



WS-SecurityPolicy 1.2

OASIS Standard

1 July 2007

Specification URIs:

This Version:

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

Previous Version:

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

Latest Version:

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

Artifact Type:

specification

Technical Committee:

OASIS Web Services Secure Exchange TC

Chair(s):

Kelvin Lawrence, IBM
Chris Kaler, Microsoft

Editor(s):

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

Related work:

N/A

Declared XML Namespace(s):

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

Abstract:

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

Status:

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

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

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

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

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

Notices

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

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

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

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

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

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

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

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

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

Table of Contents

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

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

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

1 Introduction

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

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

1.1 Example

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

Table 1: Example security policy.

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

```

(20) </sp:SignedParts>
(21) <sp:EncryptedParts>
(22)   <sp:Body/>
(23) </sp:EncryptedParts>
(24) </wsp:Policy>

```

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

1.2 Namespaces

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

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

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

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

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

| | | |
|-----|---|--------------------|
| wsa | http://www.w3.org/2005/08/addressing | [WS-Addressing] |
| sp | http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702 | This specification |

1.3 Schema Files

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

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

1.4 Terminology

Policy - A collection of policy alternatives.

Policy Alternative - A collection of policy assertions.

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

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

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

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

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

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

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

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

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

Supporting Token - A token used to provide additional claims.

1.4.1 Notational Conventions

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

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

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

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

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

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

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

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

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

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

1.5 Normative References

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

| | | |
|-----|-------------------------|---|
| 153 | | |
| 154 | [RFC2068] | IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997 |
| 155 | | |
| 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)", 9 May 2006. |
| 165 | | |
| 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 April 2006. |
| 169 | | |
| 170 | | http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/ |
| 171 | | W3C Candidate Recommendation "Web Services Policy 1.5 – Framework", 28 February 2007 |
| 172 | | |
| 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 April 2006. |
| 176 | | |
| 177 | | http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/ |
| 178 | | |
| 179 | | W3C Candidate Recommendation "Web Services Policy 1.5 – Attachment", 28 February 2007 |
| 180 | | |
| 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 2006 |
| 187 | | |
| 188 | | http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512 |
| 189 | | |
| 190 | [WSS10] | OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004. |
| 191 | | |
| 192 | | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf |
| 193 | | |
| 194 | | |
| 195 | [WSS11] | OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006. |
| 196 | | |
| 197 | | http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf |
| 198 | | |

| | | |
|-----|---------------------------|---|
| 199 | | |
| 200 | [WSS:UsernameToken1.0] | OASIS Standard, "Web Services Security: UsernameToken Profile", |
| 201 | | March 2004 |
| 202 | | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username- |
| 203 | | token-profile-1.0.pdf |
| 204 | | |
| 205 | [WSS:UsernameToken1.1] | OASIS Standard, "Web Services Security: UsernameToken Profile |
| 206 | | 1.1", February 2006 |
| 207 | | http://www.oasis-open.org/committees/download.php/16782/wss-v1.1- |
| 208 | | spec-os-UsernameTokenProfile.pdf |
| 209 | | |
| 210 | [WSS:X509Token1.0] | OASIS Standard, "Web Services Security X.509 Certificate Token |
| 211 | | Profile", March 2004 |
| 212 | | http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token- |
| 213 | | profile-1.0.pdf |
| 214 | | |
| 215 | [WSS:X509Token1.1] | OASIS Standard, "Web Services Security X.509 Certificate Token |
| 216 | | Profile", February 2006 |
| 217 | | http://www.oasis-open.org/committees/download.php/16785/wss-v1.1- |
| 218 | | spec-os-x509TokenProfile.pdf |
| 219 | | |
| 220 | [WSS:KerberosToken1.1] | OASIS Standard, "Web Services Security Kerberos Token Profile 1.1", |
| 221 | | February 2006 |
| 222 | | http://www.oasis-open.org/committees/download.php/16788/wss-v1.1- |
| 223 | | spec-os-KerberosTokenProfile.pdf |
| 224 | | |
| 225 | [WSS:SAMLTokenProfile1.0] | OASIS Standard, "Web Services Security: SAML Token Profile", |
| 226 | | December 2004 |
| 227 | | http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf |
| 228 | | |
| 229 | [WSS:SAMLTokenProfile1.1] | OASIS Standard, "Web Services Security: SAML Token Profile 1.1", |
| 230 | | February 2006 |
| 231 | | http://www.oasis-open.org/committees/download.php/16768/wss-v1.1- |
| 232 | | spec-os-SAMLTokenProfile.pdf |
| 233 | | |
| 234 | [WSS:RELTTokenProfile1.0] | OASIS Standard, "Web Services Security Rights Expression Language |
| 235 | | (REL) Token Profile", December 2004 |
| 236 | | http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf |
| 237 | | |
| 238 | [WSS:RELTTokenProfile1.1] | OASIS Standard, "Web Services Security Rights Expression Language |
| 239 | | (REL) Token Profile 1.1", February 2006 |
| 240 | | http://www.oasis-open.org/committees/download.php/16687/oasis- |
| 241 | | wss-rel-token-profile-1.1.pdf |
| 242 | | |
| 243 | [WSS:SwAPProfile1.1] | OASIS Standard, "Web Services Security SOAP Messages with |
| 244 | | Attachments (SwA) Profile 1.1", February 2006 |

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

268 1.6 Non-Normative References

269 None.
270

2 Security Policy Model

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

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

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

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

2.1 Security Assertion Model

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

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

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

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

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

2.2 Nested Policy Assertions

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

2.3 Security Binding Abstraction

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

A binding defines the following security characteristics:

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

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

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

- 357 • TransportBinding (Section 7.3)
- 358 • SymmetricBinding (Section 7.4)
- 359 • AsymmetricBinding (Section 7.5)

3 Policy Considerations

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

3.1 Nested Policy

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

3.2 Policy Subjects

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

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

[Message Policy Subject]

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

wsdl:message

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

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

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

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

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

[Operation Policy Subject]

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

wsdl:portType/wsdl:operation

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

wsdl:binding/wsdl:operation

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

[Endpoint Policy Subject]

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

wsdl:portType

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

401 wsdl:binding

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

404 wsdl:port

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

4 Protection Assertions

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

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

4.1 Integrity Assertions

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

4.1.1 SignedParts Assertion

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

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

Syntax

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

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

/sp:SignedParts

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

/sp:SignedParts/sp:Body

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

/sp:SignedParts/sp:Header

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

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

/sp:SignedParts/sp:Header/@Name

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

/sp:SignedParts/sp:Header/@Namespace

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

/sp:SignedParts/sp:Attachments

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

4.1.2 SignedElements Assertion

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

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

Syntax

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

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

/sp:SignedElements

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

/sp:SignedElements/@XPathVersion

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

/sp:SignedElements/sp:XPath

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

4.2 Confidentiality Assertions

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

4.2.1 EncryptedParts Assertion

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

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

Syntax

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

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

/sp:EncryptedParts

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

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

/sp:EncryptedParts/sp:Body

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

/sp:EncryptedParts/sp:Header

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

/sp:EncryptedParts/sp:Header/@Name

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

540 /sp:EncryptedParts/sp:Header/@Namespace

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

543 /sp:EncryptedParts/sp:Attachments

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

549 4.2.2 EncryptedElements Assertion

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

555

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

559 Syntax

```
560 <sp:EncryptedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
561   <sp:XPath>xs:string</sp:XPath>+
562   ...
563 </sp:EncryptedElements>
```

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

565 /sp:EncryptedElements

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

568 /sp:EncryptedElements/@XPathVersion

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

571 /sp:EncryptedElements/sp:XPath

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

576 4.2.3 ContentEncryptedElements Assertion

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

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

Syntax

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

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

/sp:ContentEncryptedElements

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

/sp:ContentEncryptedElements/@XPathVersion

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

/sp:ContentEncryptedElements/sp:XPath

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

4.3 Required Elements Assertion

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

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

4.3.1 RequiredElements Assertion

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

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

Syntax

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

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

/sp:RequiredElements

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

626 /sp:RequiredElements/@XPathVersion
627 This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is
628 provided, then XPath 1.0 is assumed.
629 /sp:RequiredElements/sp:XPath
630 This element contains a string specifying an XPath expression that identifies the header elements
631 that a message MUST contain. The XPath expression is evaluated against the
632 S:Envelope/S:Header element node of the message. Multiple instances of this element may
633 appear within this assertion and should be treated as a combined XPath expression.

634 4.3.2 RequiredParts Assertion

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

638
639 There MAY be multiple RequiredParts assertions present. Multiple RequiredParts assertions present
640 within a policy alternative are equivalent to a single RequiredParts assertion containing the union of all
641 specified Header elements.

642 Syntax

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

646
647 The following describes the attributes and elements listed in the schema outlined above:

648 /sp:RequiredParts/sp:Header
649 This assertion specifies the headers elements that MUST be present in the message.
650 /sp:RequiredParts/sp:Header/@Name
651 This required attribute indicates the local name of the SOAPHeader that needs to be present in
652 the message.
653 /sp:RequiredParts/sp:Header/@Namespace
654 This required attribute indicates the namespace of the SOAP header that needs to be present in
655 the message.

5 Token Assertions

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

5.1 Token Inclusion

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

5.1.1 Token Inclusion Values

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

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

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

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

5.1.2 Token Inclusion and Token References

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

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

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

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

5.2 Token Issuer and Required Claims

5.2.1 Token Issuer

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

5.2.2 Token Issuer Name

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

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

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

5.2.3 Required Claims

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

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

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

5.2.4 Processing Rules and Token Matching

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

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

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

5.3 Token Properties

5.3.1 [Derived Keys] Property

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

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

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

5.3.2 [Explicit Derived Keys] Property

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

5.3.3 [Implied Derived Keys] Property

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

5.4 Token Assertion Types

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

5.4.1 UsernameToken Assertion

This element represents a requirement to include a username token.

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

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

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

Syntax

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

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

/sp:UsernameToken

This identifies a UsernameToken assertion.

/sp:UsernameToken/@sp:IncludeToken

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

/sp:UsernameToken/sp:Issuer

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

/sp:UsernameToken/sp:IssuerName

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

/sp:UsernameToken/wst:Claims

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

/sp:UsernameToken/wsp:Policy

810 This required element identifies additional requirements for use of the sp:UsernameToken
811 assertion.

812 /sp:UsernameToken/wsp:Policy/sp:NoPassword

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

815 /sp:UsernameToken/wsp:Policy/sp:HashPassword

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

820 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

823 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

826 /sp:UsernameToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

830 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken10

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

833 /sp:UsernameToken/wsp:Policy/sp:WssUsernameToken11

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

836 5.4.2 IssuedToken Assertion

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

841 Syntax

```
842 <sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
843   (
844     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
845     <sp:IssuerName>xs:anyURI</sp:IssuerName>
846   ) ?
```

```

847 <wst:Claims Dialect="..."> ... </wst:Claims> ?
848 <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
849 ...
850 </sp:RequestSecurityTokenTemplate>
851 <wsp:Policy xmlns:wsp="...">
852 (
853   <sp:RequireDerivedKeys ... /> |
854   <sp:RequireImpliedDerivedKeys ... /> |
855   <sp:RequireExplicitDerivedKeys ... />
856 ) ?
857 <sp:RequireExternalReference ... /> ?
858 <sp:RequireInternalReference ... /> ?
859 ...
860 </wsp:Policy>
861 ...
862 </sp:IssuedToken>

```

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

864 /sp:IssuedToken

865 This identifies an IssuedToken assertion.

866 /sp:IssuedToken/@sp:IncludeToken

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

868 /sp:IssuedToken/sp:Issuer

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

871 /sp:IssuedToken/sp:IssuerName

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

873 /sp:IssuedToken/wst:Claims

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

876 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

881 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

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

884 /sp:IssuedToken/wsp:Policy

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

886 /sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys

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

889 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

892 /sp:IssuedToken/wsp:Policy/sp:RequireImpliedDerivedKeys

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

896 /sp:IssuedToken/wsp:Policy/sp:RequireInternalReference

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

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

900 /sp:IssuedToken/wsp:Policy/sp:RequireExternalReference

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

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

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

909 5.4.3 X509Token Assertion

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

911 Syntax

```
912 <sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >  
913   (  
914     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |  
915     <sp:IssuerName>xs:anyURI</sp:IssuerName>  
916   ) ?  
917   <wst:Claims Dialect="..."> ... </wst:Claims> ?  
918   <wsp:Policy xmlns:wsp="...">  
919     (  
920       <sp:RequireDerivedKeys ... /> |  
921       <sp:RequireExplicitDerivedKeys ... /> |  
922       <sp:RequireImpliedDerivedKeys ... />  
923     ) ?  
924     <sp:RequireKeyIdentifierReference ... /> ?  
925     <sp:RequireIssuerSerialReference ... /> ?  
926     <sp:RequireEmbeddedTokenReference ... /> ?  
927     <sp:RequireThumbprintReference ... /> ?  
928     (  
929       <sp:WssX509V3Token10 ... /> |  
930       <sp:WssX509Pkcs7Token10 ... /> |  
931       <sp:WssX509PkiPathV1Token10 ... /> |  
932       <sp:WssX509V1Token11 ... /> |  
933       <sp:WssX509V3Token11 ... /> |  
934       <sp:WssX509Pkcs7Token11 ... /> |  
935       <sp:WssX509PkiPathV1Token11 ... />  
936     ) ?  
937     ...  
938   </wsp:Policy>  
939   ...  
940 </sp:X509Token>
```

941

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

943 /sp:X509Token

944 This identifies an X509Token assertion.

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

947 /sp:X509Token/sp:Issuer
 948 This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of
 949 the `sp:X509Token`.

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

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

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

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

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

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

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

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

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

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

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

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

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

988 /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

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

991 /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

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

994 /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

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

997 /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

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

5.4.4 KerberosToken Assertion

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

Syntax

```
<sp:KerberosToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  (
    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
    <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
  <wst:Claims Dialect="..."> ... </wst:Claims> ?
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:RequireDerivedKeys ... /> |
      <sp:RequireImpliedDerivedKeys ... /> |
      <sp:RequireExplicitDerivedKeys ... />
    ) ?
    <sp:RequireKeyIdentifierReference ... /> ?
    (
      <sp:WssKerberosV5ApReqToken11 ... /> |
      <sp:WssGssKerberosV5ApReqToken11 ... />
    ) ?
    ...
  </wsp:Policy>
  ...
</sp:KerberosToken>
```

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

/sp:KerberosToken

This identifies a KerberosV5ApReqToken assertion.

/sp:KerberosToken/@sp:IncludeToken

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

/sp:KerberosToken/sp:Issuer

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

/sp:KerberosToken/sp:IssuerName

1035 This optional element, of type xs:anyURI, contains the logical name of the sp:KerberosToken
 1036 issuer.

1037 /sp:KerberosToken/wst:Claims

1038 This optional element identifies the required claims that a security token must contain in order to
 1039 satisfy the token assertion requirements.

1040 /sp:KerberosToken/wsp:Policy

1041 This required element identifies additional requirements for use of the sp:KerberosToken
 1042 assertion.

1043 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1044 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1045 and [Implied Derived Keys] properties for this token to 'true'.

1046 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1049 /sp:KerberosToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1050 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1051 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1052 'false'.

