and signature in both directions. This assertion  
1954 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

### 1955 Syntax

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

1981

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



- 1983 /sp:SymmetricBinding
- 1984 This identifies a SymmetricBinding assertion.
- 1985 /sp:SymmetricBinding/wsp:Policy
- 1986 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding  
1987 assertion.
- 1988 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
- 1989 This OPTIONAL element is a policy assertion that indicates a requirement for an Encryption  
1990 Token. The specified token populates the [Encryption Token] property and is used for encryption.  
1991 It is an error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.
- 1992 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
- 1993 The policy contained here MUST identify exactly one token to use for encryption.
- 1994 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
- 1995 This OPTIONAL element is a policy assertion that indicates a requirement for a Signature Token.  
1996 The specified token populates the [Signature Token] property and is used for the message  
1997 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be  
1998 specified.
- 1999 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
- 2000 The policy contained here MUST identify exactly one token to use for signatures.
- 2001 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
- 2002 This OPTIONAL element is a policy assertion that indicates a requirement for a Protection Token.  
2003 The specified token populates the [Encryption Token] and [Signature Token properties] and is  
2004 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken  
2005 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be  
2006 specified.
- 2007 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
- 2008 The policy contained here MUST identify exactly one token to use for protection.
- 2009 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
- 2010 This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
2011 Suite] property. See Section 6.1 for more details.
- 2012 /sp:SymmetricBinding/wsp:Policy/sp:Layout
- 2013 This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
2014 Header Layout] property. See Section 6.7 for more details.
- 2015 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
- 2016 This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
2017 to 'true'.
- 2018 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
- 2019 This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
2020 set to 'EncryptBeforeSigning'.
- 2021 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
- 2022 This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
2023 property is set to 'true'.
- 2024 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
- 2025 This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
2026 set to 'true'.

2027 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
2028 This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
2029 Signatures] property is set to 'true'.

## 2030 7.5 AsymmetricBinding Assertion

2031 The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means  
2032 defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology. Commonly  
2033 used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and  
2034 signature. However it is also common practice to use distinct keys for encryption and signature, because  
2035 of their different lifecycles.

2036  
2037 This binding enables either of these practices by means of four binding specific token properties: [Initiator  
2038 Signature Token], [Initiator Encryption Token], [Recipient Signature Token] and [Recipient Encryption  
2039 Token].

2040  
2041 If the same key pair is used for signature and encryption, then [Initiator Signature Token] and [Initiator  
2042 Encryption Token] will both refer to the same token. Likewise [Recipient Signature Token] and [Recipient  
2043 Encryption Token] will both refer to the same token.

2044  
2045 If distinct key pairs are used for signature and encryption then [Initiator Signature Token] and [Initiator  
2046 Encryption Token] will refer to different tokens. Likewise [Recipient Signature Token] and [Recipient  
2047 Encryption Token] will refer to different tokens.

2048  
2049 If the message pattern requires multiple messages, the [Initiator Signature Token] is used for the  
2050 message signature from initiator to the recipient. The [Initiator Encryption Token] is used for the response  
2051 message encryption from recipient to the initiator. The [Recipient Signature Token] is used for the  
2052 response message signature from recipient to the initiator. The [Recipient Encryption Token] is used for  
2053 the message encryption from initiator to the recipient. Note that in each case, the token is associated with  
2054 the party (initiator or recipient) who knows the secret.

2055 This assertion SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy  
2056 Subject].

### 2057 Syntax

```
2058 <sp:AsymmetricBinding xmlns:sp="..." ... >  
2059   <wsp:Policy xmlns:wsp="...">  
2060     (  
2061       <sp:InitiatorToken>  
2062         <wsp:Policy> ... </wsp:Policy>  
2063       </sp:InitiatorToken>  
2064     ) | (  
2065       <sp:InitiatorSignatureToken>  
2066         <wsp:Policy> ... </wsp:Policy>  
2067       </sp:InitiatorSignatureToken>  
2068       <sp:InitiatorEncryptionToken>  
2069         <wsp:Policy> ... </wsp:Policy>  
2070       </sp:InitiatorEncryptionToken>  
2071     )  
2072     (  
2073       <sp:RecipientToken>  
2074         <wsp:Policy> ... </wsp:Policy>  
2075       </sp:RecipientToken>  
2076     ) | (  
2077
```

```

2077     <sp:RecipientSignatureToken>
2078         <wsp:Policy> ... </wsp:Policy>
2079     </sp:RecipientSignatureToken>
2080     <sp:RecipientEncryptionToken>
2081         <wsp:Policy> ... </wsp:Policy>
2082     </sp:RecipientEncryptionToken>
2083 )
2084 <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2085 <sp:Layout ... > ... </sp:Layout> ?
2086 <sp:IncludeTimestamp ... /> ?
2087 <sp:EncryptBeforeSigning ... /> ?
2088 <sp:EncryptSignature ... /> ?
2089 <sp:ProtectTokens ... /> ?
2090 <sp:OnlySignEntireHeadersAndBody ... /> ?
2091 ...
2092 </wsp:Policy>
2093 ...
2094 </sp:AsymmetricBinding>

```

2095  
2096 The following describes the attributes and elements listed in the schema outlined above:

2097 /sp:AsymmetricBinding

2098 This identifies a AsymmetricBinding assertion.

2099 /sp:AsymmetricBinding/wsp:Policy

2100 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding  
2101 assertion.

2102 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2103 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator Token.  
2104 The specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]  
2105 properties and is used for the message signature from initiator to recipient, and encryption from  
2106 recipient to initiator.

2107 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2108 The policy contained here MUST identify one or more token assertions.

2109 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2110 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2111 Signature Token. The specified token populates the [Initiator Signature Token] property and is  
2112 used for the message signature from initiator to recipient.

2113 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2114 The policy contained here MUST identify one or more token assertions.

2115 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2116 This OPTIONAL element is a policy assertion that indicates a requirement for an Initiator  
2117 Encryption Token. The specified token populates the [Initiator Encryption Token] property and is  
2118 used for the message encryption from recipient to initiator.

2119 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2120 The policy contained here MUST identify one or more token assertions.

2121 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2122 This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient Token.  
2123 The specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]  
2124 property and is used for encryption from initiator to recipient, and for the message signature from  
2125 recipient to initiator.

- 2126 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy  
 2127       The policy contained here MUST identify one or more token assertions.
- 2128 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken  
 2129       This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
 2130       Signature Token. The specified token populates the [Recipient Signature Token] property and is  
 2131       used for the message signature from recipient to initiator.
- 2132 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy  
 2133       The policy contained here MUST identify one or more token assertions.
- 2134 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken  
 2135       This OPTIONAL element is a policy assertion that indicates a requirement for a Recipient  
 2136       Encryption Token. The specified token populates the [Recipient Encryption Token] property and  
 2137       is used for the message encryption from initiator to recipient.
- 2138 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy  
 2139       The policy contained here MUST identify one or more token assertions.
- 2140 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite  
 2141       This REQUIRED element is a policy assertion that indicates a value that populates the [Algorithm  
 2142       Suite] property. See Section 6.1 for more details.
- 2143 /sp:AsymmetricBinding/wsp:Policy/sp:Layout  
 2144       This OPTIONAL element is a policy assertion that indicates a value that populates the [Security  
 2145       Header Layout] property. See Section 6.7 for more details.
- 2146 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp  
 2147       This OPTIONAL element is a policy assertion that indicates that the [Timestamp] property is set  
 2148       to 'true'.
- 2149 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning  
 2150       This OPTIONAL element is a policy assertion that indicates that the [Protection Order] property is  
 2151       set to 'EncryptBeforeSigning'.
- 2152 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature  
 2153       This OPTIONAL element is a policy assertion that indicates that the [Signature Protection]  
 2154       property is set to 'true'.
- 2155 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens  
 2156       This OPTIONAL element is a policy assertion that indicates that the [Token Protection] property is  
 2157       set to 'true'.
- 2158 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody  
 2159       This OPTIONAL element is a policy assertion that indicates that the [Entire Header And Body  
 2160       Signatures] property is set to 'true'.

2161

## 8 Supporting Tokens

2162 Security Bindings use tokens to secure the message exchange. The Security Binding will require one to  
2163 create a signature using the token identified in the Security Binding policy. This signature will here-to-fore  
2164 be referred to as the “message signature”. In case of Transport Binding the message is signed outside of  
2165 the message XML by the underlying transport protocol and the signature itself is not part of the message.  
2166 Additional tokens MAY be specified to augment the claims provided by the token associated with the  
2167 “message signature” provided by the Security Binding. This section defines seven properties related to  
2168 supporting token requirements which MAY be referenced by a Security Binding: [Supporting Tokens],  
2169 [Signed Supporting Tokens], [Endorsing Supporting Tokens], [Signed Endorsing Supporting Tokens],  
2170 [Signed Encrypted Supporting Tokens], [Endorsing Encrypted Supporting Tokens] and [Signed Endorsing  
2171 Encrypted Supporting Tokens]. Seven assertions are defined to populate those properties:  
2172 SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens,  
2173 SignedEndorsingSupportingTokens, SignedEncryptedSupportingTokens,  
2174 EndorsingEncryptedSupportingTokens and SignedEndorsingEncryptedSupportingTokens. These  
2175 assertions SHOULD apply to [Endpoint Policy Subject]. These assertions MAY apply to [Message Policy  
2176 Subject] or [Operation Policy Subject].

2177

2178 Supporting tokens MAY be specified at a different scope than the binding assertion which provides  
2179 support for securing the exchange. For instance, a binding is specified at the scope of an endpoint, while  
2180 the supporting tokens might be defined at the scope of a message. When assertions that populate this  
2181 property are defined in overlapping scopes, the sender SHOULD merge the requirements by including all  
2182 tokens from the outer scope and any additional tokens for a specific message from the inner scope.

2183

2184 In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the  
2185 tokens SHOULD sign and encrypt the various message parts. In such cases ordering of elements  
2186 (tokens, signatures, reference lists etc.) in the security header would be used to determine which order  
2187 signature and encryptions occurred in.

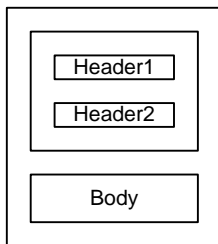
2188

2189 Policy authors need to ensure that the tokens they specify as supporting tokens can satisfy any additional  
2190 constraints defined by the supporting token assertion. For example, if the supporting token assertion  
2191 specifies message parts that need to be encrypted, the specified tokens need to be capable of  
2192 encryption.

2193

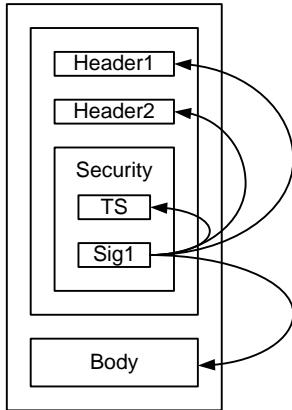
2194 To illustrate the different ways that supporting tokens MAY be bound to the message, let’s consider a  
2195 message with three components: Header1, Header2, and Body.

2196

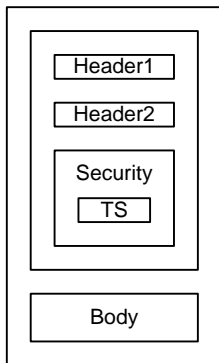


2197

2198 Even before any supporting tokens are added, each binding requires that the message is signed using a  
 2199 token satisfying the REQUIRED usage for that binding, and that the signature (Sig1) covers important  
 2200 parts of the message including the message timestamp (TS) facilitate replay detection. The signature is  
 2201 then included as part of the Security header as illustrated below:  
 2202



2203  
 2204 Note: if REQUIRED, the initiator may also include in the Security header the token used as the basis for  
 2205 the message signature (Sig1), not shown in the diagram.  
 2206 If transport security is used, only the message timestamp (TS) is included in the Security header as  
 2207 illustrated below. The “message signature” is provided by the underlying transport protocol and is not part  
 2208 of the message XML.



2209

## 2210 8.1 SupportingTokens Assertion

2211 Supporting tokens are included in the security header and MAY OPTIONALLY include additional  
 2212 message parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and  
 2213 do not require any protection (signature or encryption) to be applied to the message before they are  
 2214 added. More specifically there is no requirement on “message signature” being present before the  
 2215 supporting tokens are added. However it is RECOMMENDED to employ underlying protection  
 2216 mechanism to ensure that the supporting tokens are cryptographically bound to the message during the  
 2217 transmission.

### 2218 Syntax

```
2219 <sp:SupportingTokens xmlns:sp="..." ... >
2220   <wsp:Policy xmlns:wsp="...">
2221     [Token Assertion]+
2222     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2223     (
2224       <sp:SignedParts ... > ... </sp:SignedParts> |
```

```

2225     <sp:SignedElements ... > ... </sp:SignedElements> |
2226     <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2227     <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2228     <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2229   ) *
2230   ...
2231 </wsp:Policy>
2232   ...
2233 </sp:SupportingTokens>

```

2234

2235 The following describes the attributes and elements listed in the schema outlined above:

2236 /sp:SupportingTokens

2237 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting  
2238 Tokens] property.

2239 /sp:SupportingTokens/wsp:Policy

2240 This describes additional requirements for satisfying the SupportingTokens assertion.

2241 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2242 The policy MUST identify one or more token assertions.

2243 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2244 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2245 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2246 by this policy assertion.

2247 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2248 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2249 and describes additional message parts that MUST be included in the signature generated with  
2250 the token identified by this policy assertion.