1053 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1054 This optional element is a policy assertion that indicates that a key identifier reference is required
 1055 when referencing this token.

1056 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

1057 This optional element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ token
 1058 should be used as defined in [\[WSS:KerberosTokenProfile1.1\]](#).

1059 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1060 This optional element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-REQ
 1061 token should be used as defined in [\[WSS:KerberosTokenProfile1.1\]](#).

1062 5.4.5 SpnegoContextToken Assertion

1063 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
 1064 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1065 Syntax

```

1066 <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1067   (
1068     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1069     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1070   ) ?
1071   <wst:Claims Dialect="..."> ... </wst:Claims> ?
1072   <wsp:Policy xmlns:wsp="...">
1073     (
1074       <sp:RequireDerivedKeys ... /> |
1075       <sp:RequireImpliedDerivedKeys ... /> |
1076       <sp:RequireExplicitDerivedKeys ... />
1077     ) ?
1078     <sp:MustNotSendCancel ... /> ?
1079     <sp:MustNotSendAmend ... /> ?
  
```

```

1080     <sp:MustNotSendRenew ... /> ?
1081     ...
1082     </wsp:Policy>
1083     ...
1084 </sp:SpnegoContextToken>

```

1085

1086 The following describes the attributes and elements listed in the schema outlined above:

1087 /sp:SpnegoContextToken

1088 This identifies a SpnegoContextToken assertion.

1089 /sp:SpnegoContextToken/@sp:IncludeToken

1090 This optional attribute identifies the token inclusion value for this token assertion.

1091 /sp:SpnegoContextToken/sp:Issuer

1092 This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for
1093 the Spnego Context Token.

1094 /sp:SpnegoContextToken/sp:IssuerName

1095 This optional element, of type xs:anyURI, contains the logical name of the
1096 sp:SpnegoContextToken issuer.

1097 /sp:SpnegoContextToken/wst:Claims

1098 This optional element identifies the required claims that a security token must contain in order to
1099 satisfy the token assertion requirements.

1100 /sp:SpnegoContextToken/wsp:Policy

1101 This required element identifies additional requirements for use of the sp:SpnegoContextToken
1102 assertion.

1103 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1104 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1105 and [Implied Derived Keys] properties for this token to 'true'.

1106 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1109 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1110 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1111 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1112 'false'.

1113 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

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

1117 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend

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

1121 /sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew

1122 This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token
1123 does not support SCT/Renew RST messages. If this assertion is missing it means that
1124 SCT/Renew RST messages are supported by the STS.

5.4.6 SecurityContextToken Assertion

This element represents a requirement for a SecurityContextToken token.

Syntax

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

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

/sp:SecurityContextToken

This identifies a SecurityContextToken assertion.

/sp:SecurityContextToken/@sp:IncludeToken

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

/sp:SecurityContextToken/sp:Issuer

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

/sp:SecurityContextToken/sp:IssuerName

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

/sp:SecurityContextToken/wst:Claims

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

/sp:SecurityContextToken/wsp:Policy

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

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

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

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

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

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

1171 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1172 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1173 'false'.

1174 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1175 This optional element is a policy assertion that indicates that an external URI reference is
1176 required when referencing this token.

1177 /sp:SecurityContextToken/wsp:Policy/sp:SC13SecurityContextToken

1178 This optional element is a policy assertion that indicates that a Security Context Token should be
1179 used as defined in [\[WS-SecureConversation\]](#).

1180

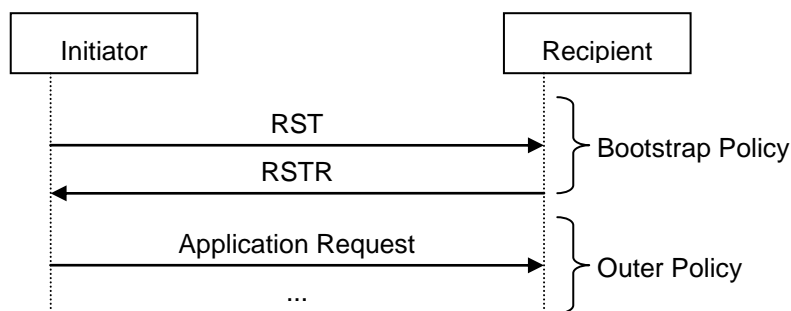
1181 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
1182 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
1183 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
1184 sp:SecureConversationToken or the sp:IssuedToken assertion should be used instead.

1185 5.4.7 SecureConversationToken Assertion

1186 This element represents a requirement for a Security Context Token retrieved from the indicated issuer
1187 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
1188 service endpoint address.

1189

1190 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1191 the target service and may not have a separate port (with separate policy), this assertion SHOULD
1192 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1193 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1194 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1195 below.



1196

1197 Syntax

```
1198 <sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1199 (
1200   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1201   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1202 ) ?
1203 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1204 <wsp:Policy xmlns:wsp="...">
1205   (
1206     <sp:RequireDerivedKeys ... /> |
1207     <sp:RequireImpliedDerivedKeys ... /> |
1208     <sp:RequireExplicitDerivedKeys ... />
1209   ) ?
1210   <sp:RequireExternalUriReference ... /> ?
1211   <sp:SC13SecurityContextToken ... /> ?
```

```

1212     <sp:MustNotSendCancel ... /> ?
1213     <sp:MustNotSendAmend ... /> ?
1214     <sp:MustNotSendRenew ... /> ?
1215     <sp:BootstrapPolicy ... >
1216         <wsp:Policy> ... </wsp:Policy>
1217     </sp:BootstrapPolicy> ?
1218 </wsp:Policy>
1219 ...
1220 </sp:SecureConversationToken>

```

- 1221
- 1222 The following describes the attributes and elements listed in the schema outlined above:
- 1223 /sp:SecureConversationToken
- 1224 This identifies a SecureConversationToken assertion.
- 1225 /sp:SecureConversationToken/@sp:IncludeToken
- 1226 This optional attribute identifies the token inclusion value for this token assertion.
- 1227 /sp:SecureConversationToken/sp:Issuer
- 1228 This optional element, of type `wsa:EndpointReferenceType`, contains a reference to the issuer for
- 1229 the Security Context Token.
- 1230 /sp:SecureConversationToken/sp:IssuerName
- 1231 This optional element, of type `xs:anyURI`, contains the logical name of the
- 1232 sp:SecureConversationToken issuer.
- 1233 /sp:SpnegoContextToken/wst:Claims
- 1234 This optional element identifies the required claims that a security token must contain in order to
- 1235 satisfy the token assertion requirements.
- 1236 /sp:SecureConversationToken/wsp:Policy
- 1237 This required element identifies additional requirements for use of the
- 1238 sp:SecureConversationToken assertion.
- 1239 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys
- 1240 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
- 1241 and [Implied Derived Keys] properties for this token to 'true'.
- 1242 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys
- 1243 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
- 1244 properties for this token to 'true' and the [Implied Derived Keys] property for this token to 'false'.
- 1245 /sp:SecureConversationToken/wsp:Policy/sp:RequireImpliedDerivedKeys
- 1246 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
- 1247 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
- 1248 'false'.
- 1249 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference
- 1250 This optional element is a policy assertion that indicates that an external URI reference is
- 1251 required when referencing this token.
- 1252 /sp:SecureConversationToken/wsp:Policy/sp:SC13SecurityContextToken
- 1253 This optional element is a policy assertion that indicates that a Security Context Token should be
- 1254 used as obtained using the protocol defined in [\[WS-SecureConversation\]](#).
- 1255 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

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

1259 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

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

1263 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1264 This optional element is a policy assertion that indicates that the STS issuing the secure
 1265 conversation token does not support SCT/Renew RST messages. If this assertion is missing it
 1266 means that SCT/Renew RST messages are supported by the STS.

1267 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1268 This optional element is a policy assertion that contains the policy indicating the requirements for
 1269 obtaining the Security Context Token.

1270 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1271 This element contains the security binding requirements for obtaining the Security Context Token.
 1272 It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
 1273 protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
 1274 are to be protected.

1275 **Example**

```

1276 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1277   <sp:SymmetricBinding>
1278     <wsp:Policy>
1279       <sp:ProtectionToken>
1280         <wsp:Policy>
1281           <sp:SecureConversationToken>
1282             <sp:Issuer>
1283               <wsa:Address>http://example.org/sts</wsa:Address>
1284             </sp:Issuer>
1285           </wsp:Policy>

```

```

1286         <sp:SC134SecurityContextToken />
1287         <sp:BootstrapPolicy>
1288             <wsp:Policy>
1289                 <sp:AsymmetricBinding>
1290                     <wsp:Policy>
1291                         <sp:InitiatorToken>
1292                             ...
1293                         </sp:InitiatorToken>
1294                         <sp:RecipientToken>
1295                             ...
1296                         </sp:RecipientToken>
1297                     </wsp:Policy>
1298                 </sp:AsymmetricBinding>
1299                 <sp:SignedParts>
1300                     ...
1301                 </sp:SignedParts>
1302                 ...
1303             </wsp:Policy>
1304         </sp:BootstrapPolicy>
1305     </wsp:Policy>
1306 </sp:SecureConversationToken>
1307 </wsp:Policy>
1308 </sp:ProtectionToken>
1309 ...
1310 </wsp:Policy>
1311 </sp:SymmetricBinding>
1312 <sp:SignedParts>
1313     ...
1314 </sp:SignedParts>
1315 ...
1316 </wsp:Policy>

```

5.4.8 SamlToken Assertion

This element represents a requirement for a SAML token.

Syntax

```

1320 <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1321 (
1322     <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1323     <sp:IssuerName>xs:anyURI</sp:IssuerName>
1324 ) ?
1325 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1326 <wsp:Policy xmlns:wsp="...">
1327 (
1328     <sp:RequireDerivedKeys ... /> |
1329     <sp:RequireImpliedDerivedKeys ... /> |
1330     <sp:RequireExplicitDerivedKeys ... />
1331 ) ?
1332 <sp:RequireKeyIdentifierReference ... /> ?
1333 (
1334     <sp:WssSamlV11Token10 ... /> |
1335     <sp:WssSamlV11Token11 ... /> |
1336     <sp:WssSamlV20Token11 ... />
1337 ) ?
1338 ...
1339 </wsp:Policy>
1340 ...
1341 </sp:SamlToken>

```

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

1344 /sp:SamIToken
 1345 This identifies a SamIToken assertion.

1346 /sp:SamIToken/@sp:IncludeToken
 1347 This optional attribute identifies the token inclusion value for this token assertion.

1348 /sp:SamIToken/sp:Issuer
 1349 This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of
 1350 the `sp:SamIToken`.

1351 /sp:SamIToken/sp:IssuerName
 1352 This optional element, of type `xs:anyURI`, contains the logical name of the `sp:SamIToken` issuer.

1353 /sp:SamIToken/wst:Claims
 1354 This optional element identifies the required claims that a security token must contain in order to
 1355 satisfy the token assertion requirements.

1356 /sp:SamIToken/wsp:Policy
 1357 This required element identifies additional requirements for use of the `sp:SamIToken` assertion.

1358 /sp:SamIToken/wsp:Policy/sp:RequireDerivedKeys
 1359 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
 1360 and [Implied Derived Keys] properties for this token to 'true'.

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

1364 /sp:SamIToken/wsp:Policy/sp:RequireImpliedDerivedKeys
 1365 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
 1366 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
 1367 'false'.

1368 /sp:SamIToken/wsp:Policy/sp:RequireKeyIdentifierReference
 1369 This optional element is a policy assertion that indicates that a key identifier reference is required
 1370 when referencing this token.

1371 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token10
 1372 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1373 used as defined in [\[WSS:SAMLTokenProfile1.0\]](#).

1374 /sp:SamIToken/wsp:Policy/sp:WssSamIV11Token11
 1375 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
 1376 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1377 /sp:SamIToken/wsp:Policy/sp:WssSamIV20Token11
 1378 This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be
 1379 used as defined in [\[WSS:SAMLTokenProfile1.1\]](#).

1380
 1381 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
 1382 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
 1383 of the mechanism for obtaining the SAML Token is desired in policy, the `sp:IssuedToken` assertion should
 1384 be used instead.

5.4.9 RelToken Assertion

This element represents a requirement for a REL token.

Syntax

```
<sp:RelToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
(
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
  <sp:IssuerName>xs:anyURI</sp:IssuerName>
) ?
<wst:Claims Dialect="..."> ... </wst:Claims> ?
<wsp:Policy xmlns:wsp="...">
(
  <sp:RequireDerivedKeys ... /> |
  <sp:RequireImpliedDerivedKeys ... /> |
  <sp:RequireExplicitDerivedKeys ... />
) ?
<sp:RequireKeyIdentifierReference ... /> ?
(
  <sp:WssRelV10Token10 ... /> |
  <sp:WssRelV20Token10 ... /> |
  <sp:WssRelV10Token11 ... /> |
  <sp:WssRelV20Token11 ... />
) ?
...
</wsp:Policy>
...
</sp:RelToken>
```

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

/sp:RelToken

This identifies a RelToken assertion.

/sp:RelToken/@sp:IncludeToken

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

/sp:RelToken/sp:Issuer

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

/sp:RelToken/sp:IssuerName

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

/sp:RelToken/wst:Claims

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

/sp:RelToken/wsp:Policy

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

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

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

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

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

1433 /sp:RelToken/wsp:Policy/sp:RequireImpliedDerivedKeys

1434 This optional element is a policy assertion that sets the [Derived Keys] and [Implied Derived
1435 Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this token to
1436 'false'.

1437 /sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1438 This optional element is a policy assertion that indicates that a key identifier reference is required
1439 when referencing this token.

1440 /sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1441 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1442 used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1443 /sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1444 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1445 used as defined in [\[WSS:RELTOKENProfile1.0\]](#).

1446 /sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1447 This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1448 used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1449 /sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1450 This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1451 used as defined in [\[WSS:RELTOKENProfile1.1\]](#).

1452

1453 Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
1454 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1455 of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion should
1456 be used instead.

1457 5.4.10 HttpsToken Assertion

1458 This element represents a requirement for a transport binding to support the use of HTTPS.

1459 Syntax

```
1460 <sp:HttpsToken xmlns:sp="..." ... >
1461 (
1462   <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1463   <sp:IssuerName>xs:anyURI</sp:IssuerName>
1464 ) ?
1465 <wst:Claims Dialect="..."> ... </wst:Claims> ?
1466 <wsp:Policy xmlns:wsp="...">
1467   (
1468     <sp:HttpBasicAuthentication /> |
1469     <sp:HttpDigestAuthentication /> |
1470     <sp:RequireClientCertificate /> |
1471     ...
1472   ) ?
1473   ...
1474 </wsp:Policy>
1475 ...
1476 </sp:HttpsToken>
```

1477 The following describes the attributes and elements listed in the schema outlined above:

1478 /sp:HttpsToken

1479 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1480 supported.

1481 /sp:HttpsToken/sp:Issuer
 1482 This optional element, of type `wsa:EndpointReferenceType`, contains reference to the issuer of
 1483 the `sp:HttpsToken`.

1484 /sp:HttpsToken/sp:IssuerName
 1485 This optional element, of type `xs:anyURI`, contains the logical name of the `sp:HttpsToken` issuer.