2251 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2252 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2253 and describes additional message elements that MUST be included in the signature generated  
2254 with the token identified by this policy assertion.

2255 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2256 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2257 and describes additional message parts that MUST be encrypted using the token identified by  
2258 this policy assertion.

2259 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2260 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2261 and describes additional message elements that MUST be encrypted using the token identified  
2262 by this policy assertion.

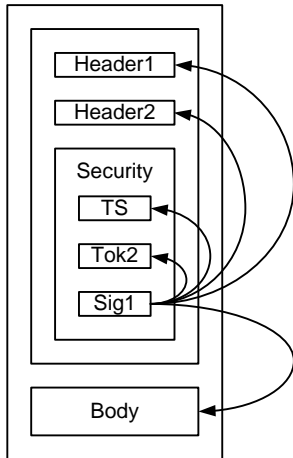
2263 /sp:SupportingTokens/wsp:Policy/sp:ContentEncryptedElements

2264 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2265 and describes additional message elements whose content MUST be encrypted using the token  
2266 identified by this policy assertion.

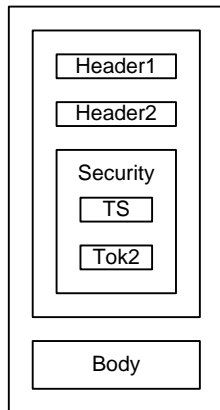
2267

2268 **8.2 SignedSupportingTokens Assertion**

2269 Signed tokens are included in the “message signature” as defined above and MAY OPTIONALLY include  
2270 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token  
2271 (Tok2) is signed by the message signature (Sig1):  
2272



2273  
2274 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:  
2275



2276  
2277 **Syntax**

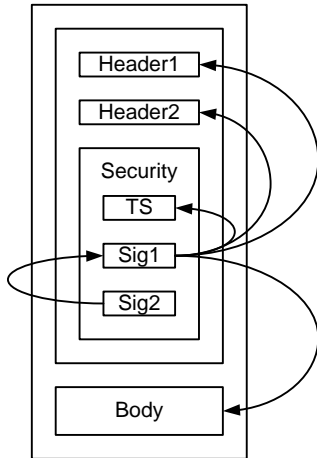
```
2278 <sp:SignedSupportingTokens xmlns:sp="..." ... >  
2279   <wsp:Policy xmlns:wsp="...">  
2280     [Token Assertion]+  
2281     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?  
2282     (  
2283       <sp:SignedParts ... > ... </sp:SignedParts> |  
2284       <sp:SignedElements ... > ... </sp:SignedElements> |  
2285       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |  
2286       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |  
2287       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>  
2288     ) *  
2289     ...  
2290   </wsp:Policy>  
2291   ...  
2292 </sp:SignedSupportingTokens>
```



2294 The following describes the attributes and elements listed in the schema outlined above:  
2295 /sp:SignedSupportingTokens  
2296 This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed  
2297 Supporting Tokens] property.  
2298 /sp:SignedSupportingTokens/wsp:Policy  
2299 This describes additional requirements for satisfying the SignedSupportingTokens assertion.  
2300 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]  
2301 The policy MUST identify one or more token assertions.  
2302 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite  
2303 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2304 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2305 by this policy assertion.  
2306 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts  
2307 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2308 and describes additional message parts that MUST be included in the signature generated with  
2309 the token identified by this policy assertion.  
2310 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements  
2311 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2  
2312 and describes additional message elements that MUST be included in the signature generated  
2313 with the token identified by this policy assertion.  
2314 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts  
2315 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2316 and describes additional message parts that MUST be encrypted using the token identified by  
2317 this policy assertion.  
2318 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements  
2319 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2320 and describes additional message elements that MUST be encrypted using the token identified  
2321 by this policy assertion.  
2322 /sp:SignedSupportingTokens/wsp:Policy/sp:ContentEncryptedElements  
2323 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2324 and describes additional message elements whose content MUST be encrypted using the token  
2325 identified by this policy assertion.

### 2326 **8.3 EndorsingSupportingTokens Assertion**

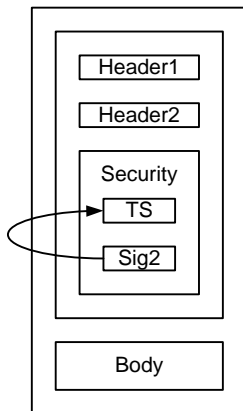
2327 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element  
2328 produced from the message signature and MAY OPTIONALLY include additional message parts to sign  
2329 and/or encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message  
2330 signature (Sig1):  
2331



2332

2333 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated  
 2334 below:

2335



2336

2337 **Syntax**

```

2338 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2339   <wsp:Policy xmlns:wsp="...">
2340     [Token Assertion]+
2341     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2342     (
2343       <sp:SignedParts ... > ... </sp:SignedParts> |
2344       <sp:SignedElements ... > ... </sp:SignedElements> |
2345       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2346       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2347       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2348     ) *
2349     ...
2350   </wsp:Policy>
2351   ...
2352 </sp:EndorsingSupportingTokens>
  
```

2353

2354 The following describes the attributes and elements listed in the schema outlined above:

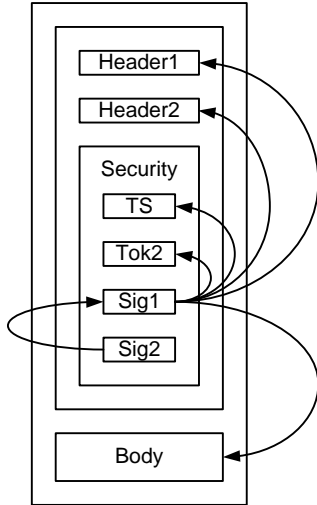
2355 /sp:EndorsingSupportingTokens

2356 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the  
 2357 [Endorsing Supporting Tokens] property.

- 2358 /sp:EndorsingSupportingTokens/wsp:Policy
- 2359 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.
- 2360 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]
- 2361 The policy MUST identify one or more token assertions.
- 2362 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite
- 2363 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and
- 2364 describes the algorithms to use for cryptographic operations performed with the tokens identified
- 2365 by this policy assertion.
- 2366 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts
- 2367 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1
- 2368 and describes additional message parts that MUST be included in the signature generated with
- 2369 the token identified by this policy assertion.
- 2370 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements
- 2371 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.2
- 2372 and describes additional message elements that MUST be included in the signature generated
- 2373 with the token identified by this policy assertion.
- 2374 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
- 2375 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1
- 2376 and describes additional message parts that MUST be encrypted using the token identified by
- 2377 this policy assertion.
- 2378 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
- 2379 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2
- 2380 and describes additional message elements that MUST be encrypted using the token identified
- 2381 by this policy assertion.
- 2382 /sp:EndorsingSupportingTokens/wsp:Policy/sp:ContentEncryptedElements
- 2383 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3
- 2384 and describes additional message elements whose content MUST be encrypted using the token
- 2385 identified by this policy assertion.

## 2386 **8.4 SignedEndorsingSupportingTokens Assertion**

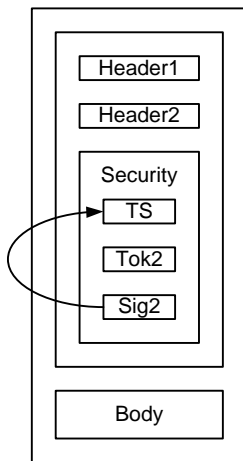
- 2387 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
- 2388 and are themselves signed by that message signature, that is both tokens (the token used for the
- 2389 message signature and the signed endorsing token) sign each other. This assertion MAY OPTIONALLY
- 2390 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
- 2391 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
- 2392 message signature (Sig1):
- 2393



2394

2395 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)  
 2396 SHOULD cover the message timestamp as illustrated below:

2397



2398

2399 **Syntax**

```

2400 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2401   <wsp:Policy xmlns:wsp="...">
2402     [Token Assertion]+
2403     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2404     (
2405       <sp:SignedParts ... > ... </sp:SignedParts> |
2406       <sp:SignedElements ... > ... </sp:SignedElements> |
2407       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2408       <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2409       <sp:ContentEncryptedElements ... > ... </sp:ContentEncryptedElements>
2410     ) *
2411     ...
2412   </wsp:Policy>
2413   ...
2414 </sp:SignedEndorsingSupportingTokens>
  
```

2415

2416 The following describes the attributes and elements listed in the schema outlined above:

2417 /sp:SignedEndorsingSupportingTokens

2418 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the  
2419 [Signed Endorsing Supporting Tokens] property.

2420 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2421 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2422 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2423 The policy MUST identify one or more token assertions.

2424 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2425 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 7.1 and  
2426 describes the algorithms to use for cryptographic operations performed with the tokens identified  
2427 by this policy assertion.

2428 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2429 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.1.1  
2430 and describes additional message parts that MUST be included in the signature generated with  
2431 the token identified by this policy assertion.

2432 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2433 This OPTIONAL element follows the schema outlined in Section 4.1.2 and describes additional  
2434 message elements that MUST be included in the signature generated with the token identified by  
2435 this policy assertion.

2436 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2437 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.1  
2438 and describes additional message parts that MUST be encrypted using the token identified by  
2439 this policy assertion.

2440 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2441 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.2  
2442 and describes additional message elements that MUST be encrypted using the token identified  
2443 by this policy assertion.

2444 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:ContentEncryptedElements

2445 This OPTIONAL element is a policy assertion that follows the schema outlined in Section 4.2.3  
2446 and describes additional message elements whose content MUST be encrypted using the token  
2447 identified by this policy assertion.

## 2448 **8.5 SignedEncryptedSupportingTokens Assertion**

2449 Signed, encrypted supporting tokens are Signed supporting tokens (See section 8.2) that are also  
2450 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2451 encrypting the supporting tokens.

2452 The syntax for the sp:SignedEncryptedSupportingTokens differs from the syntax of  
2453 sp:SignedSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2454 sp:SignedSupportingTokens assertion.

## 2455 **8.6 EncryptedSupportingTokens Assertion**

2456 Encrypted supporting tokens are supporting tokens (See section 8.1) that are included in the security  
2457 header and MUST be encrypted when they appear in the security header. Element encryption SHOULD  
2458 be used for encrypting these tokens. The encrypted supporting tokens can be added to any SOAP  
2459 message and do not require the “message signature” being present before the encrypted supporting  
2460 tokens are added.

2461 The syntax for the sp:EncryptedSupportingTokens differs from the syntax of sp:SupportingTokens only in  
2462 the name of the assertion itself. All nested policy is as per the sp:SupportingTokens assertion.

2463 The encrypted supporting tokens SHOULD be used only when the sender cannot provide the “message  
2464 signature” and it is RECOMMENDED that the receiver employs some security mechanisms external to  
2465 the message to prevent the spoofing attacks. In all other cases it is RECOMMENDED to use signed  
2466 encrypted supporting tokens instead to ensure that the encrypted tokens are cryptographically bound to  
2467 the message (See section 8.5).

## 2468 **8.7 EndorsingEncryptedSupportingTokens Assertion**

2469 Endorsing, encrypted supporting tokens are Endorsing supporting tokens (See section 8.3) that are also  
2470 encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD be used for  
2471 encrypting the supporting tokens.

2472 The syntax for the sp:EndorsingEncryptedSupportingTokens differs from the syntax of  
2473 sp:EndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per the  
2474 sp:EndorsingSupportingTokens assertion.

## 2475 **8.8 SignedEndorsingEncryptedSupportingTokens Assertion**

2476 Signed, endorsing, encrypted supporting tokens are signed, endorsing supporting tokens (See section  
2477 8.4) that are also encrypted when they appear in the wsse:SecurityHeader. Element Encryption SHOULD  
2478 be used for encrypting the supporting tokens.

2479 The syntax for the sp:SignedEndorsingEncryptedSupportingTokens differs from the syntax of  
2480 sp:SignedEndorsingSupportingTokens only in the name of the assertion itself. All nested policy is as per  
2481 the sp:SignedEndorsingSupportingTokens assertion.

## 2482 **8.9 Interaction between [Token Protection] property and supporting 2483 token assertions**

2484 If [Token Protection] (see Section 6.5) is true, then each signature covers the token that generated that  
2485 signature and the following statements hold with respect to the various tokens that sign or are signed;

- 2486 • The message signature, generated from the [Initiator Token] in the Asymmetric Binding case or  
2487 the [Signature Token] in the Symmetric binding case, covers that token.
- 2488 • Endorsing signatures cover the main signature and the endorsing token.
- 2489 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the  
2490 message signature and once by the endorsing signature.

2491 In addition, signed supporting tokens are covered by the message signature, although this is independent  
2492 of [Token Protection].

## 2493 **8.10 Example**

2494 Example policy containing supporting token assertions:

```
2495 <!-- Example Endpoint Policy -->
```

```

2496 <wsp:Policy xmlns:wsp="...">
2497   <sp:SymmetricBinding xmlns:sp="...">
2498     <wsp:Policy>
2499       <sp:ProtectionToken>
2500         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2501           <sp:Issuer>...</sp:Issuer>
2502           <sp:RequestSecurityTokenTemplate>
2503             ...
2504           </sp:RequestSecurityTokenTemplate>
2505         </sp:IssuedToken>
2506       </sp:ProtectionToken>
2507       <sp:AlgorithmSuite>
2508         <wsp:Policy>
2509           <sp:Basic256 />
2510         </wsp:Policy>
2511       </sp:AlgorithmSuite>
2512       ...
2513     </wsp:Policy>
2514   </sp:SymmetricBinding>
2515   ...
2516   <sp:SignedSupportingTokens>
2517     <wsp:Policy>
2518       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2519     </wsp:Policy>
2520   </sp:SignedSupportingTokens>
2521   <sp:SignedEndorsingSupportingTokens>
2522     <wsp:Policy>
2523       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2524         <wsp:Policy>
2525           <sp:WssX509v3Token10 />
2526         </wsp:Policy>
2527       </sp:X509Token>
2528     </wsp:Policy>
2529   </sp:SignedEndorsingSupportingTokens>
2530   ...
2531 </wsp:Policy>

```

2532 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be  
2533 included in the security header and covered by the message signature. The  
2534 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the  
2535 security header and covered by the message signature. In addition, a signature over the message  
2536 signature based on the key material associated with the X509 certificate must be included in the security  
2537 header.

2538

## 9 WSS: SOAP Message Security Options

2539 There are several OPTIONAL aspects to the WSS: SOAP Message Security specification that are  
2540 independent of the trust and token taxonomies. This section describes another class of properties and  
2541 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The  
2542 assertions defined here MUST apply to [Endpoint Policy Subject].

2543 The properties and assertions dealing with token references defined in this section indicate whether the  
2544 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and  
2545 recipient MAY send a fault if such references are encountered.

2546

2547 Note: This approach is chosen because:

2548 A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used  
2549 in a single reference.

2550 B) In a multi-message exchange, a token MAY be referenced using different mechanisms depending  
2551 on which of a series of messages is being secured.

2552

2553 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a  
2554 `wsse:InvalidSecurity` fault.

2555

### 2556 **WSS: SOAP Message Security 1.0 Properties**

#### 2557 **[Direct References]**

2558 This property indicates whether the initiator and recipient MUST be able to process direct token  
2559 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations  
2560 MUST be able to process such references.

2561

#### 2562 **[Key Identifier References]**

2563 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific  
2564 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2565 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST  
2566 NOT generate such references and that the initiator and recipient MAY send a fault if such references are  
2567 encountered. This property has a default value of 'false'.

2568

#### 2569 **[Issuer Serial References]**

2570 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2571 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST  
2572 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT  
2573 generate such references and that the initiator and recipient MAY send a fault if such references are  
2574 encountered. This property has a default value of 'false'.

2575

#### 2576 **[External URI References]**

2577 This boolean property indicates whether the initiator and recipient MUST be able to process references to  
2578 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST  
2579 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT



2580 generate such references and that the initiator and recipient MAY send a fault if such references are  
2581 encountered. This property has a default value of 'false'.

### 2582 **[Embedded Token References]**

2583 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2584 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to  
2585 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2586 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2587 This property has a default value of 'false'.

2588

## 2589 **WSS: SOAP Message Security 1.1 Properties**

### 2590 **[Thumbprint References]**

2591 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2592 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to  
2593 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2594 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2595 This property has a default value of 'false'.

2596

### 2597 **[EncryptedKey References]**

2598 This boolean property indicates whether the initiator and recipient MUST be able to process references  
2599 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to  
2600 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate  
2601 such references and that the initiator and recipient MAY send a fault if such references are encountered.  
2602 This property has a default value of 'false'.

2603

### 2604 **[Signature Confirmation]**

2605 This boolean property specifies whether `wss11:SignatureConfirmation` elements SHOULD be  
2606 used as defined in WSS: Soap Message Security 1.1. If the value is 'true',  
2607 `wss11:SignatureConfirmation` elements MUST be used and signed by the message signature. If  
2608 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property  
2609 applies to all signatures that are included in the security header. This property has a default value of  
2610 'false'.

## 2611 **9.1 Wss10 Assertion**

2612 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are  
2613 supported.

### 2614 **Syntax**

```
2615 <sp:Wss10 xmlns:sp="..." ... >  
2616   <wsp:Policy xmlns:wsp="...">  
2617     <sp:MustSupportRefKeyIdentifier ... /> ?  
2618     <sp:MustSupportRefIssuerSerial ... /> ?  
2619     <sp:MustSupportRefExternalURI ... /> ?  
2620     <sp:MustSupportRefEmbeddedToken ... /> ?  
2621     ...  
2622   </wsp:Policy>  
2623   ...  
2624 </sp:Wss10>
```

2625

2626 The following describes the attributes and elements listed in the schema outlined above:

2627 /sp:Wss10  
 2628 This identifies a WSS10 assertion.  
 2629 /sp:Wss10/wsp:Policy  
 2630 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.  
 2631 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2632 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2633 property is set to 'true'.  
 2634 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2635 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2636 property is set to 'true'.  
 2637 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI  
 2638 This OPTIONAL element is a policy assertion indicates that the [External URI References]  
 2639 property is set to 'true'.  
 2640 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken  
 2641 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
 2642 property is set to 'true'.

## 2643 9.2 Wss11 Assertion

2644 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are  
 2645 supported.

### 2646 Syntax

```

2647 <sp:Wss11 xmlns:sp="..." ... >
2648   <wsp:Policy xmlns:wsp="...">
2649     <sp:MustSupportRefKeyIdentifier ... /> ?
2650     <sp:MustSupportRefIssuerSerial ... /> ?
2651     <sp:MustSupportRefExternalURI ... /> ?
2652     <sp:MustSupportRefEmbeddedToken ... /> ?
2653     <sp:MustSupportRefThumbprint ... /> ?
2654     <sp:MustSupportRefEncryptedKey ... /> ?
2655     <sp:RequireSignatureConfirmation ... /> ?
2656     ...
2657   </wsp:Policy>
2658 </sp:Wss11>
  
```

2659  
 2660 The following describes the attributes and elements listed in the schema outlined above:

2661 /sp:Wss11  
 2662 This identifies an WSS11 assertion.  
 2663 /sp:Wss11/wsp:Policy  
 2664 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.  
 2665 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier  
 2666 This OPTIONAL element is a policy assertion indicates that the [Key Identifier References]  
 2667 property is set to 'true'.  
 2668 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial  
 2669 This OPTIONAL element is a policy assertion indicates that the [Issuer Serial References]  
 2670 property is set to 'true'.

2671 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI  
2672 This OPTIONAL element is a policy assertion indicates that the [External URI References]  
2673 property is set to 'true'.

2674 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken  
2675 This OPTIONAL element is a policy assertion indicates that the [Embedded Token References]  
2676 property is set to 'true'.

2677 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint  
2678 This OPTIONAL element is a policy assertion indicates that the [Thumbprint References] property  
2679 is set to 'true'.

2680 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey  
2681 This OPTIONAL element is a policy assertion indicates that the [EncryptedKey References]  
2682 property is set to 'true'.

2683 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation  
2684 This OPTIONAL element is a policy assertion indicates that the [Signature Confirmation] property  
2685 is set to 'true'.

2686

## 10 WS-Trust Options

2687 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically  
2688 with client and server challenges and entropy behaviors. These assertions relate to interactions with a  
2689 Security Token Service and MAY augment the behaviors defined by the Binding Property Assertions  
2690 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2691

### 2692 **WS-Trust 1.3 Properties**

#### 2693 **[Client Challenge]**

2694 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a  
2695 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of  
2696 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of  
2697 messages exchanged by the client and service in satisfying the RST. This property has a default value of  
2698 'false'.

2699

#### 2700 **[Server Challenge]**

2701 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a  
2702 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of  
2703 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server MAY  
2704 increase the number of messages exchanged by the client and service in order to accommodate the  
2705 `wst:SignChallengeResponse` element sent by the client to the server in response to the  
2706 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the  
2707 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2708

#### 2709 **[Client Entropy]**

2710 This boolean property indicates whether client entropy is REQUIRED to be used as key material for a  
2711 requested proof token. A value of 'true' indicates that client entropy is REQUIRED. A value of 'false'  
2712 indicates that client entropy is NOT REQUIRED. This property has a default value of 'false'.

2713

#### 2714 **[Server Entropy]**

2715 This boolean property indicates whether server entropy is REQUIRED to be used as key material for a  
2716 requested proof token. A value of 'true' indicates that server entropy is REQUIRED. A value of 'false'  
2717 indicates that server entropy is NOT REQUIRED. This property has a default value of 'false'.

2718 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy  
2719 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm  
2720 Suite] property.

2721

#### 2722 **[Issued Tokens]**

2723 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in  
2724 WS-Trust. A value of 'true' indicates that the `wst:IssuedTokens` header is supported. A value of 'false'  
2725 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of  
2726 'false'.

#### 2727 **[Collection]**

2728 This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A  
2729 value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and  
2730 MUST be integrity protected either by transport or message level security. A value of 'false' indicates that  
2731 the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default  
2732 value of 'false'.  
2733

## 2734 10.1 Trust13 Assertion

2735 The Trust13 assertion allows you to specify which WS-Trust 1.3 options are supported.

### 2736 Syntax

```
2737 <sp:Trust13 xmlns:sp="..." ... >  
2738   <wsp:Policy xmlns:wsp="...">  
2739     <sp:MustSupportClientChallenge ... />?  
2740     <sp:MustSupportServerChallenge ... />?  
2741     <sp:RequireClientEntropy ... />?  
2742     <sp:RequireServerEntropy ... />?  
2743     <sp:MustSupportIssuedTokens ... />?  
2744     <sp:RequireRequestSecurityTokenCollection />?  
2745     <sp:RequireAppliesTo />?  
2746     ...  
2747   </wsp:Policy>  
2748   ...  
2749 </sp:Trust13 ... >
```

2750  
2751 The following describes the attributes and elements listed in the schema outlined above:

2752 /sp:Trust13

2753       This identifies a Trust13 assertion.

2754 /sp:Trust13/wsp:Policy

2755       This indicates a policy that controls WS-Trust 1.3 options.

2756 /sp:Trust13/wsp:Policy/sp:MustSupportClientChallenge

2757       This OPTIONAL element is a policy assertion indicates that the [Client Challenge] property is set  
2758 to 'true'.

2759 /sp:Trust13/wsp:Policy/sp:MustSupportServerChallenge

2760       This OPTIONAL element is a policy assertion indicates that the [Server Challenge] property is set  
2761 to 'true'.

2762 /sp:Trust13/wsp:Policy/sp:RequireClientEntropy

2763       This OPTIONAL element is a policy assertion indicates that the [Client Entropy] property is set to  
2764 'true'.

2765 /sp:Trust13/wsp:Policy/sp:RequireServerEntropy

2766       This OPTIONAL element is a policy assertion indicates that the [Server Entropy] property is set to  
2767 'true'.

2768 /sp:Trust13/wsp:Policy/sp:MustSupportIssuedTokens

2769       This OPTIONAL element is a policy assertion indicates that the [Issued Tokens] property is set to  
2770 'true'.

2771 /sp:Trust13/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2772       This OPTIONAL element is a policy assertion that indicates that the [Collection] property is set to  
2773 'true'.

2774 /sp:Trust13/wsp:Policy/sp:RequireAppliesTo

2775 This OPTIONAL element is a policy assertion that indicates that the STS requires the requestor  
2776 to specify the scope for the issued token using wsp:AppliesTo in the RST.

---

2777 **11 Guidance on creating new assertions and assertion**  
2778 **extensibility**

2779 This non-normative appendix provides guidance for designers of new assertions intended for use with this  
2780 specification.

2781 **11.1 General Design Points**

- 2782
- Prefer Distinct QNames
  - 2783 • Parameterize using nested policy where possible.
  - 2784 • Parameterize using attributes and/or child elements where necessary.

2785 **11.2 Detailed Design Guidance**

2786 Assertions in WS-SP are XML elements that are identified by their QName. Matching of assertions per  
2787 WS-Policy is performed by matching element QNames. Matching does not take into account attributes  
2788 that are present on the assertion element. Nor does it take into account child elements except for  
2789 `wsp:Policy` elements. If a `wsp:Policy` element is present, then matching occurs against the assertions  
2790 nested inside that `wsp:Policy` element recursively (see [Policy Assertion Nesting \[WS-Policy\]](#)).

2791  
2792 When designing new assertions for use with WS-SP, the above matching behaviour needs to be taken  
2793 into account. In general, multiple assertions with distinct QNames are preferably to a single assertion that  
2794 uses attributes and/or content to distinguish different cases. For example, given two possible assertion  
2795 designs;

2796

```
2797 Design 1  
2798  
2799     <A1/>  
2800     <A2/>  
2801     <A3/>  
2802  
2803 Design 2.  
2804  
2805     <A Parameter='1' />  
2806     <A Parameter='2' />  
2807     <A Parameter='3' />  
2808
```

2809 then design 1. would generally be preferred because it allows the policy matching logic to provide more  
2810 accurate matches between policies.

2811  
2812 A good example of design 1 is the token assertions defined in Section 5. The section defines 10 distinct  
2813 token assertions, rather than a single `sp:Token` assertion with, for example, a `TokenType` attribute. These  
2814 distinct token assertions make policy matching much more useful as less false positives are generated  
2815 when performing policy matching.

2816  
2817 There are cases where using attributes or child elements as parameters in assertion design is  
2818 reasonable. Examples include cases when implementations are expected to understand all the values for  
2819 a given parameter and when encoding the parameter information into the assertion QName would result  
2820 in an unmanageable number of assertions. A good example is the `sp:IncludeToken` attribute that appears

2821 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken  
2822 attribute and implementations are expected to understand the meaning of all 5 values. If this information  
2823 was encoded into the assertion QNames, each existing token assertion would require five variants, one  
2824 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2825

2826 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For  
2827 example, the token version assertions defined in Section 5 use such an approach. The overall token type  
2828 assertion is parameterized by the nested token version assertions. Policy matching can use these  
2829 parameters to find matches between policies where the broad token type is support by both parties but  
2830 they might not support the same specific versions.