1486 /sp:HttpsToken/wst:Claims
 1487 This optional element identifies the required claims that a security token must contain in order to
 1488 satisfy the token assertion requirements.

1489 /sp:HttpsToken/wsp:Policy
 1490 This required element identifies additional requirements for use of the `sp:HttpsToken` assertion.

1491 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication
 1492 This optional element is a policy assertion that indicates that the client **MUST** use HTTP Basic
 1493 Authentication [RFC2068] to authenticate to the service.

1494 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication
 1495 This optional element is a policy assertion that indicates that the client **MUST** use HTTP Digest
 1496 Authentication [RFC2068] to authenticate to the service.

1497 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate
 1498 This optional element is a policy assertion that indicates that the client **MUST** provide a certificate
 1499 when negotiating the HTTPS session.

1500 5.4.11 KeyValueToken Assertion

1501 This element represents a requirement for a KeyValue token. The next section defines the KeyValue
 1502 security token abstraction for purposes of this token assertion.

1503
 1504 This document defines requirements for KeyValue token when used in combination with RSA
 1505 cryptographic algorithm. Additional cryptographic algorithms can be introduced in other specifications by
 1506 introducing new nested assertions besides `sp:RsaKeyValue`.

1507 Syntax

```
1508 <sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1509   <wsp:Policy xmlns:wsp="...">
1510     <sp:RsaKeyValue ... /> ?
1511     ...
1512   </wsp:Policy>
1513   ...
1514 </sp:KeyValueToken>
```

1515 The following describes the attributes listed in the schema outlined above:

1516 /sp:KeyValueToken
 1517 This identifies a RsaToken assertion.

1518 /sp:KeyValueToken/@sp:IncludeToken
 1519 This optional attribute identifies the token inclusion value for this token assertion.

1520 /sp:KeyValueToken/wsp:Policy
 1521 This required element identifies additional requirements for use of the `sp:KeyValueToken`
 1522 assertion.

1523 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1524 This optional element is a policy assertion that indicates that the `ds:RSAKeyValue` element must
1525 be present in the `KeyValue` token. This indicates that an RSA key pair must be used.

1526 5.4.11.1 KeyValue Token

1527 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key
1528 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this
1529 section is to define the `KeyValue` token abstraction that represents such key pair referencing mechanism.

1530 Although the `ds:KeyValue` element as defined in the XML Signature specification is generic enough to be
1531 used with any asymmetric cryptographic algorithm this document only profiles the usage of `ds:KeyValue`
1532 element in combination with RSA cryptographic algorithm.

1533 The RSA key pair is represented by the `ds:KeyInfo` element containing the `ds:KeyValue` element with the
1534 RSA public key value in `ds:RSAKeyValue` as defined in the XML Signature specification:

```
1537 <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">  
1538   <ds:KeyValue>  
1539     <ds:RSAKeyValue>  
1540       <ds:Modulus>ds:CryptoBinary</ds:Modulus>  
1541       <ds:Exponent>ds:CryptoBinary</ds:Exponent>  
1542     </ds:RSAKeyValue>  
1543   </ds:KeyValue>  
1544 </ds:KeyInfo>
```

1545 When the `KeyValue` token is used the corresponding public key value appears directly in the signature or
1546 encrypted data `ds:KeyInfo` element like in the following example. There is no `KeyValue` token
1547 manifestation outside the `ds:KeyInfo` element.

```
1549 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">  
1550   <SignedInfo>  
1551     <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-  
1552 c14n#" />  
1553     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1554     <Reference URI="#_1">  
1555       <Transforms>  
1556         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1557       </Transforms>  
1558       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />  
1559       <DigestValue>...</DigestValue>  
1560     </Reference>  
1561   </SignedInfo>  
1562   <SignatureValue>...</SignatureValue>  
1563   <KeyInfo>  
1564     <KeyValue>  
1565       <RSAKeyValue>  
1566         <Modulus>...</Modulus>  
1567         <Exponent>...</Exponent>  
1568       </RSAKeyValue>  
1569     </KeyValue>  
1570   </KeyInfo>  
1571 </Signature>
```

1572 Since there is no representation of the `KeyValue` token outside the `ds:KeyInfo` element and thus no
1573 identifier can be associated with the token, the `KeyValue` token cannot be referenced by using
1574 `wsse:SecurityTokenReference` element. However the `ds:KeyInfo` element representing the `KeyValue`
1575 token can be used whenever a security token can be used as illustrated on the following example:

```
1577 <t:RequestSecurityToken xmlns:t="...">  
1578   <t:RequestType>...</t:RequestType>  
1579   ...  
1580   <t:UseKey>  
1581     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
```

```
1582     <KeyValue>
1583         <RSAKeyValue>
1584             <Modulus>...</Modulus>
1585             <Exponent>...</Exponent>
1586         </RSAKeyValue>
1587     </KeyValue>
1588 </KeyInfo>
1589 </t:UseKey>
1590 </t:RequestSecurityToken>
```

6 Security Binding Properties

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

6.1 [Algorithm Suite] Property

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

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

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

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

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

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

Sha512 <http://www.w3.org/2001/04/xmlenc#sha512>
 Aes128 <http://www.w3.org/2001/04/xmlenc#aes128-cbc>
 Aes192 <http://www.w3.org/2001/04/xmlenc#aes192-cbc>
 Aes256 <http://www.w3.org/2001/04/xmlenc#aes256-cbc>
 TripleDes <http://www.w3.org/2001/04/xmlenc#tripledes-cbc>
 KwAes128 <http://www.w3.org/2001/04/xmlenc#kw-aes128>
 KwAes192 <http://www.w3.org/2001/04/xmlenc#kw-aes192>
 KwAes256 <http://www.w3.org/2001/04/xmlenc#kw-aes256>
 KwTripleDes <http://www.w3.org/2001/04/xmlenc#kw-tripledes>
 KwRsaOaep <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>
 KwRsa15 http://www.w3.org/2001/04/xmlenc#rsa-1_5
 PSha1 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L128 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L192 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 PSha1L256 http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk/p_sha1
 XPath <http://www.w3.org/TR/1999/REC-xpath-19991116>
 XPath20 <http://www.w3.org/2002/06/xmldsig-filter2>
 C14n <http://www.w3.org/2001/10/xml-c14n#>
 ExC14n <http://www.w3.org/2001/10/xml-exc-c14n#>
 SNT <http://www.w3.org/TR/soap12-n11n>
 STRT10 <http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-soap-message-security-1.0#STR-Transform>
 AbsXPath <http://docs.oasis-open.org/...TBD.../AbsXPath>

1627

1628 The tables below show all the base algorithm suites defined by this specification. This table defines
 1629 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 |

1630 This table defines additional properties whose values can be specified along with the default value for that
 1631 property.

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

1632 This table defines values for the remaining components for each algorithm suite.

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

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

6.2 [Timestamp] Property

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

6.3 [Protection Order] Property

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

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

The default value for this property is 'SignBeforeEncrypting'.

6.4 [Signature Protection] Property

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

6.5 [Token Protection] Property

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

6.6 [Entire Header and Body Signatures] Property

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

6.7 [Security Header Layout] Property

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

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

6.7.1 Strict Layout Rules for WSS 1.0

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

- 1684 b. A Username token (usually in encrypted form) MUST occur before the signature that
1685 signs it.
- 1686 c. A primary signature MUST occur before the supporting token signature that signs the
1687 primary signature's signature value element.
- 1688 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1689 has the same order requirements as the source plain text element, unless requirement 4
1690 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1691 supporting token signature per 2.c above and an encrypted token has the same ordering
1692 requirements as the unencrypted token.
- 1693 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1694 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1695 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1696 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1697 Layout Rules for WSS 1.1
- 1698 1. Tokens that are included in the message MUST be declared before use. For example:
- 1699 a. A local signing token MUST occur before the signature that uses it.
- 1700 b. A local token serving as the source token for a derived key token MUST occur before that
1701 derived key token.
- 1702 c. A local encryption token MUST occur before the reference list that points to
1703 xenc:EncryptedData elements that use it.
- 1704 d. If the same token is used for both signing and encryption, then it should appear before
1705 the ds:Signature and xenc:ReferenceList elements in the security header that are
1706 generated using the token.
- 1707 2. Signed elements inside the security header MUST occur before the signature that signs them.
1708 For example:
- 1709 a. A timestamp MUST occur before the signature that signs it.
- 1710 b. A Username token (usually in encrypted form) MUST occur before the signature that
1711 signs it.
- 1712 c. A primary signature MUST occur before the supporting token signature that signs the
1713 primary signature's signature value element.
- 1714 d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
- 1715 3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1716 has the same order requirements as the source plain text element, unless requirement 4
1717 indicates otherwise. For example, an encrypted primary signature MUST occur before any
1718 supporting token signature per 2.c above and an encrypted token has the same ordering
1719 requirements as the unencrypted token.
- 1720 4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1721 MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1722 xenc:EncryptedData elements in the security header that are referenced from the reference list.
1723 However, the xenc:ReferenceList is not required to appear before independently encrypted
1724 tokens such as the xenc:EncryptedKey token as defined in WSS.
- 1725 5. An xenc:EncryptedKey element without an internal reference list [[WSS: SOAP Message Security](#)
1726 1.1] MUST obey rule 1 above.

7 Security Binding Assertions

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

7.1 AlgorithmSuite Assertion

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

Syntax

```
<sp:AlgorithmSuite xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    (<sp:Basic256 ... /> |
    <sp:Basic192 ... /> |
    <sp:Basic128 ... /> |
    <sp:TripleDes ... /> |
    <sp:Basic256Rsa15 ... /> |
    <sp:Basic192Rsa15 ... /> |
    <sp:Basic128Rsa15 ... /> |
    <sp:TripleDesRsa15 ... /> |
    <sp:Basic256Sha256 ... /> |
    <sp:Basic192Sha256 ... /> |
    <sp:Basic128Sha256 ... /> |
    <sp:TripleDesSha256 ... /> |
    <sp:Basic256Sha256Rsa15 ... /> |
    <sp:Basic192Sha256Rsa15 ... /> |
    <sp:Basic128Sha256Rsa15 ... /> |
    <sp:TripleDesSha256Rsa15 ... /> |
    ...)
    <sp:InclusiveC14N ... /> ?
    <sp:SOAPNormalization10 ... /> ?
    <sp:STRTransform10 ... /> ?
    (<sp:XPath10 ... /> |
    <sp:XPathFilter20 ... /> |
    <sp:AbsXPath ... /> |
    ...) ?
    ...
  </wsp:Policy>
  ...
</sp:AlgorithmSuite>
```

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

/sp:AlgorithmSuite

This identifies an AlgorithmSuite assertion.

/sp:AlgorithmSuite/wsp:Policy

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

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

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

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

1776 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1777 to 'Basic192'.

1778 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1779 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1780 to 'Basic128'.

1781 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1782 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1783 to 'TripleDes'.

1784 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1785 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1786 to 'Basic256Rsa15'.

1787 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1788 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1789 to 'Basic192Rsa15'.

1790 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1791 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1792 to 'Basic128Rsa15'.

1793 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1794 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1795 to 'TripleDesRsa15'.

1796 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1797 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1798 to 'Basic256Sha256'.

1799 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1800 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1801 to 'Basic192Sha256'.

1802 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1803 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1804 to 'Basic128Sha256'.

1805 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1806 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1807 to 'TripleDesSha256'.

1808 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1809 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1810 to 'Basic256Sha256Rsa15'.

1811 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1812 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1813 to 'Basic192Sha256Rsa15'.

1814 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1815 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1816 to 'Basic128Sha256Rsa15'.

1817 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1818 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
 1819 to 'TripleDesSha256Rsa15'.

1820 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1821 This optional element is a policy assertion that indicates that the [C14N] property of an algorithm
 1822 suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N] property is
 1823 'ExcC14N'.

1824 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1825 This optional element is a policy assertion that indicates that the [SOAP Norm] property is set to
 1826 'SNT'.

1827 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1828 This optional element is a policy assertion that indicates that the [STR Transform] property is set
 1829 to 'STRT10'.

1830 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1831 This optional element is a policy assertion that indicates that the [XPath] property is set to 'XPath'.

1832 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1833 This optional element is a policy assertion that indicates that the [XPath] property is set to
 1834 'XPath20'.

1835 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1836 This optional element is a policy assertion that indicates that the [XPath] property is set to
 1837 'AbsXPath' (see [AbsoluteLocationPath](#) in [XPATH]).

1838

1839 7.2 Layout Assertion

1840 This assertion indicates a requirement for a particular security header layout as defined under the
 1841 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
 1842 containing assertion.

1843 Syntax

```

1844 <sp:Layout xmlns:sp="..." ... >
1845   <wsp:Policy xmlns:wsp="...">
1846     <sp:Strict ... /> |
1847     <sp:Lax ... /> |
1848     <sp:LaxTsFirst ... /> |
1849     <sp:LaxTsLast ... /> |
1850     ...
1851   </wsp:Policy>
1852   ...
1853 </sp:Layout>
  
```

1854

1855 The following describes the attributes and elements listed in the schema outlined above:

1856 /sp:Layout

1857 This identifies a Layout assertion.

1858 /sp:Layout/wsp:Policy

1859 This required element contains one or more policy assertions that indicate the specific security
 1860 header layout to use.

1861 /sp:Layout/wsp:Policy/sp:Strict

1862 This optional element is a policy assertion that indicates that the [Security Header Layout]
 1863 property is set to 'Strict'.
 1864 /sp:Layout/wsp:Policy/sp:Lax
 1865 This optional element is a policy assertion that indicates that the [Security Header Layout]
 1866 property is set to 'Lax'.
 1867 /sp:Layout/wsp:Policy/sp:LaxTsFirst
 1868 This optional element is a policy assertion that indicates that the [Security Header Layout]
 1869 property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
 1870 'true' by the presence of an sp:IncludeTimestamp assertion.
 1871 /sp:Layout/wsp:Policy/sp:LaxTsLast
 1872 This optional element is a policy assertion that indicates that the [Security Header Layout]
 1873 property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
 1874 'true' by the presence of an sp:IncludeTimestamp assertion.

1875 7.3 TransportBinding Assertion

1876 The TransportBinding assertion is used in scenarios in which message protection and security correlation
 1877 is provided by means other than [WSS: SOAP Message Security](#), for example by a secure transport like
 1878 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
 1879 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
 1880 MUST apply to [Endpoint Policy Subject].

1881 Syntax

```
1882 <sp:TransportBinding xmlns:sp="..." ... >
1883   <wsp:Policy xmlns:wsp="...">
1884     <sp:TransportToken ... >
1885       <wsp:Policy> ... </wsp:Policy>
1886       ...
1887     </sp:TransportToken>
1888     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1889     <sp:Layout ... > ... </sp:Layout> ?
1890     <sp:IncludeTimestamp ... /> ?
1891     ...
1892   </wsp:Policy>
1893   ...
1894 </sp:TransportBinding>
```

1895
 1896 The following describes the attributes and elements listed in the schema outlined above:

1897 /sp:TransportBinding
 1898 This identifies a TransportBinding assertion.
 1899 /sp:TransportBinding/wsp:Policy
 1900 This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
 1901 assertion.
 1902 /sp:TransportBinding/wsp:Policy/sp:TransportToken
 1903 This required element is a policy assertion that indicates a requirement for a Transport Token.
 1904 The specified token populates the [Transport Token] property and indicates how the transport is
 1905 secured.
 1906 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy
 1907 This indicates a nested policy that identifies the type of Transport Token to use.

1908 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite
 1909 This required element is a policy assertion that indicates a value that populates the [Algorithm
 1910 Suite] property. See Section 6.1 for more details.

1911 /sp:TransportBinding/wsp:Policy/sp:Layout
 1912 This optional element is a policy assertion that indicates a value that populates the [Security
 1913 Header Layout] property. See Section 6.7 for more details.

1914 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp
 1915 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
 1916 'true'.

1917 7.4 SymmetricBinding Assertion

1918 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
 1919 defined in [WSS: SOAP Message Security](#). This binding has two binding specific token properties;
 1920 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
 1921 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
 1922 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
 1923 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
 1924 properties and is used as the basis for both encryption and signature in both directions. This assertion
 1925 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

1926 Syntax

```

1927 <sp:SymmetricBinding xmlns:sp="..." ... >
1928   <wsp:Policy xmlns:wsp="...">
1929     (
1930       <sp:EncryptionToken ... >
1931         <wsp:Policy> ... </wsp:Policy>
1932       </sp:EncryptionToken>
1933       <sp:SignatureToken ... >
1934         <wsp:Policy> ... </wsp:Policy>
1935       </sp:SignatureToken>
1936     ) | (
1937       <sp:ProtectionToken ... >
1938         <wsp:Policy> ... </wsp:Policy>
1939       </sp:ProtectionToken>
1940     )
1941   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1942   <sp:Layout ... > ... </sp:Layout> ?
1943   <sp:IncludeTimestamp ... /> ?
1944   <sp:EncryptBeforeSigning ... /> ?
1945   <sp:EncryptSignature ... /> ?
1946   <sp:ProtectTokens ... /> ?
1947   <sp:OnlySignEntireHeadersAndBody ... /> ?
1948   ...
1949 </wsp:Policy>
1950   ...
1951 </sp:SymmetricBinding>
  
```

1952
 1953 The following describes the attributes and elements listed in the schema outlined above:

1954 /sp:SymmetricBinding
 1955 This identifies a SymmetricBinding assertion.

1956 /sp:SymmetricBinding/wsp:Policy
 1957 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
 1958 assertion.

1959 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken
 1960 This optional element is a policy assertion that indicates a requirement for an Encryption Token.
 1961 The specified token populates the [Encryption Token] property and is used for encryption. It is an
 1962 error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

1963 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy
 1964 The policy contained here MUST identify exactly one token to use for encryption.

1965 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken
 1966 This optional element is a policy assertion that indicates a requirement for a Signature Token.
 1967 The specified token populates the [Signature Token] property and is used for the message
 1968 signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
 1969 specified.

1970 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy
 1971 The policy contained here MUST identify exactly one token to use for signatures.

1972 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken
 1973 This optional element is a policy assertion that indicates a requirement for a Protection Token.
 1974 The specified token populates the [Encryption Token] and [Signature Token properties] and is
 1975 used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
 1976 assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
 1977 specified.

1978 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy
 1979 The policy contained here MUST identify exactly one token to use for protection.

1980 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 1981 This required element is a policy assertion that indicates a value that populates the [Algorithm
 1982 Suite] property. See Section 6.1 for more details.

1983 /sp:SymmetricBinding/wsp:Policy/sp:Layout
 1984 This optional element is a policy assertion that indicates a value that populates the [Security
 1985 Header Layout] property. See Section 6.7 for more details.

1986 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 1987 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
 1988 'true'.

1989 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 1990 This optional element is a policy assertion that indicates that the [Protection Order] property is set
 1991 to 'EncryptBeforeSigning'.

1992 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature
 1993 This optional element is a policy assertion that indicates that the [Signature Protection] property is
 1994 set to 'true'.

1995 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens
 1996 This optional element is a policy assertion that indicates that the [Token Protection] property is
 1997 set to 'true'.

1998 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 1999 This optional element is a policy assertion that indicates that the [Entire Header And Body
 2000 Signatures] property is set to 'true'.

7.5 AsymmetricBinding Assertion

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

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

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

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

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

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

Syntax

```
<sp:AsymmetricBinding xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:InitiatorToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorToken>
    ) | (
      <sp:InitiatorSignatureToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorSignatureToken>
      <sp:InitiatorEncryptionToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:InitiatorEncryptionToken>
    )
    (
      <sp:RecipientToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientToken>
    ) | (
      <sp:RecipientSignatureToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientSignatureToken>
      <sp:RecipientEncryptionToken>
        <wsp:Policy> ... </wsp:Policy>
      </sp:RecipientEncryptionToken>
    )
  </wsp:Policy>
</sp:AsymmetricBinding>
```

```

2053     </sp:RecipientEncryptionToken>
2054   )
2055   <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2056   <sp:Layout ... > ... </sp:Layout> ?
2057   <sp:IncludeTimestamp ... /> ?
2058   <sp:EncryptBeforeSigning ... /> ?
2059   <sp:EncryptSignature ... /> ?
2060   <sp:ProtectTokens ... /> ?
2061   <sp:OnlySignEntireHeadersAndBody ... /> ?
2062   ...
2063 </wsp:Policy>
2064   ...
2065 </sp:AsymmetricBinding>

```

2066

2067 The following describes the attributes and elements listed in the schema outlined above:

2068 /sp:AsymmetricBinding

2069 This identifies a AsymmetricBinding assertion.

2070 /sp:AsymmetricBinding/wsp:Policy

2071 This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
 2072 assertion.

2073 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2074 This optional element is a policy assertion that indicates a requirement for an Initiator Token. The
 2075 specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
 2076 properties and is used for the message signature from initiator to recipient, and encryption from
 2077 recipient to initiator.

2078 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2079 The policy contained here MUST identify one or more token assertions.

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

2081 This optional element is a policy assertion that indicates a requirement for an Initiator Signature
 2082 Token. The specified token populates the [Initiator Signature Token] property and is used for the
 2083 message signature from initiator to recipient.

2084 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2085 The policy contained here MUST identify one or more token assertions.

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

2087 This optional element is a policy assertion that indicates a requirement for an Initiator Encryption
 2088 Token. The specified token populates the [Initiator Encryption Token] property and is used for the
 2089 message encryption from recipient to initiator.

2090 /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2091 The policy contained here MUST identify one or more token assertions.

2092 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2093 This optional element is a policy assertion that indicates a requirement for a Recipient Token. The
 2094 specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
 2095 property and is used for encryption from initiator to recipient, and for the message signature from
 2096 recipient to initiator.

2097 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2098 The policy contained here MUST identify one or more token assertions.

2099 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2100 This optional element is a policy assertion that indicates a requirement for a Recipient Signature
 2101 Token. The specified token populates the [Recipient Signature Token] property and is used for
 2102 the message signature from R_{recipient} to recipientinitiator.

2103 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy
 2104 The policy contained here MUST identify one or more token assertions.

2105 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken
 2106 This optional element is a policy assertion that indicates a requirement for a Recipient Encryption
 2107 Token. The specified token populates the [Recipient Encryption Token] property and is used for
 2108 the message encryption from recipientinitiator to R_{recipient}.

2109 /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy
 2110 The policy contained here MUST identify one or more token assertions.

2111 /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite
 2112 This required element is a policy assertion that indicates a value that populates the [Algorithm
 2113 Suite] property. See Section 6.1 for more details.

2114 /sp:AsymmetricBinding/wsp:Policy/sp:Layout
 2115 This optional element is a policy assertion that indicates a value that populates the [Security
 2116 Header Layout] property. See Section 6.7 for more details.

2117 /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp
 2118 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
 2119 'true'.

2120 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning
 2121 This optional element is a policy assertion that indicates that the [Protection Order] property is set
 2122 to 'EncryptBeforeSigning'.

2123 /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature
 2124 This optional element is a policy assertion that indicates that the [Signature Protection] property is
 2125 set to 'true'.

2126 /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens
 2127 This optional element is a policy assertion that indicates that the [Token Protection] property is
 2128 set to 'true'.

2129 /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody
 2130 This optional element is a policy assertion that indicates that the [Entire Header And Body
 2131 Signatures] property is set to 'true'.

8 Supporting Tokens

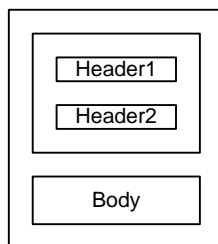
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. In case of Transport Binding the message is signed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

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

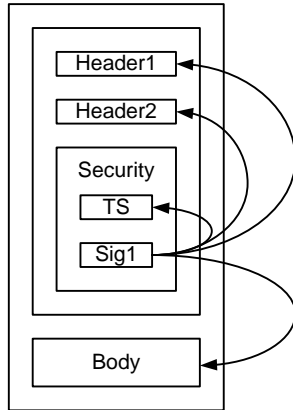
In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens should sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine which order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

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

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

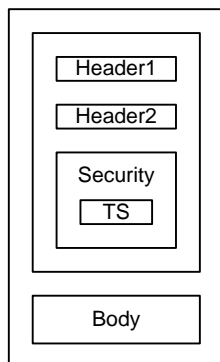


Even before any supporting tokens are added, each binding requires that the message is signed using a token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection. The signature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



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

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



8.1 SupportingTokens Assertion

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

Syntax

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

2196
2197
2198
2199
2200
2201
2202

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

2203

2204 The following describes the attributes and elements listed in the schema outlined above:

2205 /sp:SupportingTokens

2206 This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2207 Tokens] property.

2208 /sp:SupportingTokens/wsp:Policy

2209 This describes additional requirements for satisfying the SupportingTokens assertion.

2210 /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2211 The policy MUST identify one or more token assertions.

2212 /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2213 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2214 describes the algorithms to use for cryptographic operations performed with the tokens identified
2215 by this policy assertion.

2216 /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2217 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2218 describes additional message parts that MUST be included in the signature generated with the
2219 token identified by this policy assertion.

2220 /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2221 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
2222 describes additional message elements that MUST be included in the signature generated with
2223 the token identified by this policy assertion.

2224 /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2225 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2226 describes additional message parts that MUST be encrypted using the token identified by this
2227 policy assertion.

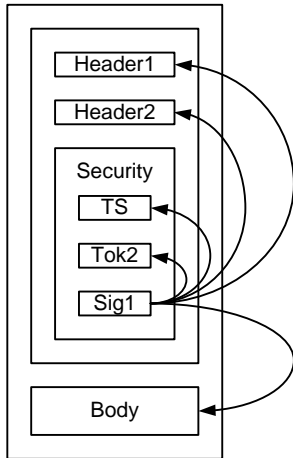
2228 /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2229 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2230 describes additional message elements that MUST be encrypted using the token identified by this
2231 policy assertion.

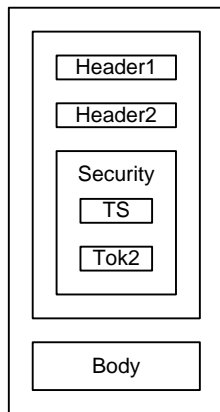
2232 8.2 SignedSupportingTokens Assertion

2233 Signed tokens are included in the “message signature” as defined above and may optionally include
2234 additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
2235 (Tok2) is signed by the message signature (Sig1):

2236



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



Syntax

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

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

/sp:SignedSupportingTokens

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

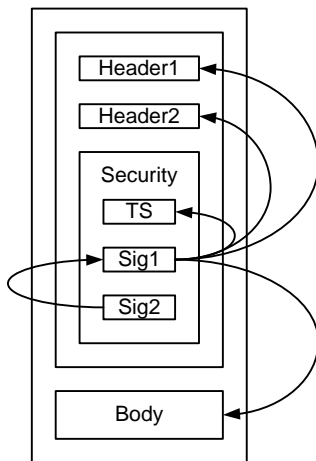
/sp:SignedSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the SignedSupportingTokens assertion.

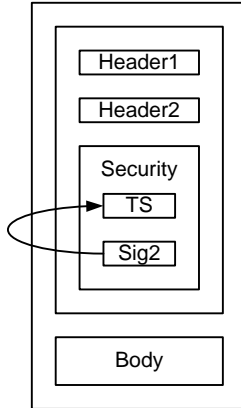
2263 /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]
 2264 The policy MUST identify one or more token assertions.
 2265 /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite
 2266 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
 2267 describes the algorithms to use for cryptographic operations performed with the tokens identified
 2268 by this policy assertion.
 2269 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts
 2270 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
 2271 describes additional message parts that MUST be included in the signature generated with the
 2272 token identified by this policy assertion.
 2273 /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements
 2274 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
 2275 describes additional message elements that MUST be included in the signature generated with
 2276 the token identified by this policy assertion.
 2277 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts
 2278 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
 2279 describes additional message parts that MUST be encrypted using the token identified by this
 2280 policy assertion.
 2281 /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements
 2282 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
 2283 describes additional message elements that MUST be encrypted using the token identified by this
 2284 policy assertion.

2285 8.3 EndorsingSupportingTokens Assertion

2286 Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
 2287 produced from the message signature and may optionally include additional message parts to sign and/or
 2288 encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message signature
 2289 (Sig1):
 2290



2291
 2292 If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
 2293 below:
 2294



2295

2296 Syntax

```

2297 <sp:EndorsingSupportingTokens xmlns:sp="..." ... >
2298   <wsp:Policy xmlns:wsp="...">
2299     [Token Assertion]+
2300     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2301     (
2302       <sp:SignedParts ... > ... </sp:SignedParts> |
2303       <sp:SignedElements ... > ... </sp:SignedElements> |
2304       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2305       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2306     ) *
2307     ...
2308   </wsp:Policy>
2309   ...
2310 </sp:EndorsingSupportingTokens>

```

2311

2312 The following describes the attributes and elements listed in the schema outlined above:

2313 /sp:EndorsingSupportingTokens

2314 This identifies an EndorsingSupportingTokens assertion. The specified tokens populate the
2315 [Endorsing Supporting Tokens] property.

2316 /sp:EndorsingSupportingTokens/wsp:Policy

2317 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2318 /sp:EndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2319 The policy MUST identify one or more token assertions.

2320 /sp:EndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2321 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2322 describes the algorithms to use for cryptographic operations performed with the tokens identified
2323 by this policy assertion.

2324 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2325 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2326 describes additional message parts that MUST be included in the signature generated with the
2327 token identified by this policy assertion.

2328 /sp:EndorsingSupportingTokens/wsp:Policy/sp:SignedElements

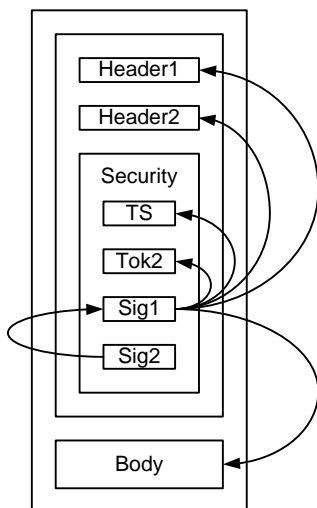
2329 This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
2330 describes additional message elements that MUST be included in the signature generated with
2331 the token identified by this policy assertion.