2831

2832 Note, when designing assertions for new token types such assertions SHOULD allow the  
2833 sp:IncludeToken attribute and SHOULD allow nested policy.

2834



---

2835 **12 Security Considerations**

2836 It is strongly recommended that policies and assertions be signed to prevent tampering.

2837 It is recommended that policies should not be accepted unless they are signed and have an associated  
2838 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely  
2839 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that  
2840 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2841  
2842 It should be noted that the mechanisms described in this document could be secured as part of a SOAP  
2843 message using WSS: SOAP Message Security [[WSS10](#), [WSS11](#)] or embedded within other objects using  
2844 object-specific security mechanisms.

2845  
2846 It is recommended that policies not specify two (or more) SignedSupportingTokens or  
2847 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are  
2848 subject to modification which may be undetectable.

2849  
2850 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest  
2851 of the policy in order to combat certain XML substitution attacks.

---

## 2852 **Appendix A. Assertions and WS-PolicyAttachment**

2853 This non-normative appendix classifies assertions according to their suggested scope in WSDL 1.1 per  
2854 Section 4 of [WS-PolicyAttachment]. See Figure 1 in Section 4.1 of [WS-PolicyAttachment] for a graphical  
2855 representation of the relationship between policy scope and WSDL. Unless otherwise noted above, any  
2856 assertion that is listed under multiple [Policy Subjects] below MUST only apply to only one [Policy  
2857 Subject] in a WSDL 1.1 hierarchy for calculating an Effective Policy.

### 2858 **A.1 Endpoint Policy Subject Assertions**

#### 2859 **A.1.1 Security Binding Assertions**

- 2860 [TransportBinding Assertion](#) (Section 7.3)
- 2861 [SymmetricBinding Assertion](#) (Section 7.4)
- 2862 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2863 **A.1.2 Token Assertions**

- 2864 [SupportingTokens Assertion](#) (Section 8.1)
- 2865 [SignedSupportingTokens Assertion](#) (Section 8.2)
- 2866 [EndorsingSupportingTokens Assertion](#) (Section 8.3)
- 2867 [SignedEndorsingSupportingTokens Assertion](#) (Section 8.4)
- 2868 [SignedEncryptedSupportingTokens Assertion](#) (Section 8.5)
- 2869 [EndorsingEncryptedSupportingTokens Assertion](#) (Section 8.6)
- 2870 [SignedEndorsingEncryptedSupportingTokens Assertion](#) (Section 8.7)

#### 2871 **A.1.3 WSS: SOAP Message Security 1.0 Assertions**

- 2872 [Wss10 Assertion](#) (Section 9.1)

#### 2873 **A.1.4 WSS: SOAP Message Security 1.1 Assertions**

- 2874 [Wss11 Assertion](#) (Section 9.2)

#### 2875 **A.1.5 Trust 1.0 Assertions**

- 2876 [Trust13 Assertion](#) (Section 10.1)

### 2877 **A.2 Operation Policy Subject Assertions**

#### 2878 **A.2.1 Security Binding Assertions**

- 2879 [SymmetricBinding Assertion](#) (Section 7.4)
- 2880 [AsymmetricBinding Assertion](#) (Section 7.5)

#### 2881 **A.2.2 Supporting Token Assertions**

- 2882 [SupportingTokens Assertion](#) (Section 8.1)
- 2883 [SignedSupportingTokens Assertion](#) (Section 8.2)

2884	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2885	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2886	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2887	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2888	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

## 2889 **A.3 Message Policy Subject Assertions**

### 2890 **A.3.1 Supporting Token Assertions**

2891	<a href="#">SupportingTokens Assertion</a>	(Section 8.1)
2892	<a href="#">SignedSupportingTokens Assertion</a>	(Section 8.2)
2893	<a href="#">EndorsingSupportingTokens Assertion</a>	(Section 8.3)
2894	<a href="#">SignedEndorsingSupportingTokens Assertion</a>	(Section 8.4)
2895	<a href="#">SignedEncryptedSupportingTokens Assertion</a>	(Section 8.5)
2896	<a href="#">EndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.6)
2897	<a href="#">SignedEndorsingEncryptedSupportingTokens Assertion</a>	(Section 8.7)

### 2898 **A.3.2 Protection Assertions**

2899	<a href="#">SignedParts Assertion</a>	(Section 4.1.1)
2900	<a href="#">SignedElements Assertion</a>	(Section 4.1.2)
2901	<a href="#">EncryptedParts Assertion</a>	(Section 4.2.1)
2902	<a href="#">EncryptedElements Assertion</a>	(Section 4.2.2)
2903	<a href="#">ContentEncryptedElements Assertion</a>	(Section 4.2.3)
2904	<a href="#">RequiredElements Assertion</a>	(Section 4.3.1)
2905	<a href="#">RequiredParts Assertion</a>	(Section 4.3.2)

## 2906 **A.4 Assertions With Undefined Policy Subject**

2907 The assertions listed in this section do not have a defined policy subject because they appear nested  
 2908 inside some other assertion which does have a defined policy subject. This list is derived from nested  
 2909 assertions in the specification that have independent sections. It is not a complete list of nested  
 2910 assertions. Many of the assertions previously listed in this appendix as well as the ones below have  
 2911 additional nested assertions.

### 2912 **A.4.1 General Assertions**

2913	<a href="#">AlgorithmSuite Assertion</a>	(Section 7.1)
2914	<a href="#">Layout Assertion</a>	(Section 7.2)

### 2915 **A.4.2 Token Usage Assertions**

2916 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)  
 2917 assertions.

### 2918 **A.4.3 Token Assertions**

2919	<a href="#">UsernameToken Assertion</a>	(Section 5.3.1)
------	---	-----------------

2920	<a href="#">IssuedToken Assertion</a>	(Section 5.3.2)
2921	<a href="#">X509Token Assertion</a>	(Section 5.3.3)
2922	<a href="#">KerberosToken Assertion</a>	(Section 5.3.4)
2923	<a href="#">SpnegoContextToken Assertion</a>	(Section 5.3.5)
2924	<a href="#">SecurityContextToken Assertion</a>	(Section 5.3.6)
2925	<a href="#">SecureConversationToken Assertion</a>	(Section 5.3.7)
2926	<a href="#">SamlToken Assertion</a>	(Section 5.3.8)
2927	<a href="#">RelToken Assertion</a>	(Section 5.3.9)
2928	<a href="#">HttpsToken Assertion</a>	(Section 5.3.10)

## 2929 Appendix B. Issued Token Policy

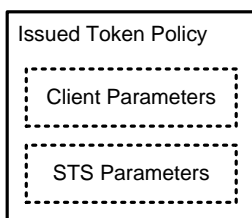
2930 The section provides further detail about behavior associated with the IssuedToken assertion in section  
2931 5.3.2.

2932

2933 The issued token security model involves a three-party setup. There's a target Server, a Client, and a  
2934 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from  
2935 STS to Client. Policy MAY be embedded inside an Issued Token assertion, or acquired out-of-band.  
2936 There MAY be an explicit trust relationship between the Server and the STS. There MUST be a trust  
2937 relationship between the Client and the STS.

2938

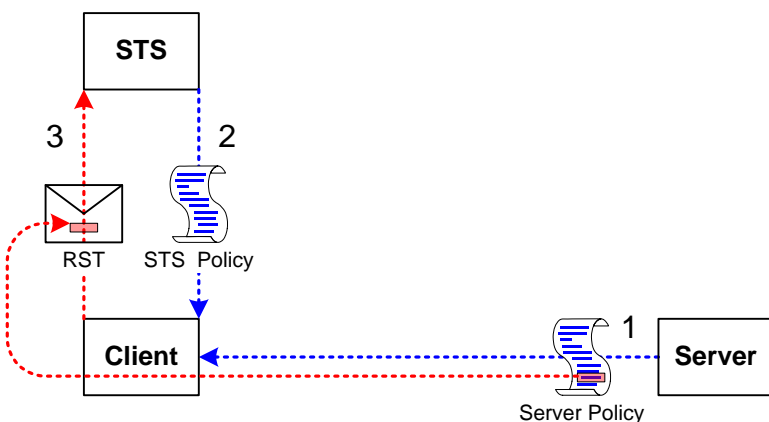
2939 The Issued Token policy assertion includes two parts: 1) client-specific parameters that MUST be  
2940 understood and processed by the client and 2) STS specific parameters which are to be processed by the  
2941 STS. The format of the Issued Token policy assertion is illustrated in the figure below.



2942

2943 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server  
2944 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are  
2945 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the  
2946 RST request sent by the Client to the STS as illustrated in the figure below.

2947



2948

2949 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to  
2950 formulate the RST request and will include any security-specific requirements of the STS.

2951

2952 The Client MAY augment or replace the contents of the RST made to the STS based on the Client-  
2953 specific parameters received from the Issued Token policy assertion contained in the Server policy, from  
2954 policy it received for the STS, or any other local parameters.

2955

2956 The Issued Token Policy Assertion contains elements which MUST be understood by the Client. The  
2957 assertion contains one element which contains a list of arbitrary elements which SHOULD be sent along  
2958 to the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST  
2959 request sent by the Client to the STS following the protocol defined in WS-Trust.

2960

2961 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)  
2962 [Trust\]](#). All items are OPTIONAL, since the Server and STS may already have a pre-arranged relationship  
2963 which specifies some or all of the conditions and constraints for issued tokens.

2964

## Appendix C. Strict Security Header Layout Examples

2965 The following sections describe the security header layout for specific bindings when applying the 'Strict'  
2966 layout rules defined in Section 6.7.

### 2967 C.1 Transport Binding

2968 This section describes how the 'Strict' security header layout rules apply to the Transport Binding.

#### 2969 C.1.1 Policy

2970 The following example shows a policy indicating a Transport Binding, an Https Token as the Transport  
2971 Token, an algorithm suite, a requirement to include tokens in the supporting signatures, a username  
2972 token attached to the message, and finally an X509 token attached to the message and endorsing the  
2973 message signature. No message protection requirements are described since the transport covers all  
2974 message parts.

```
2975 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
2976   <sp:TransportBinding>
2977     <wsp:Policy>
2978       <sp:TransportToken>
2979         <wsp:Policy>
2980           <sp:HttpsToken />
2981         </wsp:Policy>
2982       </sp:TransportToken>
2983       <sp:AlgorithmSuite>
2984         <wsp:Policy>
2985           <sp:Basic256 />
2986         </wsp:Policy>
2987       </sp:AlgorithmSuite>
2988       <sp:Layout>
2989         <wsp:Policy>
2990           <sp:Strict />
2991         </wsp:Policy>
2992       </sp:Layout>
2993       <sp:IncludeTimestamp />
2994     </wsp:Policy>
2995   </sp:TransportBinding>
2996   <sp:SignedSupportingTokens>
2997     <wsp:Policy>
2998       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2999     </wsp:Policy>
3000   </sp:SignedSupportingTokens>
3001   <sp:SignedEndorsingSupportingTokens>
3002     <wsp:Policy>
3003       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3004         <wsp:Policy>
3005           <sp:WssX509v3Token10 />
3006         </wsp:Policy>
3007       </sp:X509Token>
3008     </wsp:Policy>
3009   </sp:SignedEndorsingSupportingTokens>
3010   <sp:Wss11>
3011     <sp:RequireSignatureConfirmation />
3012   </sp:Wss11>
3013 </wsp:Policy>
```

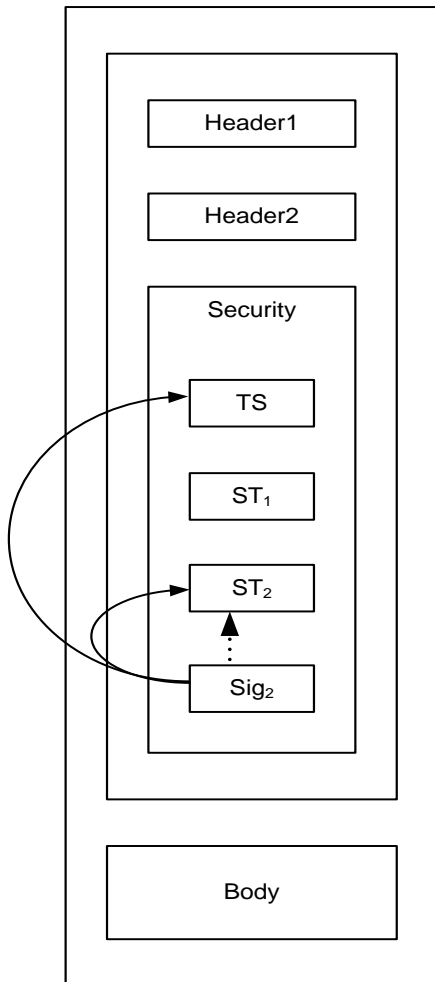
3014 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3015 header layout for this binding.