2332 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts
 2333 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
 2334 describes additional message parts that MUST be encrypted using the token identified by this
 2335 policy assertion.

2336 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements
 2337 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
 2338 describes additional message elements that MUST be encrypted using the token identified by this
 2339 policy assertion.

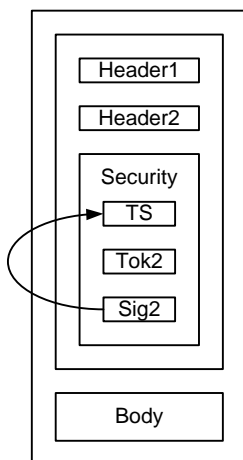
2340 8.4 SignedEndorsingSupportingTokens Assertion

2341 Signed endorsing tokens sign the entire `ds:Signature` element produced from the message signature
 2342 and are themselves signed by that message signature, that is both tokens (the token used for the
 2343 message signature and the signed endorsing token) sign each other. This assertion may optionally
 2344 include additional message parts to sign and/or encrypt. The diagram below illustrates how the signed
 2345 token (Tok2) is signed by the message signature (Sig1) and the endorsing signature (Sig2) signs the
 2346 message signature (Sig1):
 2347



2348

2349 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
 2350 should cover the message timestamp as illustrated below:
 2351



2353 Syntax

```
2354 <sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
2355   <wsp:Policy xmlns:wsp="...">
2356     [Token Assertion]+
2357     <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2358     (
2359       <sp:SignedParts ... > ... </sp:SignedParts> |
2360       <sp:SignedElements ... > ... </sp:SignedElements> |
2361       <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2362       <sp:EncryptedElements ... > ... </sp:EncryptedElements>
2363     ) *
2364     ...
2365   </wsp:Policy>
2366   ...
2367 </sp:SignedEndorsingSupportingTokens>
```

2368

2369 The following describes the attributes and elements listed in the schema outlined above:

2370 /sp:SignedEndorsingSupportingTokens

2371 This identifies a SignedEndorsingSupportingTokens assertion. The specified tokens populate the
2372 [Signed Endorsing Supporting Tokens] property.

2373 /sp:SignedEndorsingSupportingTokens/wsp:Policy

2374 This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

2375 /sp:SignedEndorsingSupportingTokens/wsp:Policy/[Token Assertion]

2376 The policy MUST identify one or more token assertions.

2377 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2378 This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2379 describes the algorithms to use for cryptographic operations performed with the tokens identified
2380 by this policy assertion.

2381 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedParts

2382 This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2383 describes additional message parts that MUST be included in the signature generated with the
2384 token identified by this policy assertion.

2385 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:SignedElements

2386 This optional element follows the schema outlined in Section 4.1.2 and describes additional
2387 message elements that MUST be included in the signature generated with the token identified by
2388 this policy assertion.

2389 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2390 This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2391 describes additional message parts that MUST be encrypted using the token identified by this
2392 policy assertion.

2393 /sp:SignedEndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2394 This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2395 describes additional message elements that MUST be encrypted using the token identified by this
2396 policy assertion.

8.5 SignedEncryptedSupportingTokens Assertion

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

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

8.6 EncryptedSupportingTokens Assertion

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

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

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

8.7 EndorsingEncryptedSupportingTokens Assertion

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

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

8.8 SignedEndorsingEncryptedSupportingTokens Assertion

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

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

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

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

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

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

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

8.10 Example

Example policy containing supporting token assertions:

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

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

9 WSS: SOAP Message Security Options

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

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

Note: This approach is chosen because:

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

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

WSS: SOAP Message Security 1.0 Properties

[Direct References]

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

[Key Identifier References]

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

[Issuer Serial References]

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

[External URI References]

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

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

[Embedded Token References]

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

WSS: SOAP Message Security 1.1 Properties

[Thumbprint References]

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

[EncryptedKey References]

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

[Signature Confirmation]

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

9.1 Wss10 Assertion

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

Syntax

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

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

2577 /sp:Wss10
 2578 This identifies a WSS10 assertion.

2579 /sp:Wss10/wsp:Policy
 2580 This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2581 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2582 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2583 is set to 'true'.

2584 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial
 2585 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2586 set to 'true'.

2587 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI
 2588 This optional element is a policy assertion indicates that the [External URI References] property is
 2589 set to 'true'.

2590 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken
 2591 This optional element is a policy assertion indicates that the [Embedded Token References]
 2592 property is set to 'true'.

2593 9.2 Wss11 Assertion

2594 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
 2595 supported.

2596 Syntax

```

2597 <sp:Wss11 xmlns:sp="..." ... >
2598   <wsp:Policy xmlns:wsp="...">
2599     <sp:MustSupportRefKeyIdentifier ... /> ?
2600     <sp:MustSupportRefIssuerSerial ... /> ?
2601     <sp:MustSupportRefExternalURI ... /> ?
2602     <sp:MustSupportRefEmbeddedToken ... /> ?
2603     <sp:MustSupportRefThumbprint ... /> ?
2604     <sp:MustSupportRefEncryptedKey ... /> ?
2605     <sp:RequireSignatureConfirmation ... /> ?
2606     ...
2607   </wsp:Policy>
2608 </sp:Wss11>
  
```

2609
 2610 The following describes the attributes and elements listed in the schema outlined above:

2611 /sp:Wss11
 2612 This identifies an WSS11 assertion.

2613 /sp:Wss11/wsp:Policy
 2614 This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2615 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier
 2616 This optional element is a policy assertion indicates that the [Key Identifier References] property
 2617 is set to 'true'.

2618 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial
 2619 This optional element is a policy assertion indicates that the [Issuer Serial References] property is
 2620 set to 'true'.

2621 /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI
2622 This optional element is a policy assertion indicates that the [External URI References] property is
2623 set to 'true'.

2624 /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken
2625 This optional element is a policy assertion indicates that the [Embedded Token References]
2626 property is set to 'true'.

2627 /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint
2628 This optional element is a policy assertion indicates that the [Thumbprint References] property is
2629 set to 'true'.

2630 /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey
2631 This optional element is a policy assertion indicates that the [EncryptedKey References] property
2632 is set to 'true'.

2633 /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation
2634 This optional element is a policy assertion indicates that the [Signature Confirmation] property is
2635 set to 'true'.

10 WS-Trust Options

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

WS-Trust 1.3 Properties

[Client Challenge]

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

[Server Challenge]

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

[Client Entropy]

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

[Server Entropy]

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

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

[Issued Tokens]

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

[Collection]

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

10.1 Trust13 Assertion

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

Syntax

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

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

/sp:Trust13

This identifies a Trust13 assertion.

/sp:Trust13/wsp:Policy

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

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

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

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

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

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

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

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

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

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

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

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

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

2724 /sp:Trust10/wsp:Policy/sp:RequireAppliesTo

2725 This optional element is a policy assertion that indicates that the STS requires the requestor to
2726 specify the scope for the issued token using wsp:AppliesTo in the RST.

11 Guidance on creating new assertions and assertion extensibility

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

11.1 General Design Points

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

11.2 Detailed Design Guidance

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

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

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

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

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

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

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

2771 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2772 attribute and implementations are expected to understand the meaning of all 5 values. If this information
2773 was encoded into the assertion QNames, each existing token assertion would require five variants, one
2774 for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2775
2776 Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2777 example, the token version assertions defined in Section 5 use such an approach. The overall token type
2778 assertion is parameterized by the nested token version assertions. Policy matching can use these
2779 parameters to find matches between policies where the broad token type is support by both parties but
2780 they might not support the same specific versions.

2781
2782 Note, when designing assertions for new token types such assertions SHOULD allow the
2783 sp:IncludeToken attribute and SHOULD allow nested policy.

2784

12 Security Considerations

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

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

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

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

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

A. Assertions and WS-PolicyAttachment

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

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

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

A.1.2 Token Assertions

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

A.1.3 WSS: SOAP Message Security 1.0 Assertions

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

A.1.4 WSS: SOAP Message Security 1.1 Assertions

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

A.1.5 Trust 1.0 Assertions

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

A.2 Operation Policy Subject Assertions

A.2.1 Security Binding Assertions

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

A.2.2 Supporting Token Assertions

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

| | | |
|------|--|---------------|
| 2834 | EndorsingSupportingTokens Assertion | (Section 8.3) |
| 2835 | SignedEndorsingSupportingTokens Assertion | (Section 8.4) |
| 2836 | SignedEncryptedSupportingTokens Assertion | (Section 8.5) |
| 2837 | EndorsingEncryptedSupportingTokens Assertion | (Section 8.6) |
| 2838 | SignedEndorsingEncryptedSupportingTokens Assertion | (Section 8.7) |

2839 **A.3 Message Policy Subject Assertions**

2840 **A.3.1 Supporting Token Assertions**

| | | |
|------|--|---------------|
| 2841 | SupportingTokens Assertion | (Section 8.1) |
| 2842 | SignedSupportingTokens Assertion | (Section 8.2) |
| 2843 | EndorsingSupportingTokens Assertion | (Section 8.3) |
| 2844 | SignedEndorsingSupportingTokens Assertion | (Section 8.4) |
| 2845 | SignedEncryptedSupportingTokens Assertion | (Section 8.5) |
| 2846 | EndorsingEncryptedSupportingTokens Assertion | (Section 8.6) |
| 2847 | SignedEndorsingEncryptedSupportingTokens Assertion | (Section 8.7) |

2848 **A.3.2 Protection Assertions**

| | | |
|------|--|-----------------|
| 2849 | SignedParts Assertion | (Section 4.1.1) |
| 2850 | SignedElements Assertion | (Section 4.1.2) |
| 2851 | EncryptedParts Assertion | (Section 4.2.1) |
| 2852 | EncryptedElements Assertion | (Section 4.2.2) |
| 2853 | ContentEncryptedElements Assertion | (Section 4.2.3) |
| 2854 | RequiredElements Assertion | (Section 4.3.1) |
| 2855 | RequiredParts Assertion | (Section 4.3.2) |

2856 **A.4 Assertions With Undefined Policy Subject**

2857 The assertions listed in this section do not have a defined policy subject because they appear nested
 2858 inside some other assertion which does have a defined policy subject. This list is derived from nested
 2859 assertions in the specification that have independent sections. It is not a complete list of nested
 2860 assertions. Many of the assertions previously listed in this appendix as well as the ones below have
 2861 additional nested assertions.

2862 **A.4.1 General Assertions**

| | | |
|------|--|---------------|
| 2863 | AlgorithmSuite Assertion | (Section 7.1) |
| 2864 | Layout Assertion | (Section 7.2) |

2865 **A.4.2 Token Usage Assertions**

2866 See the nested assertions under the [TransportBinding](#), [SymmetricBinding](#) and [AssymetricBinding](#)
 2867 assertions.

2868 **A.4.3 Token Assertions**

| | | |
|------|---|-----------------|
| 2869 | UsernameToken Assertion | (Section 5.3.1) |
|------|---|-----------------|

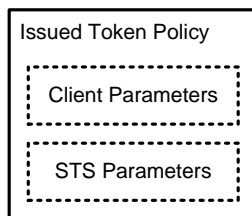
| | | |
|------|---|------------------|
| 2870 | IssuedToken Assertion | (Section 5.3.2) |
| 2871 | X509Token Assertion | (Section 5.3.3) |
| 2872 | KerberosToken Assertion | (Section 5.3.4) |
| 2873 | SpnegoContextToken Assertion | (Section 5.3.5) |
| 2874 | SecurityContextToken Assertion | (Section 5.3.6) |
| 2875 | SecureConversationToken Assertion | (Section 5.3.7) |
| 2876 | SamlToken Assertion | (Section 5.3.8) |
| 2877 | RelToken Assertion | (Section 5.3.9) |
| 2878 | HttpsToken Assertion | (Section 5.3.10) |

B. Issued Token Policy

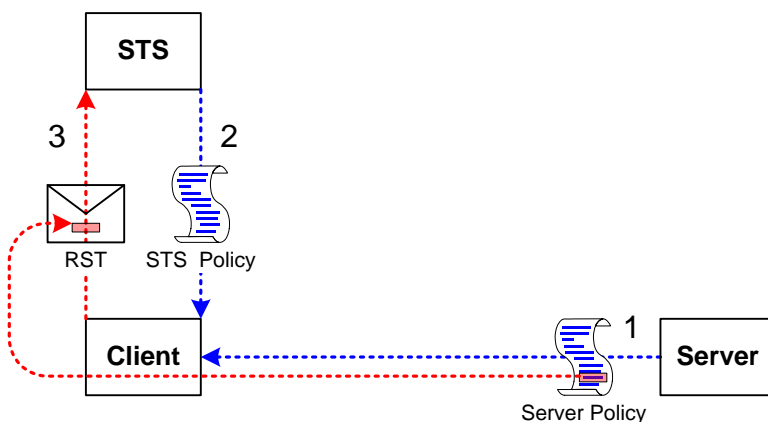
The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

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

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



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



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

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

2906 The Issued Token Policy Assertion contains elements which must be understood by the Client. The
2907 assertion contains one element which contains a list of arbitrary elements which should be sent along to
2908 the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
2909 request sent by the Client to the STS following the protocol defined in WS-Trust.
2910
2911 Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [\[WS-](#)
2912 [Trust\]](#). All items are optional, since the Server and STS may already have a pre-arranged relationship
2913 which specifies some or all of the conditions and constraints for issued tokens.

C. Strict Security Header Layout Examples

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

C.1 Transport Binding

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

C.1.1 Policy

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

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
    </wsp:Policy>
  </sp:TransportBinding>
  <sp:SignedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <sp:RequireSignatureConfirmation />
  </sp:Wss11>
</wsp:Policy>
```

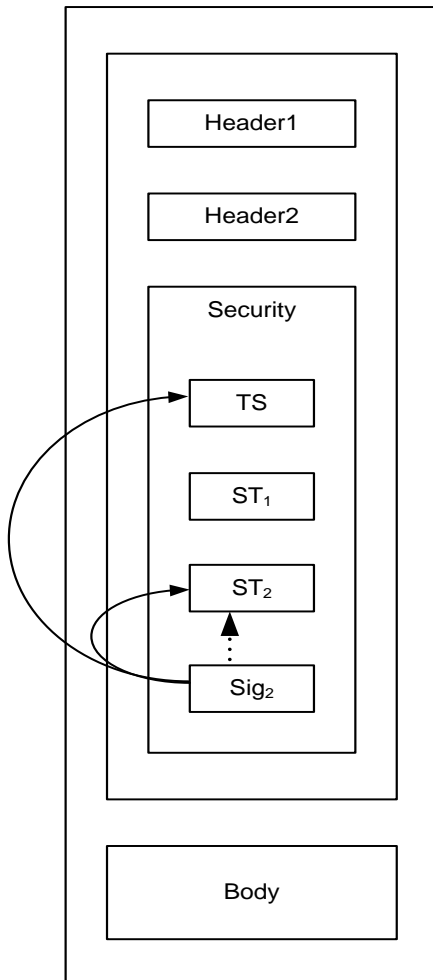
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.1.2 Initiator to Recipient Messages

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

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

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



The outer box shows that the entire message is protected (signed and encrypted) by the transport. The arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token labeled ST₂, namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST₂. The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering of the items in the security header follows the most optimal layout for a receiver to process its contents.

Example:

Initiator to recipient message

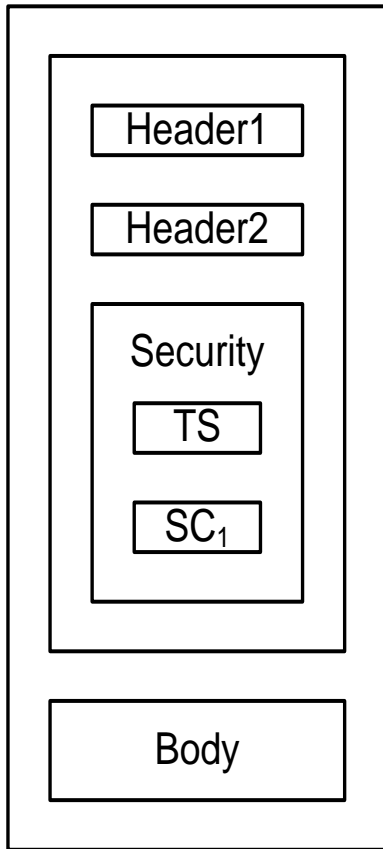
```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id='SomeSignedToken' >
        ...
      </wsse:UsernameToken>
      <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
        ...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:References>
            <ds:Reference URI="#timestamp" />
            <ds:Reference URI="#SomeSignedEndorsingToken" />
          </ds:References>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SomeSignedEndorsingToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      ...
    </wsse:Security>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

C.1.3 Recipient to Initiator Messages

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

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

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



3037

3038 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
 3039 `wsse11:SignatureConfirmation` element labeled `SC1` corresponding to the signature in the initial
 3040 message illustrated previously is included. In general, the ordering of the items in the security header
 3041 follows the most optimal layout for a receiver to process its contents.