3016 **C.1.2 Initiator to Recipient Messages**

3017 Messages sent from initiator to recipient have the following layout for the security header:

- 3018 1. A `wsu:Timestamp` element.
- 3019 2. Any tokens contained in the [Signed Supporting Tokens] property.
- 3020 3. Any tokens contained in the [Signed Endorsing Supporting Tokens] property each followed by the  
3021 corresponding signature. Each signature **MUST** cover the `wsu:Timestamp` element from 1  
3022 above and **SHOULD** cover any other unique identifier for the message in order to prevent  
3023 replays. If [Token Protection] is 'true', the signature **MUST** also cover the supporting token. If  
3024 [Derived Keys] is 'true' and the supporting token is associated with a symmetric key, then a  
3025 Derived Key Token, based on the supporting token, appears between the supporting token and  
3026 the signature.
- 3027 4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each  
3028 signature **MUST** cover the `wsu:Timestamp` element from 1 above and **SHOULD** cover at least  
3029 some other unique identifier for the message in order to prevent replays. If [Token Protection] is  
3030 'true', the signature **MUST** also cover the supporting token. If [Derived Keys] is 'true' and the  
3031 supporting token is associated with a symmetric key, then a Derived Key Token, based on the  
3032 supporting token, appears before the signature.

3033 The following diagram illustrates the security header layout for the initiator to recipient message:



3034



3035 The outer box shows that the entire message is protected (signed and encrypted) by the transport. The  
3036 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>,  
3037 namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST<sub>2</sub>.  
3038 The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering  
3039 of the items in the security header follows the most optimal layout for a receiver to process its contents.

3040 *Example:*

3041 Initiator to recipient message

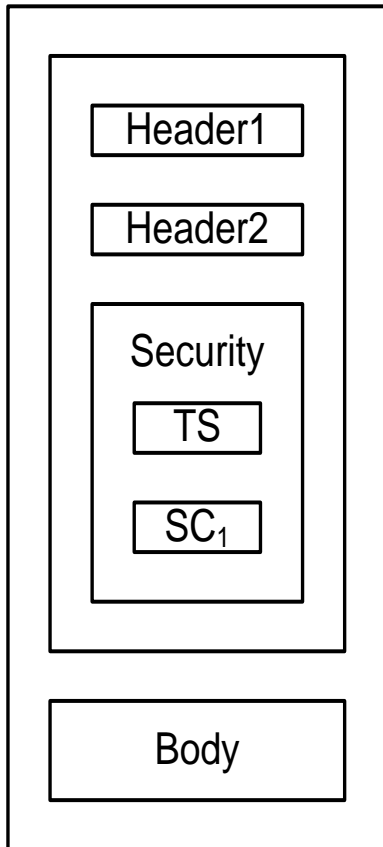
```
3042 <S:Envelope xmlns:S="..." xmlns:wss="..." xmlns:wsu="..." xmlns:ds="...">  
3043   <S:Header>  
3044     ...  
3045     <wsse:Security>  
3046       <wsu:Timestamp wsu:Id="timestamp">  
3047         <wsu:Created>[datetime]</wsu:Created>  
3048         <wsu:Expires>[datetime]</wsu:Expires>  
3049       </wsu:Timestamp>  
3050       <wsse:UsernameToken wsu:Id='SomeSignedToken' >  
3051         ...  
3052       </wsse:UsernameToken>  
3053       <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >  
3054         ...  
3055       </wsse:BinarySecurityToken>  
3056       <ds:Signature>  
3057         <ds:SignedInfo>  
3058           <ds:References>  
3059             <ds:Reference URI="#timestamp" />  
3060             <ds:Reference URI="#SomeSignedEndorsingToken" />  
3061           </ds:References>  
3062         </ds:SignedInfo>  
3063         <ds:SignatureValue>...</ds:SignatureValue>  
3064         <ds:KeyInfo>  
3065           <wsse:SecurityTokenReference>  
3066             <wsse:Reference URI="#SomeSignedEndorsingToken" />  
3067           </wsse:SecurityTokenReference>  
3068         </ds:KeyInfo>  
3069       </ds:Signature>  
3070     ...  
3071   </wsse:Security>  
3072   ...  
3073 </S:Header>  
3074 <S:Body>  
3075   ...  
3076 </S:Body>  
3077 </S:Envelope>
```

### 3078 **C.1.3 Recipient to Initiator Messages**

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

- 3080 1. A `wsu:Timestamp` element.
- 3081 2. If the [Signature Confirmation] property has a value of 'true', then a  
3082 `wsse11:SignatureConfirmation` element for each signature in the corresponding message  
3083 sent from initiator to recipient. If there are no signatures in the corresponding message from the  
3084 initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no `Value`  
3085 attribute.

3086 The following diagram illustrates the security header layout for the recipient to initiator message:



3087

3088 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One  
 3089 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial  
 3090 message illustrated previously is included. In general, the ordering of the items in the security header  
 3091 follows the most optimal layout for a receiver to process its contents.

3092 *Example:*

3093 Recipient to initiator message

```

3094 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3095   <S:Header>
3096     ...
3097     <wsse:Security>
3098       <wsu:Timestamp wsu:Id="timestamp">
3099         <wsu:Created>[datetime]</wsu:Created>
3100         <wsu:Expires>[datetime]</wsu:Expires>
3101       </wsu:Timestamp>
3102       <wsse11:SignatureConfirmation Value="..." />
3103     ...
3104   </wsse:Security>
3105   ...
3106 </S:Header>
3107 <S:Body>
3108   ...
3109 </S:Body>
3110 </S:Envelope>
  
```

## 3111 C.2 Symmetric Binding

3112 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

## 3113 C.2.1 Policy

3114 The following example shows a policy indicating a Symmetric Binding, a symmetric key based  
3115 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message  
3116 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in  
3117 the message signature and the supporting signatures, a username token attached to the message, and  
3118 finally an X509 token attached to the message and endorsing the message signature. Minimum message  
3119 protection requirements are described as well.

```
3120 <!-- Example Endpoint Policy -->
3121 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3122   <sp:SymmetricBinding>
3123     <wsp:Policy>
3124       <sp:ProtectionToken>
3125         <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3126           <sp:Issuer>...</sp:Issuer>
3127           <sp:RequestSecurityTokenTemplate>
3128             ...
3129           </sp:RequestSecurityTokenTemplate>
3130         </sp:IssuedToken>
3131       </sp:ProtectionToken>
3132       <sp:AlgorithmSuite>
3133         <wsp:Policy>
3134           <sp:Basic256 />
3135         </wsp:Policy>
3136       </sp:AlgorithmSuite>
3137       <sp:Layout>
3138         <wsp:Policy>
3139           <sp:Strict />
3140         </wsp:Policy>
3141       </sp:Layout>
3142       <sp:IncludeTimestamp />
3143       <sp:EncryptBeforeSigning />
3144       <sp:EncryptSignature />
3145       <sp:ProtectTokens />
3146     </wsp:Policy>
3147   </sp:SymmetricBinding>
3148   <sp:SignedSupportingTokens>
3149     <wsp:Policy>
3150       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3151     </wsp:Policy>
3152   </sp:SignedSupportingTokens>
3153   <sp:SignedEndorsingSupportingTokens>
3154     <wsp:Policy>
3155       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3156         <wsp:Policy>
3157           <sp:WssX509v3Token10 />
3158         </wsp:Policy>
3159       </sp:X509Token>
3160     </wsp:Policy>
3161   </sp:SignedEndorsingSupportingTokens>
3162   <sp:Wss11>
3163     <wsp:Policy>
3164       <sp:RequireSignatureConfirmation />
3165     </wsp:Policy>
3166   </sp:Wss11>
3167 </wsp:Policy>
3168
```

```

3169
3170 <!-- Example Message Policy -->
3171 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3172   <sp:SignedParts>
3173     <sp:Header Name="Header1" Namespace="..." />
3174     <sp:Header Name="Header2" Namespace="..." />
3175     <sp:Body/>
3176   </sp:SignedParts>
3177   <sp:EncryptedParts>
3178     <sp:Header Name="Header2" Namespace="..." />
3179     <sp:Body/>
3180   </sp:EncryptedParts>
3181 </wsp:Policy>

```

3182 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3183 header layout for this binding.

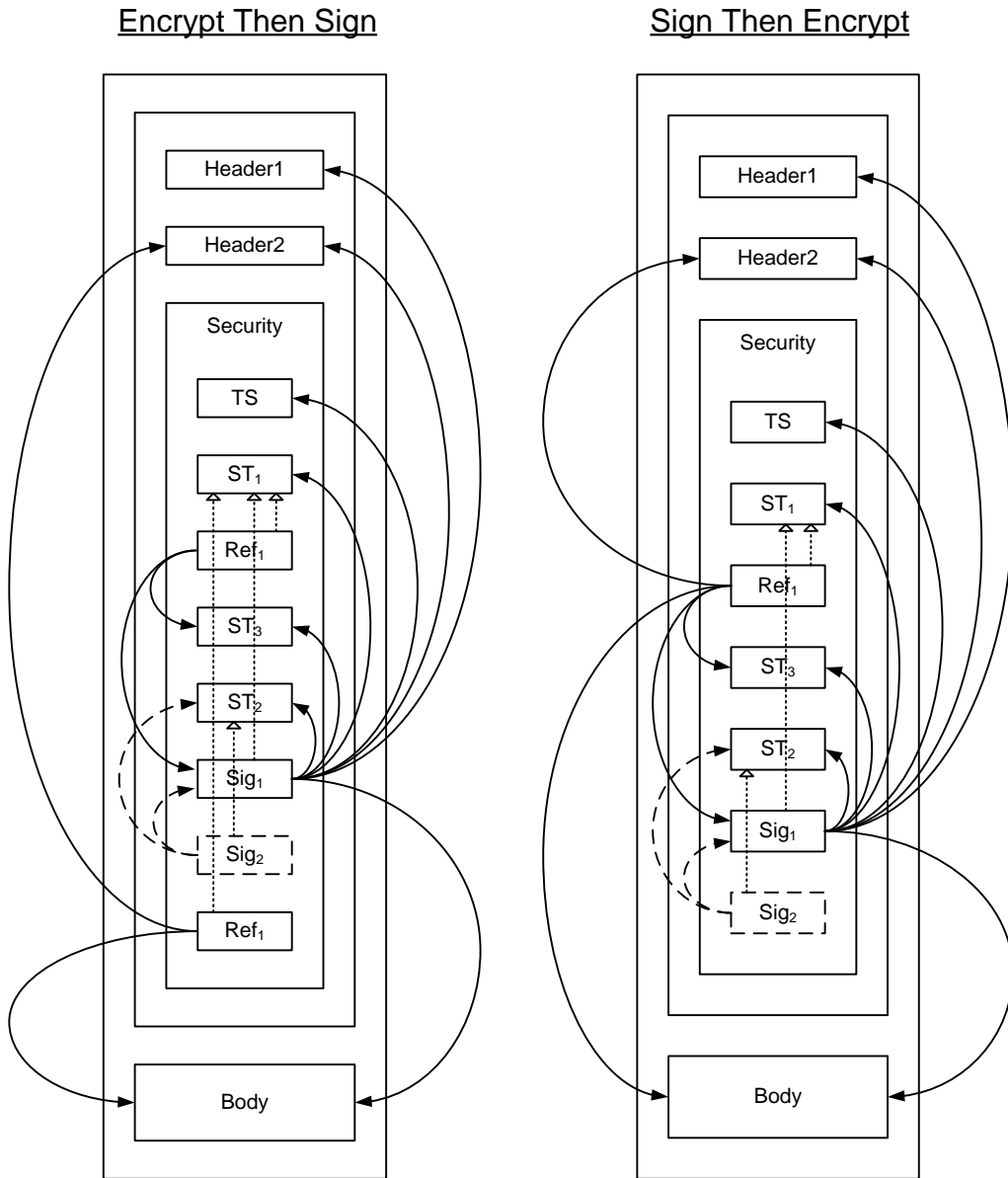
## 3184 C.2.2 Initiator to Recipient Messages

3185 Messages sent from initiator to recipient have the following layout for the security header:

- 3186 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3187 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or  
3188 `.../IncludeToken/Always`, then the [Encryption Token].
- 3189 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3190 Derived Key Token is used for encryption.
- 3191 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3192 reference list MUST include a reference to the message signature. If [Protection Order] is  
3193 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3194 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3195 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3196 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]  
3197 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or  
3198 `.../IncludeToken/Always`.
- 3199 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3200 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the  
3201 [Signature Token].
- 3202 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This  
3203 Derived Key Token is used for signature.
- 3204 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of  
3205 whether they are included in the message, and any message parts specified in SignedParts  
3206 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature  
3207 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in  
3208 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3209 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing  
3210 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]  
3211 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the  
3212 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the  
3213 endorsing token, appears before the signature.
- 3214 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message  
3215 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
3216 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3217 above.

3218

3219 The following diagram illustrates the security header layout for the initiator to recipient message:



3220

3221 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3222 The dashed arrows on the left from the box labeled Sig<sub>2</sub> indicate the parts signed by the supporting token  
 3223 labeled ST<sub>2</sub>, namely the message signature labeled Sig<sub>1</sub> and the token used as the basis for the  
 3224 signature labeled ST<sub>2</sub>. The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts  
 3225 encrypted using a key based on the Shared Secret Token labeled ST<sub>1</sub>. The dotted arrows inside the box  
 3226 labeled Security indicate the token that was used as the basis for each cryptographic operation. In  
 3227 general, the ordering of the items in the security header follows the most optimal layout for a receiver to  
 3228 process its contents.