3042 *Example:*

3043 Recipient to initiator message

```

3044 <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3045   <S:Header>
3046     ...
3047     <wsse:Security>
3048       <wsu:Timestamp wsu:Id="timestamp">
3049         <wsu:Created>[datetime]</wsu:Created>
3050         <wsu:Expires>[datetime]</wsu:Expires>
3051       </wsu:Timestamp>
3052       <wsse11:SignatureConfirmation Value="..." />
3053     ...
3054   </wsse:Security>
3055   ...
3056 </S:Header>
3057 <S:Body>
3058   ...
3059 </S:Body>
3060 </S:Envelope>

```

3061 C.2 Symmetric Binding

3062 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

C.2.1 Policy

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

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

```

3119 <!-- Example Message Policy -->
3120 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3121   <sp:SignedParts>
3122     <sp:Header Name="Header1" Namespace="..." />
3123     <sp:Header Name="Header2" Namespace="..." />
3124     <sp:Body/>
3125   </sp:SignedParts>
3126   <sp:EncryptedParts>
3127     <sp:Header Name="Header2" Namespace="..." />
3128     <sp:Body/>
3129   </sp:EncryptedParts>
3130 </wsp:Policy>
3131

```

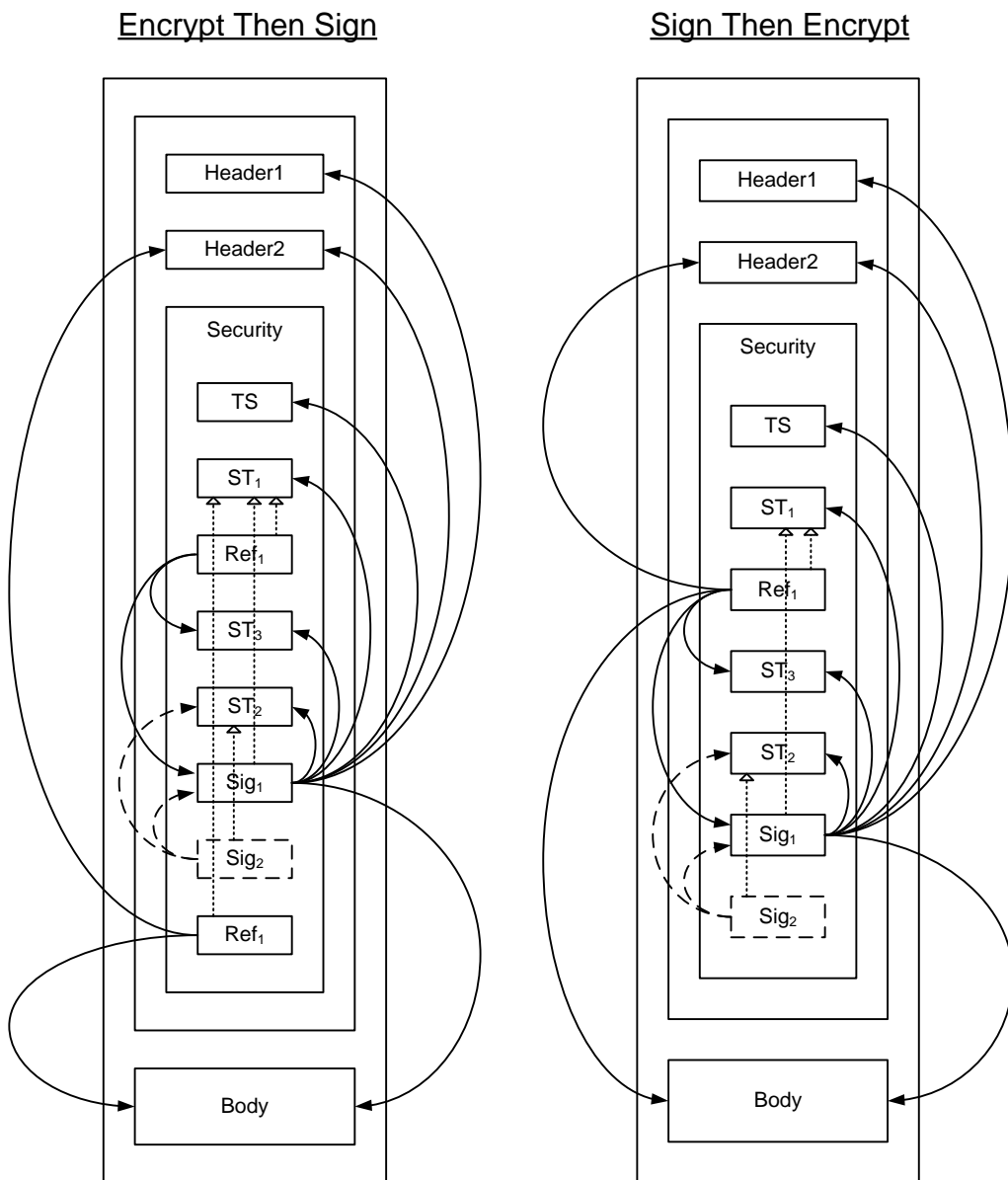
3132 This policy is used as the basis for the examples shown in the subsequent section describing the security
3133 header layout for this binding.

3134 C.2.2 Initiator to Recipient Messages

3135 Messages sent from initiator to recipient have the following layout for the security header:

- 3136 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
- 3137 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is `.../IncludeToken/Once` or
3138 `.../IncludeToken/Always`, then the [Encryption Token].
- 3139 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3140 Derived Key Token is used for encryption.
- 3141 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3142 reference list MUST include a reference to the message signature. If [Protection Order] is
3143 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3144 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3145 the token from 3 above MUST be used, otherwise the key in the [Encryption Token].
- 3146 5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens]
3147 properties whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or
3148 `.../IncludeToken/Always`.
- 3149 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3150 attribute on the [Signature Token] is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the
3151 [Signature Token].
- 3152 7. If [Derived Keys] is 'true', then a Derived Key Token based on the [Signature Token]. This
3153 Derived Key Token is used for signature.
- 3154 8. A signature over the `wsu:Timestamp` from 1 above, any tokens from 5 above regardless of
3155 whether they are included in the message, and any message parts specified in SignedParts
3156 assertions in the policy. If [Token Protection] is 'true', the signature MUST cover the [Signature
3157 Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in
3158 the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
- 3159 9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing
3160 Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection]
3161 is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the
3162 endorsing token is associated with a symmetric key, then a Derived Key Token, based on the
3163 endorsing token, appears before the signature.
- 3164 10. If [Protection Order] is 'EncryptBeforeSigning', then a reference list referencing all the message
3165 parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3166 in the token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2
3167 above.

3168
 3169 The following diagram illustrates the security header layout for the initiator to recipient message:



3170
 3171 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁.
 3172 The dashed arrows on the left from the box labeled Sig₂ indicate the parts signed by the supporting token
 3173 labeled ST₂, namely the message signature labeled Sig₁ and the token used as the basis for the
 3174 signature labeled ST₂. The arrows on the left from boxes labeled Ref₁ indicate references to parts
 3175 encrypted using a key based on the Shared Secret Token labeled ST₁. The dotted arrows inside the box
 3176 labeled Security indicate the token that was used as the basis for each cryptographic operation. In
 3177 general, the ordering of the items in the security header follows the most optimal layout for a receiver to
 3178 process its contents.
 3179 *Example:*
 3180 Initiator to recipient message using EncryptBeforeSigning:

```
3181 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3182   xmlns:wssell="..." xmlns:wsse="..." xmlns:saml="..."
3183   xmlns:xenc="..." xmlns:ds="...">
3184   <S:Header>
3185     <x:Header1 wsu:Id="Header1" >
3186       ...
3187     </x:Header1>
3188
```

```

3189 <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3190 <!-- Plaintext Header2
3191 <x:Header2 wsu:Id="Header2" >
3192 ...
3193 </x:Header2>
3194 -->
3195 ...
3196 </wsse1:EncryptedHeader>
3197 ...
3198 <wsse:Security>
3199 <wsu:Timestamp wsu:Id="Timestamp">
3200 <wsu:Created>...</wsu:Created>
3201 <wsu:Expires>...</wsu:Expires>
3202 </wsu:Timestamp>
3203 <saml:Assertion AssertionId="_SharedSecretToken" ...>
3204 ...
3205 </saml:Assertion>
3206 <xenc:ReferenceList>
3207 <xenc:DataReference URI="#enc_Signature" />
3208 <xenc:DataReference URI="#enc_SomeUsernameToken" />
3209 ...
3210 </xenc:ReferenceList>
3211 <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3212 <!-- Plaintext UsernameToken
3213 <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3214 ...
3215 </wsse:UsernameToken>
3216 -->
3217 ...
3218 <ds:KeyInfo>
3219 <wsse:SecurityTokenReference>
3220 <wsse:Reference URI="#_SharedSecretToken" />
3221 </wsse:SecurityTokenReference>
3222 </ds:KeyInfo>
3223 </xenc:EncryptedData>
3224 <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3225 ...
3226 </wsse:BinarySecurityToken>
3227 <xenc:EncryptedData ID="enc_Signature">
3228 <!-- Plaintext Signature
3229 <ds:Signature Id="Signature">
3230 <ds:SignedInfo>
3231 <ds:References>
3232 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3233 <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3234 <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3235 <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3236 <ds:Reference URI="#Header1" >...</ds:Reference>
3237 <ds:Reference URI="#Header2" >...</ds:Reference>
3238 <ds:Reference URI="#Body" >...</ds:Reference>
3239 </ds:References>
3240 </ds:SignedInfo>
3241 <ds:SignatureValue>...</ds:SignatureValue>
3242 <ds:KeyInfo>
3243 <wsse:SecurityTokenReference>
3244 <wsse:Reference URI="#_SharedSecretToken" />
3245 </wsse:SecurityTokenReference>
3246 </ds:KeyInfo>
3247 </ds:Signature>
3248 -->
3249 ...
3250 <ds:KeyInfo>
3251 <wsse:SecurityTokenReference>
3252 <wsse:Reference URI="#_SharedSecretToken" />

```

```

3253         </wsse:SecurityTokenReference>
3254     </ds:KeyInfo>
3255 </xenc:EncryptedData>
3256 <ds:Signature>
3257     <ds:SignedInfo>
3258         <ds:References>
3259             <ds:Reference URI="#Signature" >...</ds:Reference>
3260             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3261         </ds:References>
3262     </ds:SignedInfo>
3263     <ds:SignatureValue>...</ds:SignatureValue>
3264 </ds:KeyInfo>
3265     <wsse:SecurityTokenReference>
3266         <wsse:Reference URI="#SomeSupportingToken" />
3267     </wsse:SecurityTokenReference>
3268 </ds:KeyInfo>
3269 </ds:Signature>
3270 <xenc:ReferenceList>
3271     <xenc:DataReference URI="#enc_Body" />
3272     <xenc:DataReference URI="#enc_Header2" />
3273     ...
3274 </xenc:ReferenceList>
3275 </wsse:Security>
3276 </S:Header>
3277 <S:Body wsu:Id="Body">
3278     <xenc:EncryptedData Id="enc_Body">
3279         ...
3280     <ds:KeyInfo>
3281         <wsse:SecurityTokenReference>
3282             <wsse:Reference URI="#_SharedSecretToken" />
3283         </wsse:SecurityTokenReference>
3284     </ds:KeyInfo>
3285     </xenc:EncryptedData>
3286 </S:Body>
3287 </S:Envelope>

```

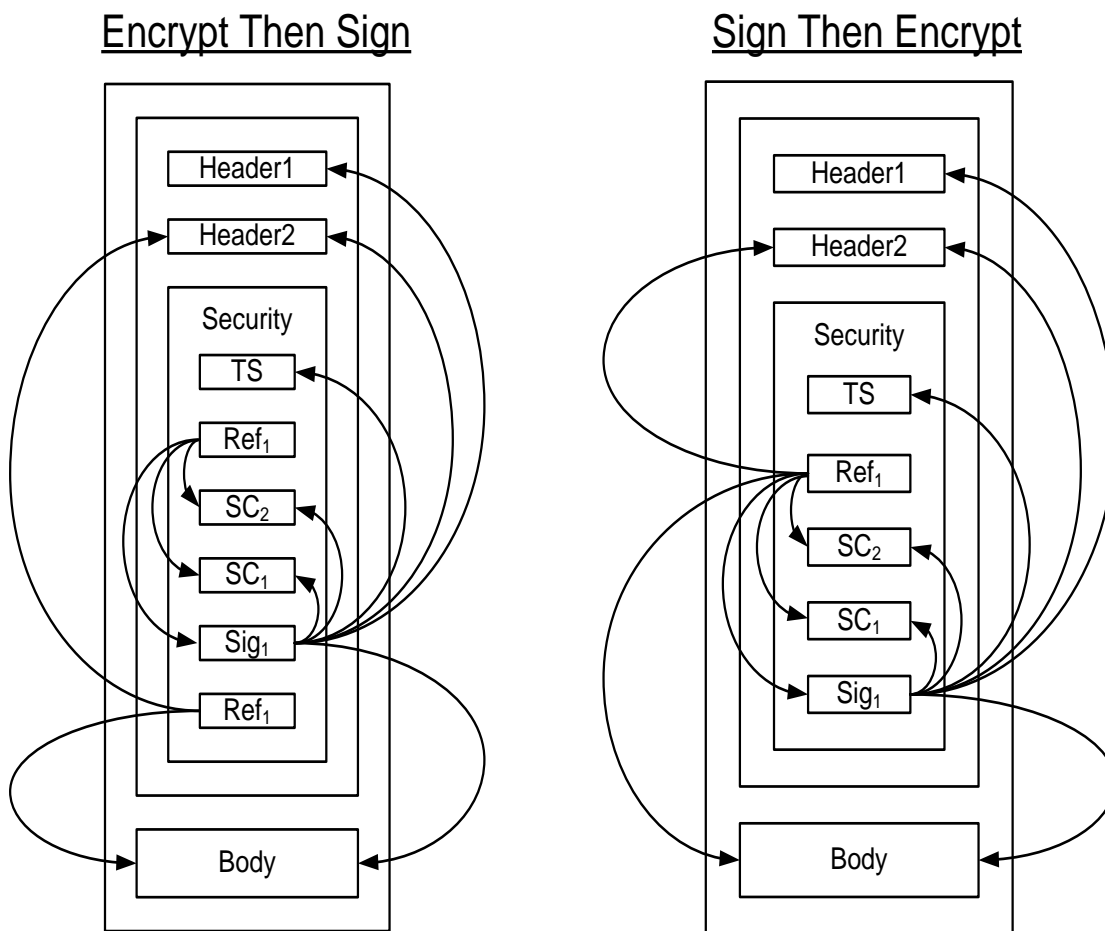
C.2.3 Recipient to Initiator Messages

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

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

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

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



The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₁. The arrows on the left from boxes labeled Ref₁ indicate references to parts encrypted using a key based on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two wssell:SignatureConfirmation elements labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is included. In general, the ordering of the items in the security header follows the most optimal layout for a receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