3229 *Example:*

3230 Initiator to recipient message using EncryptBeforeSigning:

```
3231 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3232   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3233   xmlns:xenc="..." xmlns:ds="...">
3234   <S:Header>
3235     <x:Header1 wsu:Id="Header1" >
3236       ...
3237     </x:Header1>
3238
```

```

3239 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3240 <!-- Plaintext Header2
3241 <x:Header2 wsu:Id="Header2" >
3242 ...
3243 </x:Header2>
3244 -->
3245 ...
3246 </wsse1:EncryptedHeader>
3247 ...
3248 <wsse:Security>
3249 <wsu:Timestamp wsu:Id="Timestamp">
3250 <wsu:Created>...</wsu:Created>
3251 <wsu:Expires>...</wsu:Expires>
3252 </wsu:Timestamp>
3253 <saml:Assertion AssertionId="_SharedSecretToken" ...>
3254 ...
3255 </saml:Assertion>
3256 <xenc:ReferenceList>
3257 <xenc:DataReference URI="#enc_Signature" />
3258 <xenc:DataReference URI="#enc_SomeUsernameToken" />
3259 ...
3260 </xenc:ReferenceList>
3261 <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3262 <!-- Plaintext UsernameToken
3263 <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3264 ...
3265 </wsse:UsernameToken>
3266 -->
3267 ...
3268 <ds:KeyInfo>
3269 <wsse:SecurityTokenReference>
3270 <wsse:Reference URI="#_SharedSecretToken" />
3271 </wsse:SecurityTokenReference>
3272 </ds:KeyInfo>
3273 </xenc:EncryptedData>
3274 <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3275 ...
3276 </wsse:BinarySecurityToken>
3277 <xenc:EncryptedData ID="enc_Signature">
3278 <!-- Plaintext Signature
3279 <ds:Signature Id="Signature">
3280 <ds:SignedInfo>
3281 <ds:References>
3282 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3283 <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3284 <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3285 <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3286 <ds:Reference URI="#Header1" >...</ds:Reference>
3287 <ds:Reference URI="#Header2" >...</ds:Reference>
3288 <ds:Reference URI="#Body" >...</ds:Reference>
3289 </ds:References>
3290 </ds:SignedInfo>
3291 <ds:SignatureValue>...</ds:SignatureValue>
3292 <ds:KeyInfo>
3293 <wsse:SecurityTokenReference>
3294 <wsse:Reference URI="#_SharedSecretToken" />
3295 </wsse:SecurityTokenReference>
3296 </ds:KeyInfo>
3297 </ds:Signature>
3298 -->
3299 ...
3300 <ds:KeyInfo>
3301 <wsse:SecurityTokenReference>
3302 <wsse:Reference URI="#_SharedSecretToken" />

```

```

3303     </wsse:SecurityTokenReference>
3304   </ds:KeyInfo>
3305 </xenc:EncryptedData>
3306 <ds:Signature>
3307   <ds:SignedInfo>
3308     <ds:References>
3309       <ds:Reference URI="#Signature" >...</ds:Reference>
3310       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3311     </ds:References>
3312   </ds:SignedInfo>
3313 <ds:SignatureValue>...</ds:SignatureValue>
3314 <ds:KeyInfo>
3315   <wsse:SecurityTokenReference>
3316     <wsse:Reference URI="#SomeSupportingToken" />
3317   </wsse:SecurityTokenReference>
3318 </ds:KeyInfo>
3319 </ds:Signature>
3320 <xenc:ReferenceList>
3321   <xenc:DataReference URI="#enc_Body" />
3322   <xenc:DataReference URI="#enc_Header2" />
3323   ...
3324 </xenc:ReferenceList>
3325 </wsse:Security>
3326 </S:Header>
3327 <S:Body wsu:Id="Body">
3328   <xenc:EncryptedData Id="enc_Body">
3329     ...
3330     <ds:KeyInfo>
3331       <wsse:SecurityTokenReference>
3332         <wsse:Reference URI="#_SharedSecretToken" />
3333       </wsse:SecurityTokenReference>
3334     </ds:KeyInfo>
3335   </xenc:EncryptedData>
3336 </S:Body>
3337 </S:Envelope>

```

### 3338 C.2.3 Recipient to Initiator Messages

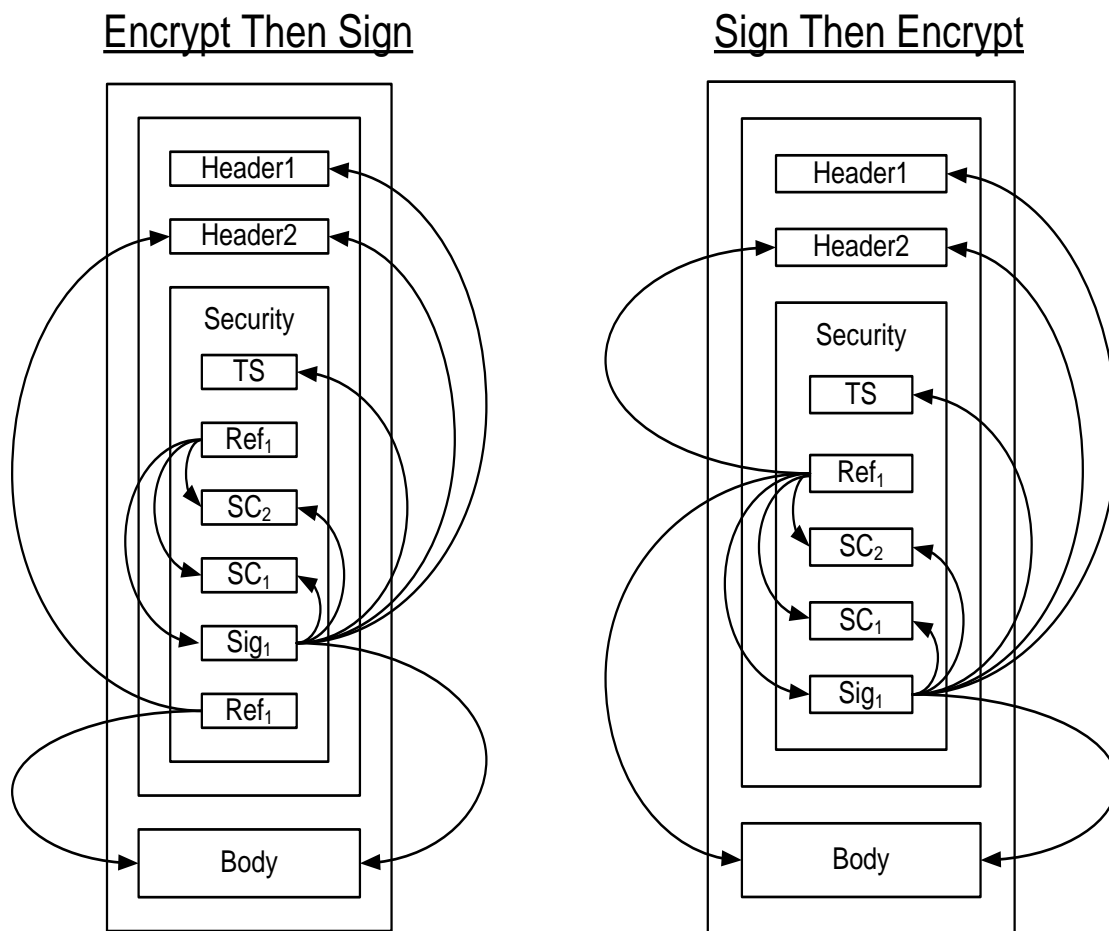
3339 Messages send from recipient to initiator have the following layout for the security header:

- 3340 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3341 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Always`, then the  
3342 [Encryption Token].
- 3343 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This  
3344 Derived Key Token is used for encryption.
- 3345 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the  
3346 reference list MUST include a reference to the message signature from 6 below, and the  
3347 `wss11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is  
3348 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts  
3349 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in  
3350 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2  
3351 above.
- 3352 5. If [Signature Confirmation] is 'true' then a `wss11:SignatureConfirmation` element for each  
3353 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3354 in the corresponding message from the initiator to the recipient, then a  
3355 `wss11:SignatureConfirmation` element with no Value attribute.
- 3356 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`  
3357 attribute on the [Signature Token] is `.../IncludeToken/Always`, then the [Signature Token].



- 3358 7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This  
 3359 Derived Key Token is used for signature.
- 3360 8. A signature over the wsu:Timestamp from 1 above, any wssell:SignatureConfirmation  
 3361 elements from 5 above, and all the message parts specified in SignedParts assertions in the  
 3362 policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]  
 3363 regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token  
 3364 from 6 above MUST be used, otherwise the key in the [Signature Token].
- 3365 9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message  
 3366 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key  
 3367 in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption  
 3368 Token].

3369 The following diagram illustrates the security header layout for the recipient to initiator message:



3370

3371 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>1</sub>.  
 3372 The arrows on the left from boxes labeled Ref<sub>1</sub> indicate references to parts encrypted using a key based  
 3373 on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two  
 3374 wssell:SignatureConfirmation elements labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures  
 3375 in the initial message illustrated previously is included. In general, the ordering of the items in the security  
 3376 header follows the most optimal layout for a receiver to process its contents. The rules used to determine  
 3377 this ordering are described in Appendix C.

3378 *Example:*

3379 Recipient to initiator message using EncryptBeforeSigning:

```
3380 <S:Envelope>
3381   <S:Header>
3382     <x:Header1 wsu:Id="Header1" >
3383       ...
3384     </x:Header1>
3385     <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3386       <!-- Plaintext Header2
3387         <x:Header2 wsu:Id="Header2" >
3388           ...
3389         </x:Header2>
3390       -->
3391       ...
3392     </wsse11:EncryptedHeader>
3393     ...
3394   <wsse:Security>
3395     <wsu:Timestamp wsu:Id="Timestamp">
3396       <wsu:Created>...</wsu:Created>
3397       <wsu:Expires>...</wsu:Expires>
3398     </wsu:Timestamp>
3399     <xenc:ReferenceList>
3400       <xenc:DataReference URI="#enc_Signature" />
3401       <xenc:DataReference URI="#enc_SigConf1" />
3402       <xenc:DataReference URI="#enc_SigConf2" />
3403       ...
3404     </xenc:ReferenceList>
3405     <xenc:EncryptedData ID="enc_SigConf1" >
3406       <!-- Plaintext SignatureConfirmation
3407         <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3408           ...
3409         </wsse11:SignatureConfirmation>
3410       -->
3411       ...
3412     </xenc:EncryptedData>
3413     <xenc:EncryptedData ID="enc_SigConf2" >
3414       <!-- Plaintext SignatureConfirmation
3415         <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3416           ...
3417         </wsse11:SignatureConfirmation>
3418       -->
3419       ...
3420     </xenc:EncryptedData>
```

3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469

```
<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>
```

### 3470 C.3 Asymmetric Binding

3471 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

#### 3472 C.3.1 Policy

3473 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator  
3474 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the  
3475 message parts before signing, a requirement to encrypt the message signature, a requirement to include  
3476 tokens in the message signature and the supporting signatures, a requirement to include  
3477 `wsse11:SignatureConfirmation` elements, a username token attached to the message, and finally

3478 an X509 token attached to the message and endorsing the message signature. Minimum message  
3479 protection requirements are described as well.

```
3480 <!-- Example Endpoint Policy -->  
3481 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
3482   <sp:AsymmetricBinding>  
3483     <wsp:Policy>  
3484       <sp:RecipientToken>  
3485         <wsp:Policy>  
3486           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3487         </wsp:Policy>  
3488       </sp:RecipientToken>  
3489       <sp:InitiatorToken>  
3490         <wsp:Policy>  
3491           <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />  
3492         </wsp:Policy>  
3493       </sp:InitiatorToken>  
3494       <sp:AlgorithmSuite>  
3495         <wsp:Policy>  
3496           <sp:Basic256 />  
3497         </wsp:Policy>  
3498       </sp:AlgorithmSuite>  
3499       <sp:Layout>  
3500         <wsp:Policy>  
3501           <sp:Strict />  
3502         </wsp:Policy>  
3503       </sp:Layout>  
3504       <sp:IncludeTimestamp />  
3505       <sp:EncryptBeforeSigning />  
3506       <sp:EncryptSignature />  
3507       <sp:ProtectTokens />  
3508     </wsp:Policy>  
3509   </sp:AsymmetricBinding>  
3510   <sp:SignedEncryptedSupportingTokens>  
3511     <wsp:Policy>  
3512       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />  
3513     </wsp:Policy>  
3514   </sp:SignedEncryptedSupportingTokens>  
3515   <sp:SignedEndorsingSupportingTokens>  
3516     <wsp:Policy>  
3517       <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">  
3518         <wsp:Policy>  
3519           <sp:WssX509v3Token10 />  
3520         </wsp:Policy>  
3521       </sp:X509Token>  
3522     </wsp:Policy>  
3523   </sp:SignedEndorsingSupportingTokens>  
3524   <sp:Wss11>  
3525     <wsp:Policy>  
3526       <sp:RequireSignatureConfirmation />  
3527     </wsp:Policy>  
3528   </sp:Wss11>  
3529 </wsp:Policy>  
3530
```

3531

```
3532 <!-- Example Message Policy -->
3533 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3534   <sp:SignedParts>
3535     <sp:Header Name="Header1" Namespace="..." />
3536     <sp:Header Name="Header2" Namespace="..." />
3537     <sp:Body/>
3538   </sp:SignedParts>
3539   <sp:EncryptedParts>
3540     <sp:Header Name="Header2" Namespace="..." />
3541     <sp:Body/>
3542   </sp:EncryptedParts>
3543 </wsp:All>
```

3544  
3545 This policy is used as the basis for the examples shown in the subsequent section describing the security  
3546 header layout for this binding.