Example:

3329 Recipient to initiator message using EncryptBeforeSigning:

```
3330 <S:Envelope>
3331   <S:Header>
3332     <x:Header1 wsu:Id="Header1" >
3333       ...
3334     </x:Header1>
3335     <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3336       <!-- Plaintext Header2
3337       <x:Header2 wsu:Id="Header2" >
3338         ...
3339       </x:Header2>
3340       -->
3341       ...
3342     </wsse1:EncryptedHeader>
3343     ...
3344   <wsse:Security>
3345     <wsu:Timestamp wsu:Id="Timestamp">
3346       <wsu:Created>...</wsu:Created>
3347       <wsu:Expires>...</wsu:Expires>
3348     </wsu:Timestamp>
3349     <xenc:ReferenceList>
3350       <xenc:DataReference URI="#enc_Signature" />
3351       <xenc:DataReference URI="#enc_SigConf1" />
3352       <xenc:DataReference URI="#enc_SigConf2" />
3353       ...
3354     </xenc:ReferenceList>
3355     <xenc:EncryptedData ID="enc_SigConf1" >
3356       <!-- Plaintext SignatureConfirmation
3357       <wsse1:SignatureConfirmation wsu:Id="SigConf1" >
3358         ...
3359       </wsse1:SignatureConfirmation>
3360       -->
3361       ...
3362     </xenc:EncryptedData>
3363     <xenc:EncryptedData ID="enc_SigConf2" >
3364       <!-- Plaintext SignatureConfirmation
3365       <wsse1:SignatureConfirmation wsu:Id="SigConf2" >
3366         ...
3367       </wsse1:SignatureConfirmation>
3368       -->
3369       ...
3370     </xenc:EncryptedData>
```

```

3371 <xenc:EncryptedData Id="enc_Signature">
3372 <!-- Plaintext Signature
3373 <ds:Signature Id="Signature">
3374 <ds:SignedInfo>
3375 <ds:References>
3376 <ds:Reference URI="#Timestamp" >...</ds:Reference>
3377 <ds:Reference URI="#SigConf1" >...</ds:Reference>
3378 <ds:Reference URI="#SigConf2" >...</ds:Reference>
3379 <ds:Reference URI="#Header1" >...</ds:Reference>
3380 <ds:Reference URI="#Header2" >...</ds:Reference>
3381 <ds:Reference URI="#Body" >...</ds:Reference>
3382 </ds:References>
3383 </ds:SignedInfo>
3384 <ds:SignatureValue>...</ds:SignatureValue>
3385 <ds:KeyInfo>
3386 <wsse:SecurityTokenReference>
3387 <wsse:Reference URI="#_SomeIssuedToken" />
3388 </wsse:SecurityTokenReference>
3389 </ds:KeyInfo>
3390 </ds:Signature>
3391 </xenc:EncryptedData>
3392 ...
3393 <ds:KeyInfo>
3394 <wsse:SecurityTokenReference>
3395 <wsse:Reference URI="#_SomeIssuedToken" />
3396 </wsse:SecurityTokenReference>
3397 </ds:KeyInfo>
3398 <xenc:EncryptedData>
3399 <xenc:ReferenceList>
3400 <xenc:DataReference URI="#enc_Body" />
3401 <xenc:DataReference URI="#enc_Header2" />
3402 ...
3403 </xenc:ReferenceList>
3404 </xenc:EncryptedData>
3405 </wsse:Security>
3406 </S:Header>
3407 <S:Body wsu:Id="Body">
3408 <xenc:EncryptedData Id="enc_Body">
3409 ...
3410 <ds:KeyInfo>
3411 <wsse:SecurityTokenReference>
3412 <wsse:Reference URI="#_SomeIssuedToken" />
3413 </wsse:SecurityTokenReference>
3414 </ds:KeyInfo>
3415 </xenc:EncryptedData>
3416 </S:Body>
3417 </S:Envelope>

```

C.3 Asymmetric Binding

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

C.3.1 Policy

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

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

```
<!-- Example Endpoint Policy -->
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:AsymmetricBinding>
    <wsp:Policy>
      <sp:RecipientToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
        </wsp:Policy>
      </sp:RecipientToken>
      <sp:InitiatorToken>
        <wsp:Policy>
          <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
        </wsp:Policy>
      </sp:InitiatorToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:Basic256 />
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EncryptBeforeSigning />
      <sp:EncryptSignature />
      <sp:ProtectTokens />
    </wsp:Policy>
  </sp:AsymmetricBinding>
  <sp:SignedEncryptedSupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
    </wsp:Policy>
  </sp:SignedEncryptedSupportingTokens>
  <sp:SignedEndorsingSupportingTokens>
    <wsp:Policy>
      <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
        <wsp:Policy>
          <sp:WssX509v3Token10 />
        </wsp:Policy>
      </sp:X509Token>
    </wsp:Policy>
  </sp:SignedEndorsingSupportingTokens>
  <sp:Wss11>
    <wsp:Policy>
      <sp:RequireSignatureConfirmation />
    </wsp:Policy>
  </sp:Wss11>
</wsp:Policy>
```

```

3482 <!-- Example Message Policy -->
3483 <wsp:All xmlns:wsp="..." xmlns:sp="...">
3484   <sp:SignedParts>
3485     <sp:Header Name="Header1" Namespace="..." />
3486     <sp:Header Name="Header2" Namespace="..." />
3487     <sp:Body/>
3488   </sp:SignedParts>
3489   <sp:EncryptedParts>
3490     <sp:Header Name="Header2" Namespace="..." />
3491     <sp:Body/>
3492   </sp:EncryptedParts>
3493 </wsp:All>

```

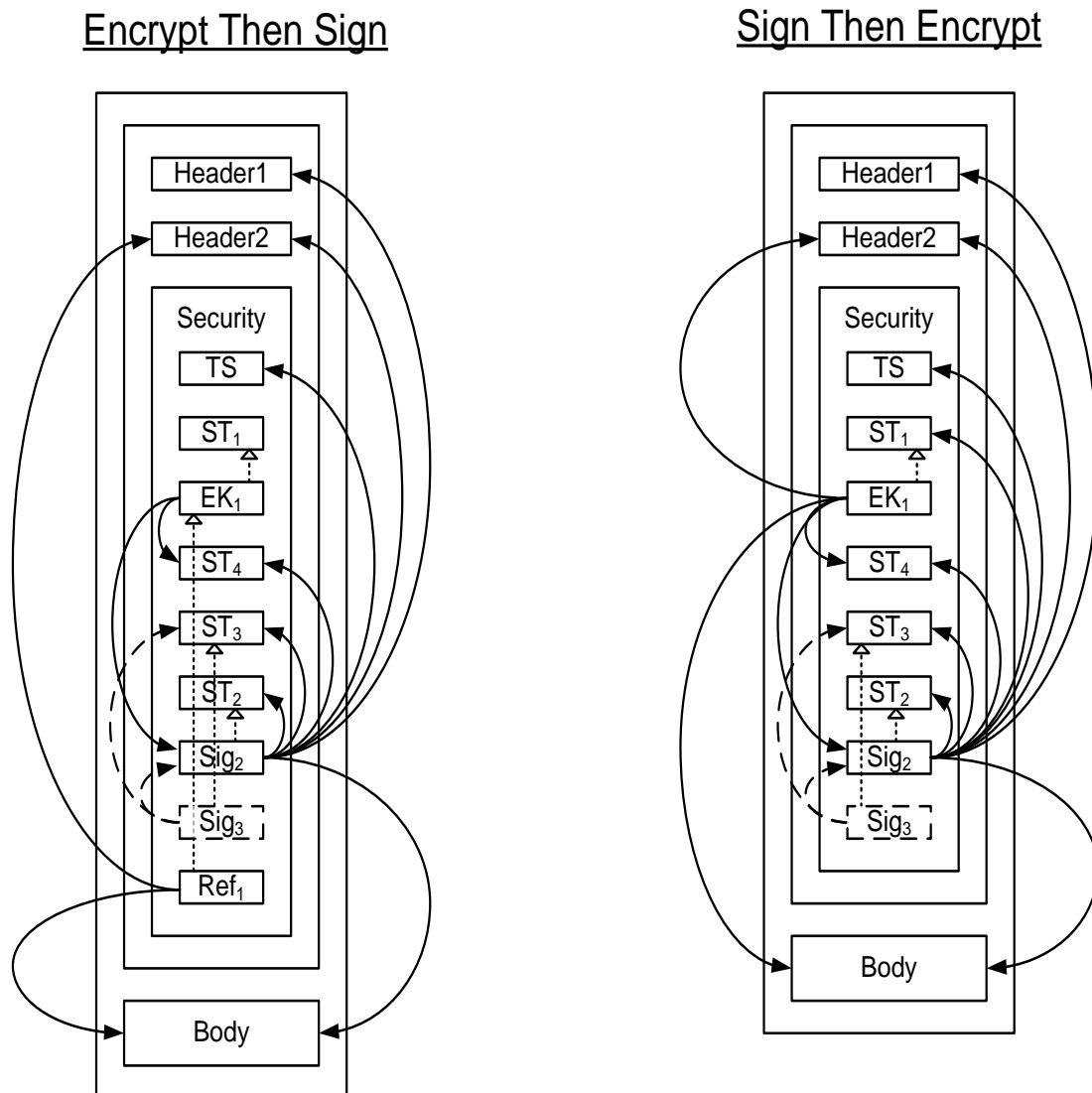
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.3.2 Initiator to Recipient Messages

Messages sent from initiator to recipient have the following layout:

1. A `wsu:Timestamp` element if `[Timestamp]` is 'true'.
2. If a `[Recipient Token]` is specified, and the associated `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Recipient Token]`.
3. If a `[Recipient Token]` is specified and `[Protection Order]` is 'SignBeforeEncrypting' or `[SignatureProtection]` is 'true' then an `xenc:EncryptedKey` element, containing a key encrypted for the recipient. The `xenc:EncryptedKey` element MUST include an `xenc:ReferenceList` containing a reference to all the message parts specified in `EncryptedParts` assertions in the policy. If `[Signature Protection]` is 'true' then the reference list MUST contain a reference to the message signature from 6 below. It is an error if `[Signature Protection]` is 'true' and there is not a message signature.
4. Any tokens from the supporting tokens properties (as defined in section 8) whose `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`.
5. If an `[Initiator Token]` is specified, and the associated `sp:IncludeToken` attribute is `.../IncludeToken/Once` or `.../IncludeToken/Always`, then the `[Initiator Token]`.
6. A signature based on the key in the `[Initiator Token]` if specified, over the `wsu:Timestamp` from 1 above, any tokens from 4 above regardless of whether they are included in the message, and any message parts specified in `SignedParts` assertions in the policy. If `[Token Protection]` is 'true', the signature MUST also cover the `[Initiator Token]` regardless of whether it is included in the message.
7. Signatures for tokens from the `[Endorsing Supporting Tokens]` and `[Signed Endorsing Supporting Tokens]` properties. If `[Derived Keys]` is 'true' and the supporting token is associated with a symmetric key, then a `Derived Key Token`, based on the supporting token, appears before the signature. If `[Token Protection]` is 'true', the signature MUST also cover the supporting token regardless of whether it is included in the message.
8. If a `[Recipient Token]` is specified and `[Protection Order]` is 'EncryptBeforeSigning' then if `[Signature Protection]` is 'false' then an `xenc:EncryptedKey` element, containing a key encrypted for the recipient and a reference list, else if `[Signature Protection]` is 'true', a reference list. The reference list includes a reference to all the message parts specified in `EncryptedParts` assertions in the policy. The encrypted parts MUST reference the key contained in the `xenc:EncryptedKey` element from 3 above.

3530 The following diagram illustrates the security header layout for the initiator to recipient messages:



3531

3532 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂

3533 using the [Initiator Token] labeled ST₂. The dashed arrows on the left from the box labeled Sig₃ indicate

3534 the parts signed by the supporting token ST₃, namely the message signature Sig₂ and the token used as

3535 the basis for the signature labeled ST₃. The arrows on the left from boxes labeled EK₁ indicate references

3536 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left

3537 from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the

3538 encrypted key labeled EK₁. The dotted arrows inside the box labeled Security indicate the token used as

3539 the basis for each cryptographic operation. In general, the ordering of the items in the security header

3540 follows the most optimal layout for a receiver to process its contents. The rules used to determine this

3541 ordering are described in Appendix C.

3542

3543 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted

3544 key contains an external reference to the token containing the encryption key. The diagram illustrates

3545 how one might attach a security token related to the encrypted key for completeness. One possible use-

3546 case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3547 wishes to include the encryption token in the message signature.