### 3547 **C.3.2 Initiator to Recipient Messages**

3548 Messages sent from initiator to recipient have the following layout:

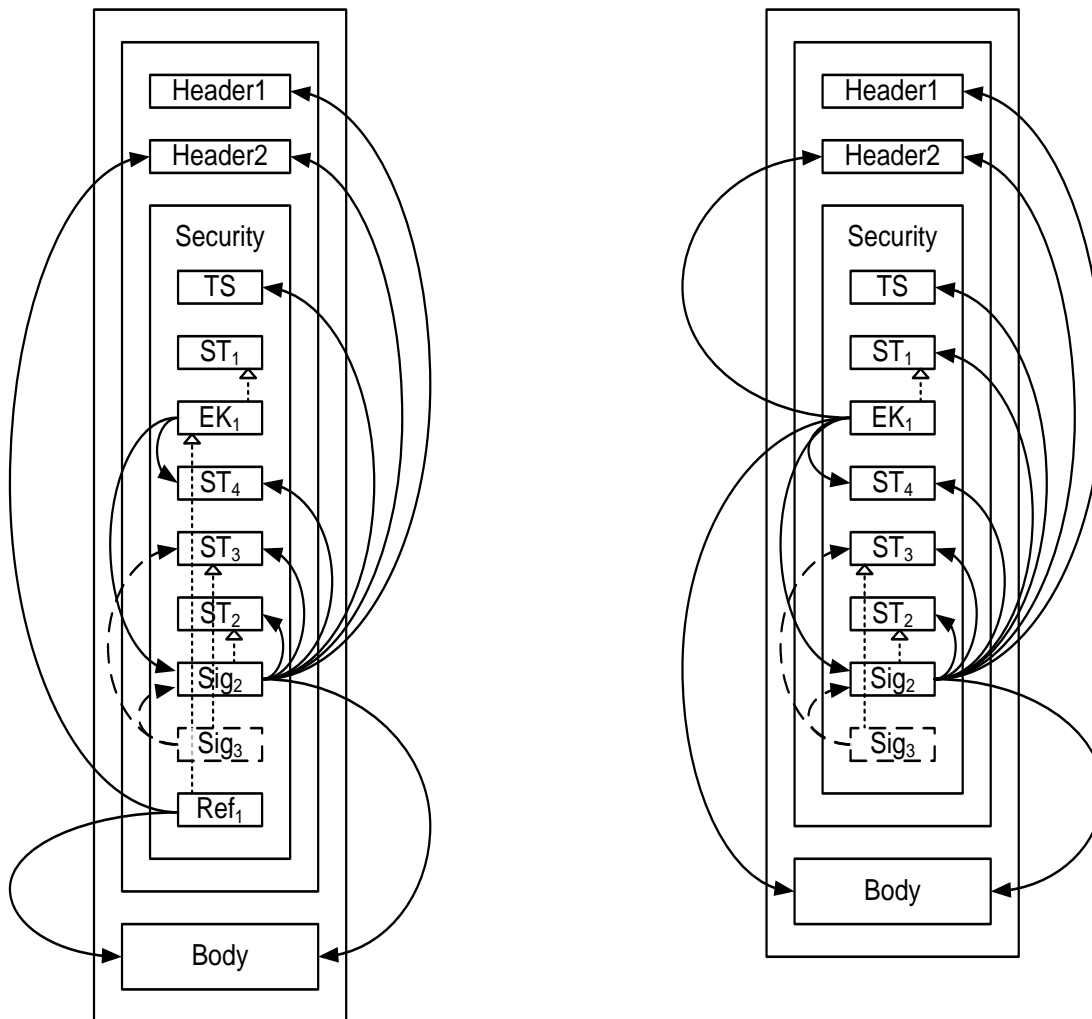
- 3549 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3550 2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3551 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Recipient Token].
- 3552 3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3553 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3554 the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3555 reference to all the message parts specified in `EncryptedParts` assertions in the policy. If  
3556 [Signature Protection] is 'true' then the reference list MUST contain a reference to the message  
3557 signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message  
3558 signature.
- 3559 4. Any tokens from the supporting tokens properties (as defined in section 8) whose  
3560 `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
- 3561 5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3562 `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the [Initiator Token].
- 3563 6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from  
3564 1 above, any tokens from 4 above regardless of whether they are included in the message, and  
3565 any message parts specified in `SignedParts` assertions in the policy. If [Token Protection] is 'true',  
3566 the signature MUST also cover the [Initiator Token] regardless of whether it is included in the  
3567 message.
- 3568 7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting  
3569 Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a  
3570 symmetric key, then a Derived Key Token, based on the supporting token, appears before the  
3571 signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token  
3572 regardless of whether it is included in the message.
- 3573 8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
3574 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
3575 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
3576 reference list includes a reference to all the message parts specified in `EncryptedParts` assertions  
3577 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
3578 element from 3 above.

3579

3580 The following diagram illustrates the security header layout for the initiator to recipient messages:

### Encrypt Then Sign

### Sign Then Encrypt



3581  
 3582 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3583 using the [Initiator Token] labeled ST<sub>2</sub>. The dashed arrows on the left from the box labeled Sig<sub>3</sub> indicate  
 3584 the parts signed by the supporting token ST<sub>3</sub>, namely the message signature Sig<sub>2</sub> and the token used as  
 3585 the basis for the signature labeled ST<sub>3</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate references  
 3586 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on the left  
 3587 from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained in the  
 3588 encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token used as  
 3589 the basis for each cryptographic operation. In general, the ordering of the items in the security header  
 3590 follows the most optimal layout for a receiver to process its contents. The rules used to determine this  
 3591 ordering are described in Appendix C.

3592  
 3593 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted  
 3594 key contains an external reference to the token containing the encryption key. The diagram illustrates  
 3595 how one might attach a security token related to the encrypted key for completeness. One possible use-

3596 case for this approach might be a stack which does not support the STR Dereferencing Transform, but  
3597 wishes to include the encryption token in the message signature.

3598 Initiator to recipient message *Example*

3599 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3600     xmlns:wssell1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3601 <S:Header>
3602   <x:Header1 wsu:Id="Header1" >
3603     ...
3604   </x:Header1>
3605   <wssell1:EncryptedHeader wsu:Id="enc_Header2">
3606     <!-- Plaintext Header2
3607     <x:Header2 wsu:Id="Header2" >
3608       ...
3609     </x:Header2>
3610     -->
3611     ...
3612   </wssell1:EncryptedHeader>
3613   ...
3614   <wsse:Security>
3615     <wsu:Timestamp wsu:Id="Timestamp">
3616       <wsu:Created>...</wsu:Created>
3617       <wsu:Expires>...</wsu:Expires>
3618     </wsu:Timestamp>
3619     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3620       ...
3621     </wsse:BinarySecurityToken>
3622     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3623       ...
3624       <xenc:ReferenceList>
3625         <xenc:DataReference URI="#enc_Signature" />
3626         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3627         ...
3628       </xenc:ReferenceList>
3629     </xenc:EncryptedKey>
3630     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3631       <!-- Plaintext UsernameToken
3632       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3633         ...
3634       </wsse:UsernameToken>
3635       -->
3636       ...
3637     </xenc:EncryptedData>
3638     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3639       ...
3640     </wsse:BinarySecurityToken>
3641     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3642       ...
3643     </wsse:BinarySecurityToken>
3644     <xenc:EncryptedData ID="enc_Signature">
3645       <!-- Plaintext Signature
3646       <ds:Signature Id="Signature">
3647         <ds:SignedInfo>
3648           <ds:References>
3649             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3650             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3651             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3652             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3653             <ds:Reference URI="#Header1" >...</ds:Reference>
3654             <ds:Reference URI="#Header2" >...</ds:Reference>
3655             <ds:Reference URI="#Body" >...</ds:Reference>
3656           </ds:References>
3657         </ds:SignedInfo>
3658         <ds:SignatureValue>...</ds:SignatureValue>
3659         <ds:KeyInfo>
3660           <wsse:SecurityTokenReference>
3661             <wsse:Reference URI="#InitiatorToken" />
3662           </wsse:SecurityTokenReference>
3663         </ds:KeyInfo>

```



```

3664     </ds:Signature>
3665     -->
3666     ...
3667 </xenc:EncryptedData>
3668 <ds:Signature>
3669   <ds:SignedInfo>
3670     <ds:References>
3671       <ds:Reference URI="#Signature" >...</ds:Reference>
3672       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3673     </ds:References>
3674   </ds:SignedInfo>
3675   <ds:SignatureValue>...</ds:SignatureValue>
3676   <ds:KeyInfo>
3677     <wsse:SecurityTokenReference>
3678       <wsse:Reference URI="#SomeSupportingToken" />
3679     </wsse:SecurityTokenReference>
3680   </ds:KeyInfo>
3681 </ds:Signature>
3682 <xenc:ReferenceList>
3683   <xenc:DataReference URI="#enc_Body" />
3684   <xenc:DataReference URI="#enc_Header2" />
3685   ...
3686 </xenc:ReferenceList>
3687 </wsse:Security>
3688 </S:Header>
3689 <S:Body wsu:Id="Body">
3690   <xenc:EncryptedData Id="enc_Body">
3691     ...
3692     <ds:KeyInfo>
3693       <wsse:SecurityTokenReference>
3694         <wsse:Reference URI="#RecipientEncryptedKey" />
3695       </wsse:SecurityTokenReference>
3696     </ds:KeyInfo>
3697   </xenc:EncryptedData>
3698 </S:Body>
3699 </S:Envelope>

```

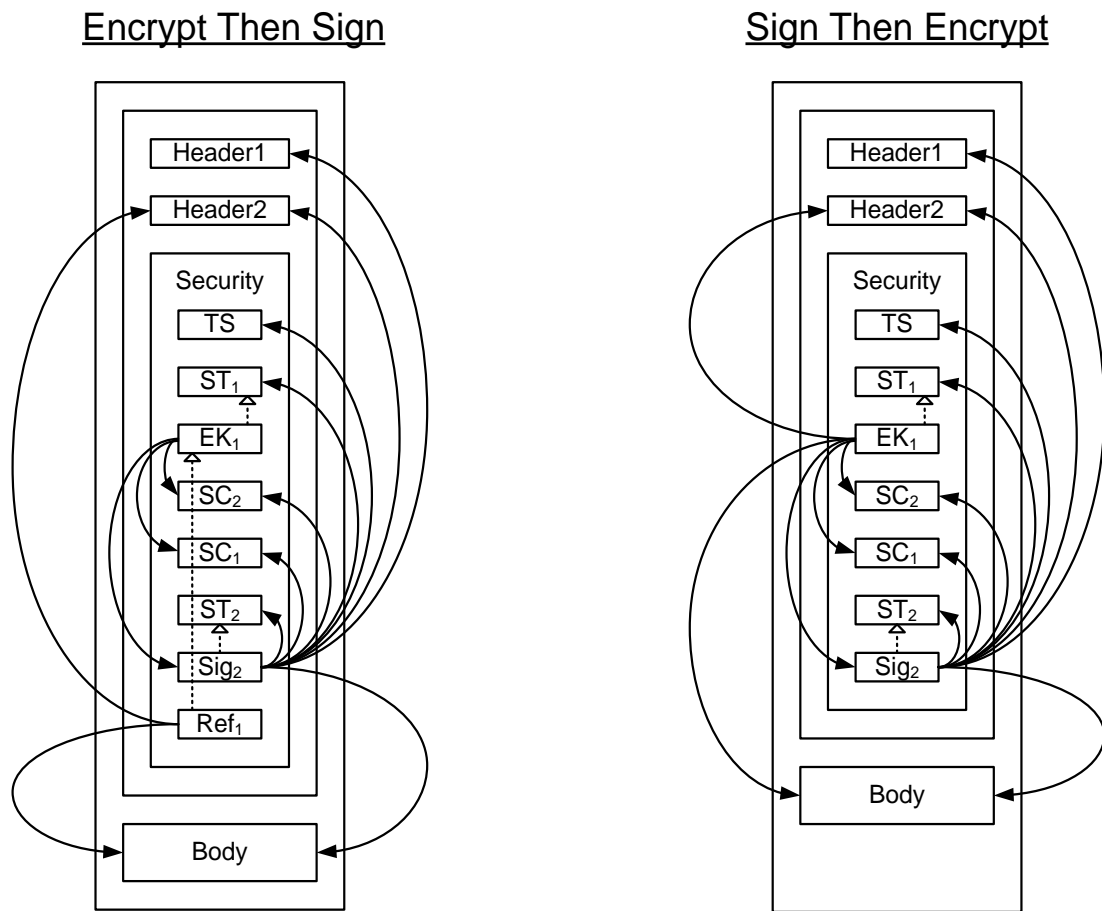
### 3700 C.3.3 Recipient to Initiator Messages

3701 Messages sent from recipient to initiator have the following layout:

- 3702 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3703 2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is  
3704 `.../IncludeToken/Always`, then the [Initiator Token].
- 3705 3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or  
3706 [SignatureProtection] is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for  
3707 the initiator. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a  
3708 reference to all the message parts specified in EncryptedParts assertions in the policy. If  
3709 [Signature Protection] is 'true' then the reference list MUST also contain a reference to the  
3710 message signature from 6 below, if any and references to the  
3711 `wss11:SignatureConfirmation` elements from 4 below, if any.
- 3712 4. If [Signature Confirmation] is 'true', then a `wss11:SignatureConfirmation` element for each  
3713 signature in the corresponding message sent from initiator to recipient. If there are no signatures  
3714 in the corresponding message from the initiator to the recipient, then a  
3715 `wss11:SignatureConfirmation` element with no Value attribute.
- 3716 5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is  
3717 `.../IncludeToken/Always`, then the [Recipient Token].

- 3718 6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],  
 3719 over the `wsu:Timestamp` from 1 above, the `wssell:SignatureConfirmation` elements  
 3720 from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token  
 3721 Protection] is 'true' then the signature MUST also cover the [Recipient Token].
- 3722 7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if  
 3723 [Signature Protection] is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted  
 3724 for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The  
 3725 reference list includes a reference to all the message parts specified in EncryptedParts assertions  
 3726 in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey`  
 3727 element from 3 above.

3728  
 3729 The following diagram illustrates the security header layout for the recipient to initiator messages:



3730

3731 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig<sub>2</sub>  
 3732 using the [Recipient Token] labeled ST<sub>2</sub>. The arrows on the left from boxes labeled EK<sub>1</sub> indicate  
 3733 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST<sub>1</sub>. The arrows on  
 3734 the left from boxes labeled Ref<sub>1</sub> indicate additional references to parts encrypted using the key contained  
 3735 in the encrypted key labeled EK<sub>1</sub>. The dotted arrows inside the box labeled Security indicate the token  
 3736 used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements  
 3737 labeled SC<sub>1</sub> and SC<sub>2</sub> corresponding to the two signatures in the initial message illustrated previously is  
 3738 included. In general, the ordering of the items in the security header follows the most optimal layout for a  
 3739 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

3740 Recipient to initiator message *Example*:

```

3741 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3742   xmlns:wssell="..." xmlns:wsse="..."
3743   xmlns:xenc="..." xmlns:ds="...">
3744 <S:Header>
3745   <x:Header1 wsu:Id="Header1" >
3746     ...
3747   </x:Header1>
3748   <wssell:EncryptedHeader wsu:Id="enc_Header2">
3749     <!-- Plaintext Header2
3750     <x:Header2 wsu:Id="Header2" >
3751       ...
3752     </x:Header2>
3753     -->
3754     ...
3755   </wssell:EncryptedHeader>
3756   ...
3757   <wsse:Security>
3758     <wsu:Timestamp wsu:Id="Timestamp">
3759       <wsu:Created>...</wsu:Created>
3760       <wsu:Expires>...</wsu:Expires>
3761     </wsu:Timestamp>
3762     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3763       ...
3764     </wsse:BinarySecurityToken>
3765     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3766       ...
3767       <xenc:ReferenceList>
3768         <xenc:DataReference URI="#enc_Signature" />
3769         <xenc:DataReference URI="#enc_SigConf1" />
3770         <xenc:DataReference URI="#enc_SigConf2" />
3771         ...
3772       </xenc:ReferenceList>
3773     </xenc:EncryptedKey>
3774     <xenc:EncryptedData ID="enc_SigConf2" >
3775       <!-- Plaintext SignatureConfirmation
3776       <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3777         ...
3778       </wssell:SignatureConfirmation>
3779       -->
3780       ...
3781     </xenc:EncryptedData>
3782     <xenc:EncryptedData ID="enc_SigConf1" >
3783       <!-- Plaintext SignatureConfirmation
3784       <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3785         ...
3786       </wssell:SignatureConfirmation>
3787       -->
3788       ...
3789     </xenc:EncryptedData>
3790     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3791       ...
3792   </wsse:BinarySecurityToken>
3793

```

```

3794 <xenc:EncryptedData ID="enc_Signature">
3795   <!-- Plaintext Signature
3796   <ds:Signature Id="Signature">
3797     <ds:SignedInfo>
3798       <ds:References>
3799         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3800         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3801         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3802         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3803         <ds:Reference URI="#Header1" >...</ds:Reference>
3804         <ds:Reference URI="#Header2" >...</ds:Reference>
3805         <ds:Reference URI="#Body" >...</ds:Reference>
3806       </ds:References>
3807     </ds:SignedInfo>
3808     <ds:SignatureValue>...</ds:SignatureValue>
3809     <ds:KeyInfo>
3810       <wsse:SecurityTokenReference>
3811         <wsse:Reference URI="#RecipientToken" />
3812       </wsse:SecurityTokenReference>
3813     </ds:KeyInfo>
3814   </ds:Signature>
3815   -->
3816   ...
3817 </xenc:EncryptedData>
3818 <xenc:ReferenceList>
3819   <xenc:DataReference URI="#enc_Body" />
3820   <xenc:DataReference URI="#enc_Header2" />
3821   ...
3822 </xenc:ReferenceList>
3823 </wsse:Security>
3824 </S:Header>
3825 <S:Body wsu:Id="Body">
3826   <xenc:EncryptedData Id="enc_Body">
3827     ...
3828     <ds:KeyInfo>
3829       <wsse:SecurityTokenReference>
3830         <wsse:Reference URI="#InitiatorEncryptedKey" />
3831       </wsse:SecurityTokenReference>
3832     </ds:KeyInfo>
3833   </xenc:EncryptedData>
3834 </S:Body>
3835 </S:Envelope>

```

3836  
3837

---

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

3838 This section lists the criteria for when various child elements of the Security header are signed and/or  
3839 encrypted at the message level including whether they are signed by the message signature or a  
3840 supporting signature. It assumes that there are no `sp:SignedElements` and no  
3841 `sp:EncryptedElements` assertions in the policy. If such assertions are present in the policy then  
3842 additional child elements of the security header might be signed and/or encrypted.

### 3843 D.1 Elements signed by the message signature

- 3844 1. The `wsu:Timestamp` element (Section 6.2).  
3845 2. All `wssell:SignatureConfirmation` elements (Section 9).  
3846 3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token],  
3847 [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption  
3848 Token] when [Token Protection] has a value of 'true' (Section 6.5).  
3849 4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed  
3850 Endorsing Supporting Tokens] (Section 8.5).

### 3851 D.2 Elements signed by all endorsing signatures

- 3852 1. The `ds:Signature` element that forms the message signature (Section 8.3).  
3853 2. The `wsu:Timestamp` element in the case of a transport binding (Section 8.3).

### 3854 D.3 Elements signed by a specific endorsing signature

- 3855 1. Security Tokens corresponding to [Endorsing Supporting Tokens] or [Signed Endorsing  
3856 Supporting Tokens] when [Token Protection] has a value of 'true' (Section 8.8).

### 3857 D.4 Elements that are encrypted

- 3858 1. The `ds:Signature` element that forms the message signature when [Signature Protection]  
3859 has a value of 'true' (Section 6.4).  
3860 2. All `wssell:SignatureConfirmation` elements when [Signature Protection] has a value  
3861 of 'true' (Section 6.4).  
3862 3. A `wsse:UsernameToken` MAY be encrypted when a transport binding is not being used  
3863 (Section 5.3.1).  
3864

---

3865 **Appendix E. Acknowledgements**

3866 The following individuals have participated in the creation of this specification and are gratefully  
3867 acknowledged:

3868 **Original Authors of the initial contribution:**

3869 Giovanni Della-Libera, Microsoft  
3870 Martin Gudgin, Microsoft  
3871 Phillip Hallam-Baker, VeriSign  
3872 Maryann Hondo, IBM  
3873 Hans Granqvist, Verisign  
3874 Chris Kaler, Microsoft (editor)  
3875 Hiroshi Maruyama, IBM  
3876 Michael McIntosh, IBM  
3877 Anthony Nadalin, IBM (editor)  
3878 Nataraj Nagaratnam, IBM  
3879 Rob Philpott, RSA Security  
3880 Hemma Prafullchandra, VeriSign  
3881 John Shewchuk, Microsoft  
3882 Doug Walter, Microsoft  
3883 Riaz Zolfonoon, RSA Security  
3884

3885 **Original Acknowledgements of the initial contribution:**

3886 Vaithialingam B. Balayoghan, Microsoft  
3887 Francisco Curbera, IBM  
3888 Christopher Ferris, IBM  
3889 Cédric Fournet, Microsoft  
3890 Andy Gordon, Microsoft  
3891 Tomasz Janczuk, Microsoft  
3892 David Melgar, IBM  
3893 Mike Perks, IBM  
3894 Bruce Rich, IBM  
3895 Jeffrey Schlimmer, Microsoft  
3896 Chris Sharp, IBM  
3897 Kent Tamura, IBM  
3898 T.R. Vishwanath, Microsoft  
3899 Elliot Waingold, Microsoft  
3900

3901 **TC Members during the development of this specification:**

3902 Don Adams, Tibco Software Inc.  
3903 Jan Alexander, Microsoft Corporation  
3904 Steve Anderson, BMC Software  
3905 Donal Arundel, IONA Technologies  
3906 Howard Bae, Oracle Corporation  
3907 Abbie Barbir, Nortel Networks Limited  
3908 Charlton Barreto, Adobe Systems  
3909 Mighael Botha, Software AG, Inc.  
3910 Toufic Boubez, Layer 7 Technologies Inc.  
3911 Norman Brickman, Mitre Corporation  
3912 Melissa Brumfield, Booz Allen Hamilton

3913 Lloyd Burch, Novell  
3914 Scott Cantor, Internet2  
3915 Greg Carpenter, Microsoft Corporation  
3916 Steve Carter, Novell  
3917 Symon Chang, BEA Systems, Inc.  
3918 Ching-Yun (C.Y.) Chao, IBM  
3919 Martin Chapman, Oracle Corporation  
3920 Kate Cherry, Lockheed Martin  
3921 Henry (Hyenvui) Chung, IBM  
3922 Luc Clement, Systinet Corp.  
3923 Paul Cotton, Microsoft Corporation  
3924 Glen Daniels, Sonic Software Corp.  
3925 Peter Davis, Neustar, Inc.  
3926 Martijn de Boer, SAP AG  
3927 Werner Dittmann, Siemens AG  
3928 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
3929 Fred Dushin, IONA Technologies  
3930 Petr Dvorak, Systinet Corp.  
3931 Colleen Evans, Microsoft Corporation  
3932 Ruchith Fernando, WSO2  
3933 Mark Fussell, Microsoft Corporation  
3934 Vijay Gajjala, Microsoft Corporation  
3935 Marc Goodner, Microsoft Corporation  
3936 Hans Granqvist, VeriSign  
3937 Martin Gudgin, Microsoft Corporation  
3938 Tony Gullotta, SOA Software Inc.  
3939 Jiandong Guo, Sun Microsystems  
3940 Phillip Hallam-Baker, VeriSign  
3941 Patrick Harding, Ping Identity Corporation  
3942 Heather Hinton, IBM  
3943 Frederick Hirsch, Nokia Corporation  
3944 Jeff Hodges, Neustar, Inc.  
3945 Will Hopkins, BEA Systems, Inc.  
3946 Alex Hristov, Otecia Incorporated  
3947 John Hughes, PA Consulting  
3948 Diane Jordan, IBM  
3949 Venugopal K, Sun Microsystems  
3950 Chris Kaler, Microsoft Corporation  
3951 Dana Kaufman, Forum Systems, Inc.  
3952 Paul Knight, Nortel Networks Limited  
3953 Ramanathan Krishnamurthy, IONA Technologies  
3954 Christopher Kurt, Microsoft Corporation  
3955 Kelvin Lawrence, IBM  
3956 Hubert Le Van Gong, Sun Microsystems  
3957 Jong Lee, BEA Systems, Inc.  
3958 Rich Levinson, Oracle Corporation  
3959 Tommy Lindberg, Dajeil Ltd.  
3960 Mark Little, JBoss Inc.  
3961 Hal Lockhart, BEA Systems, Inc.  
3962 Mike Lyons, Layer 7 Technologies Inc.  
3963 Eve Maler, Sun Microsystems  
3964 Ashok Malhotra, Oracle Corporation  
3965 Anand Mani, CrimsonLogic Pte Ltd  
3966 Jonathan Marsh, Microsoft Corporation  
3967 Robin Martherus, Oracle Corporation  
3968 Miko Matsumura, Infravio, Inc.  
3969 Gary McAfee, IBM

3970 Michael McIntosh, IBM  
3971 John Merrells, Sxip Networks SRL  
3972 Jeff Mischkinsky, Oracle Corporation  
3973 Prateek Mishra, Oracle Corporation  
3974 Bob Morgan, Internet2  
3975 Vamsi Motukuru, Oracle Corporation  
3976 Raajmohan Na, EDS  
3977 Anthony Nadalin, IBM  
3978 Andrew Nash, Reactivity, Inc.  
3979 Eric Newcomer, IONA Technologies  
3980 Duane Nickull, Adobe Systems  
3981 Toshihiro Nishimura, Fujitsu Limited  
3982 Rob Philpott, RSA Security  
3983 Denis Pilipchuk, BEA Systems, Inc.  
3984 Darren Platt, Ping Identity Corporation  
3985 Martin Raepfle, SAP AG  
3986 Nick Ragouzis, Enosis Group LLC  
3987 Prakash Reddy, CA  
3988 Alain Regnier, Ricoh Company, Ltd.  
3989 Irving Reid, Hewlett-Packard  
3990 Bruce Rich, IBM  
3991 Tom Rutt, Fujitsu Limited  
3992 Maneesh Sahu, Actional Corporation  
3993 Frank Siebenlist, Argonne National Laboratory  
3994 Joe Smith, Apani Networks  
3995 Davanum Srinivas, WSO2  
3996 Yakov Sverdlov, CA  
3997 Gene Thurston, AmberPoint  
3998 Victor Valle, IBM  
3999 Asir Vedamuthu, Microsoft Corporation  
4000 Greg Whitehead, Hewlett-Packard  
4001 Ron Williams, IBM  
4002 Corinna Witt, BEA Systems, Inc.  
4003 Kyle Young, Microsoft Corporation