3548 Initiator to recipient message *Example*

3549 `<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."`

```

3550     xmlns:wsse1="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3551 <S:Header>
3552   <x:Header1 wsu:Id="Header1" >
3553     ...
3554   </x:Header1>
3555   <wsse1:EncryptedHeader wsu:Id="enc_Header2">
3556     <!-- Plaintext Header2
3557     <x:Header2 wsu:Id="Header2" >
3558       ...
3559     </x:Header2>
3560     -->
3561     ...
3562   </wsse1:EncryptedHeader>
3563   ...
3564   <wsse:Security>
3565     <wsu:Timestamp wsu:Id="Timestamp">
3566       <wsu:Created>...</wsu:Created>
3567       <wsu:Expires>...</wsu:Expires>
3568     </wsu:Timestamp>
3569     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3570       ...
3571     </wsse:BinarySecurityToken>
3572     <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3573       ...
3574       <xenc:ReferenceList>
3575         <xenc:DataReference URI="#enc_Signature" />
3576         <xenc:DataReference URI="#enc_SomeUsernameToken" />
3577         ...
3578       </xenc:ReferenceList>
3579     </xenc:EncryptedKey>
3580     <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3581       <!-- Plaintext UsernameToken
3582       <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3583         ...
3584       </wsse:UsernameToken>
3585       -->
3586       ...
3587     </xenc:EncryptedData>
3588     <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3589       ...
3590     </wsse:BinarySecurityToken>
3591     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3592       ...
3593     </wsse:BinarySecurityToken>
3594     <xenc:EncryptedData ID="enc_Signature">
3595       <!-- Plaintext Signature
3596       <ds:Signature Id="Signature">
3597         <ds:SignedInfo>
3598           <ds:References>
3599             <ds:Reference URI="#Timestamp" >...</ds:Reference>
3600             <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3601             <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3602             <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3603             <ds:Reference URI="#Header1" >...</ds:Reference>
3604             <ds:Reference URI="#Header2" >...</ds:Reference>
3605             <ds:Reference URI="#Body" >...</ds:Reference>
3606           </ds:References>
3607         </ds:SignedInfo>
3608         <ds:SignatureValue>...</ds:SignatureValue>
3609         <ds:KeyInfo>
3610           <wsse:SecurityTokenReference>
3611             <wsse:Reference URI="#InitiatorToken" />
3612           </wsse:SecurityTokenReference>
3613         </ds:KeyInfo>

```

```

3614     </ds:Signature>
3615     -->
3616     ...
3617 </xenc:EncryptedData>
3618 <ds:Signature>
3619   <ds:SignedInfo>
3620     <ds:References>
3621       <ds:Reference URI="#Signature" >...</ds:Reference>
3622       <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3623     </ds:References>
3624   </ds:SignedInfo>
3625   <ds:SignatureValue>...</ds:SignatureValue>
3626   <ds:KeyInfo>
3627     <wsse:SecurityTokenReference>
3628       <wsse:Reference URI="#SomeSupportingToken" />
3629     </wsse:SecurityTokenReference>
3630   </ds:KeyInfo>
3631 </ds:Signature>
3632 <xenc:ReferenceList>
3633   <xenc:DataReference URI="#enc_Body" />
3634   <xenc:DataReference URI="#enc_Header2" />
3635   ...
3636 </xenc:ReferenceList>
3637 </wsse:Security>
3638 </S:Header>
3639 <S:Body wsu:Id="Body">
3640   <xenc:EncryptedData Id="enc_Body">
3641     ...
3642     <ds:KeyInfo>
3643       <wsse:SecurityTokenReference>
3644         <wsse:Reference URI="#RecipientEncryptedKey" />
3645       </wsse:SecurityTokenReference>
3646     </ds:KeyInfo>
3647   </xenc:EncryptedData>
3648 </S:Body>
3649 </S:Envelope>

```

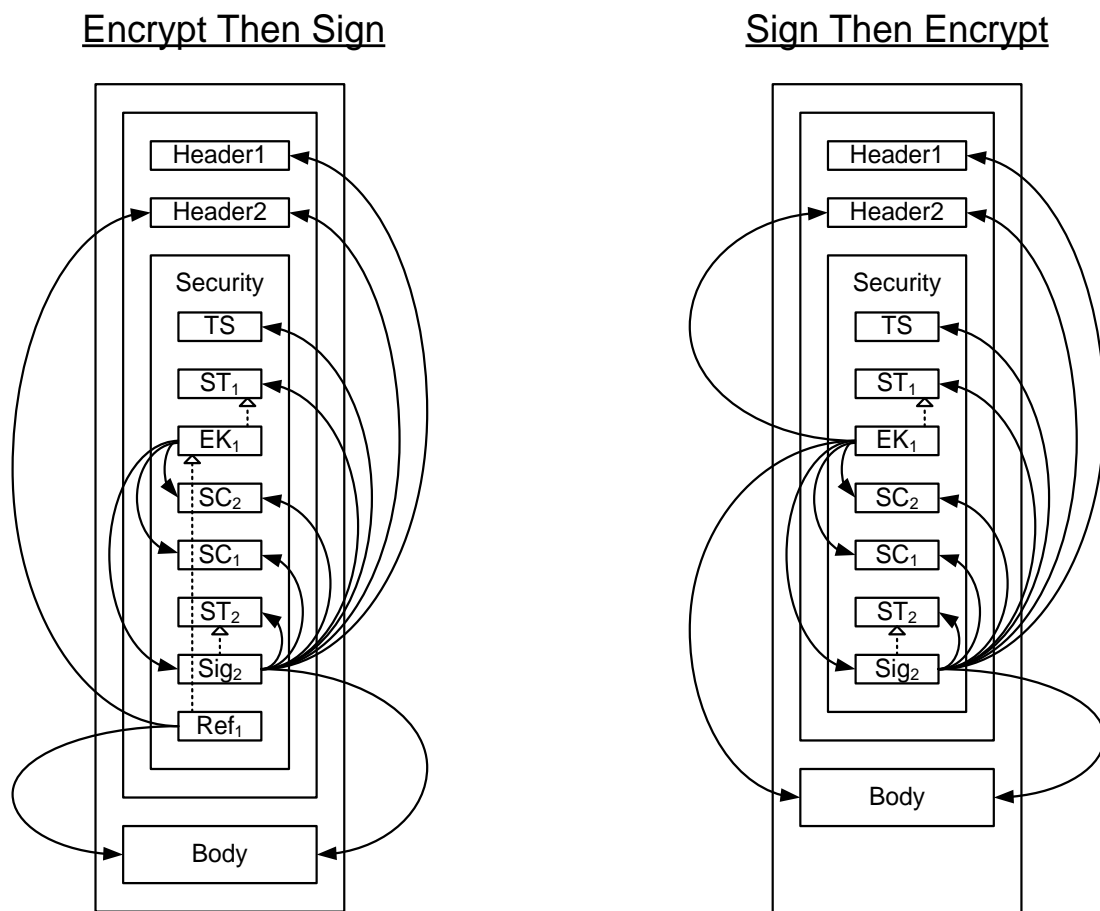
C.3.3 Recipient to Initiator Messages

Messages sent from recipient to initiator have the following layout:

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

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

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



The arrows on the right indicate parts that were signed as part of the message signature labeled Sig₂ using the [Recipient Token] labeled ST₂. The arrows on the left from boxes labeled E_K₁ indicate references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST₁. The arrows on the left from boxes labeled Ref₁ indicate additional references to parts encrypted using the key contained in the encrypted key labeled E_K₁. The dotted arrows inside the box labeled Security indicate the token used as the basis for each cryptographic operation. Two `wssell:SignatureConfirmation` elements labeled SC₁ and SC₂ corresponding to the two signatures in the initial message illustrated previously is included. In general, the ordering of the items in the security header follows the most optimal layout for a receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

Recipient to initiator message *Example*:

```

3691 <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3692     xmlns:wssell="..." xmlns:wsse="..."
3693     xmlns:xenc="..." xmlns:ds="...">
3694 <S:Header>
3695     <x:Header1 wsu:Id="Header1" >
3696         ...
3697     </x:Header1>
3698     <wssell:EncryptedHeader wsu:Id="enc_Header2">
3699         <!-- Plaintext Header2
3700         <x:Header2 wsu:Id="Header2" >
3701             ...
3702         </x:Header2>
3703         -->
3704         ...
3705     </wssell:EncryptedHeader>
3706     ...
3707 <wsse:Security>
3708     <wsu:Timestamp wsu:Id="Timestamp">
3709         <wsu:Created>...</wsu:Created>
3710         <wsu:Expires>...</wsu:Expires>
3711     </wsu:Timestamp>
3712     <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3713         ...
3714     </wsse:BinarySecurityToken>
3715     <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3716         ...
3717         <xenc:ReferenceList>
3718             <xenc:DataReference URI="#enc_Signature" />
3719             <xenc:DataReference URI="#enc_SigConf1" />
3720             <xenc:DataReference URI="#enc_SigConf2" />
3721             ...
3722         </xenc:ReferenceList>
3723     </xenc:EncryptedKey>
3724     <xenc:EncryptedData ID="enc_SigConf2" >
3725         <!-- Plaintext SignatureConfirmation
3726         <wssell:SignatureConfirmation wsu:Id="SigConf2" ...>
3727             ...
3728         </wssell:SignatureConfirmation>
3729         -->
3730         ...
3731     </xenc:EncryptedData>
3732     <xenc:EncryptedData ID="enc_SigConf1" >
3733         <!-- Plaintext SignatureConfirmation
3734         <wssell:SignatureConfirmation wsu:Id="SigConf1" ...>
3735             ...
3736         </wssell:SignatureConfirmation>
3737         -->
3738         ...
3739     </xenc:EncryptedData>
3740     <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3741         ...
3742     </wsse:BinarySecurityToken>
3743

```

```

3744 <xenc:EncryptedData ID="enc_Signature">
3745   <!-- Plaintext Signature
3746   <ds:Signature Id="Signature">
3747     <ds:SignedInfo>
3748       <ds:References>
3749         <ds:Reference URI="#Timestamp" >...</ds:Reference>
3750         <ds:Reference URI="#SigConf1" >...</ds:Reference>
3751         <ds:Reference URI="#SigConf2" >...</ds:Reference>
3752         <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3753         <ds:Reference URI="#Header1" >...</ds:Reference>
3754         <ds:Reference URI="#Header2" >...</ds:Reference>
3755         <ds:Reference URI="#Body" >...</ds:Reference>
3756       </ds:References>
3757     </ds:SignedInfo>
3758     <ds:SignatureValue>...</ds:SignatureValue>
3759     <ds:KeyInfo>
3760       <wsse:SecurityTokenReference>
3761         <wsse:Reference URI="#RecipientToken" />
3762       </wsse:SecurityTokenReference>
3763     </ds:KeyInfo>
3764   </ds:Signature>
3765   -->
3766   ...
3767 </xenc:EncryptedData>
3768 <xenc:ReferenceList>
3769   <xenc:DataReference URI="#enc_Body" />
3770   <xenc:DataReference URI="#enc_Header2" />
3771   ...
3772 </xenc:ReferenceList>
3773 </wsse:Security>
3774 </S:Header>
3775 <S:Body wsu:Id="Body">
3776   <xenc:EncryptedData Id="enc_Body">
3777     ...
3778     <ds:KeyInfo>
3779       <wsse:SecurityTokenReference>
3780         <wsse:Reference URI="#InitiatorEncryptedKey" />
3781       </wsse:SecurityTokenReference>
3782     </ds:KeyInfo>
3783   </xenc:EncryptedData>
3784 </S:Body>
3785 </S:Envelope>

```

D. Signed and Encrypted Elements in the Security Header

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

D.1 Elements signed by the message signature

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

D.2 Elements signed by all endorsing signatures

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

D.3 Elements signed by a specific endorsing signature

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

D.4 Elements that are encrypted

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

E. Acknowledgements

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

Original Authors of the initial contribution:

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

Original Acknowledgements of the initial contribution:

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

TC Members during the development of this specification:

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

3863 Lloyd Burch, Novell
3864 Scott Cantor, Internet2
3865 Greg Carpenter, Microsoft Corporation
3866 Steve Carter, Novell
3867 Symon Chang, BEA Systems, Inc.
3868 Ching-Yun (C.Y.) Chao, IBM
3869 Martin Chapman, Oracle Corporation
3870 Kate Cherry, Lockheed Martin
3871 Henry (Hyenvui) Chung, IBM
3872 Luc Clement, Systinet Corp.
3873 Paul Cotton, Microsoft Corporation
3874 Glen Daniels, Sonic Software Corp.
3875 Peter Davis, Neustar, Inc.
3876 Martijn de Boer, SAP AG
3877 Werner Dittmann, Siemens AG
3878 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
3879 Fred Dushin, IONA Technologies
3880 Petr Dvorak, Systinet Corp.
3881 Colleen Evans, Microsoft Corporation
3882 Ruchith Fernando, WSO2
3883 Mark Fussell, Microsoft Corporation
3884 Vijay Gajjala, Microsoft Corporation
3885 Marc Goodner, Microsoft Corporation
3886 Hans Granqvist, VeriSign
3887 Martin Gudgin, Microsoft Corporation
3888 Tony Gullotta, SOA Software Inc.
3889 Jiandong Guo, Sun Microsystems
3890 Phillip Hallam-Baker, VeriSign
3891 Patrick Harding, Ping Identity Corporation
3892 Heather Hinton, IBM
3893 Frederick Hirsch, Nokia Corporation
3894 Jeff Hodges, Neustar, Inc.
3895 Will Hopkins, BEA Systems, Inc.
3896 Alex Hristov, Otecia Incorporated
3897 John Hughes, PA Consulting
3898 Diane Jordan, IBM
3899 Venugopal K, Sun Microsystems
3900 Chris Kaler, Microsoft Corporation
3901 Dana Kaufman, Forum Systems, Inc.
3902 Paul Knight, Nortel Networks Limited
3903 Ramanathan Krishnamurthy, IONA Technologies
3904 Christopher Kurt, Microsoft Corporation
3905 Kelvin Lawrence, IBM
3906 Hubert Le Van Gong, Sun Microsystems
3907 Jong Lee, BEA Systems, Inc.
3908 Rich Levinson, Oracle Corporation
3909 Tommy Lindberg, Dajeil Ltd.
3910 Mark Little, JBoss Inc.
3911 Hal Lockhart, BEA Systems, Inc.
3912 Mike Lyons, Layer 7 Technologies Inc.
3913 Eve Maler, Sun Microsystems
3914 Ashok Malhotra, Oracle Corporation
3915 Anand Mani, CrimsonLogic Pte Ltd
3916 Jonathan Marsh, Microsoft Corporation
3917 Robin Martherus, Oracle Corporation
3918 Miko Matsumura, Infravio, Inc.
3919 Gary McAfee, IBM

3920 Michael McIntosh, IBM
3921 John Merrells, Sxip Networks SRL
3922 Jeff Mischkinsky, Oracle Corporation
3923 Prateek Mishra, Oracle Corporation
3924 Bob Morgan, Internet2
3925 Vamsi Motukuru, Oracle Corporation
3926 Raajmohan Na, EDS
3927 Anthony Nadalin, IBM
3928 Andrew Nash, Reactivity, Inc.
3929 Eric Newcomer, IONA Technologies
3930 Duane Nickull, Adobe Systems
3931 Toshihiro Nishimura, Fujitsu Limited
3932 Rob Philpott, RSA Security
3933 Denis Pilipchuk, BEA Systems, Inc.
3934 Darren Platt, Ping Identity Corporation
3935 Martin Raepple, SAP AG
3936 Nick Ragouzis, Enosis Group LLC
3937 Prakash Reddy, CA
3938 Alain Regnier, Ricoh Company, Ltd.
3939 Irving Reid, Hewlett-Packard
3940 Bruce Rich, IBM
3941 Tom Rutt, Fujitsu Limited
3942 Maneesh Sahu, Actional Corporation
3943 Frank Siebenlist, Argonne National Laboratory
3944 Joe Smith, Apani Networks
3945 Davanum Srinivas, WSO2
3946 Yakov Sverdlov, CA
3947 Gene Thurston, AmberPoint
3948 Victor Valle, IBM
3949 Asir Vedamuthu, Microsoft Corporation
3950 Greg Whitehead, Hewlett-Packard
3951 Ron Williams, IBM
3952 Corinna Witt, BEA Systems, Inc.
3953 Kyle Young, Microsoft Corporation