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

Table of Contents

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-wsecurity-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

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

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

[SwA] W3C Note, “SOAP Messages with Attachments”, 11 December 2000
http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211

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

[WS-Policy] W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
W3C Candidate Recommendation “Web Services Policy 1.5 – Framework”, 28 February 2007
http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/

[WS-PolicyAttachment] W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
W3C Candidate Recommendation “Web Services Policy 1.5 – Attachment”, 28 February 2007
http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/

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

[WS-SecureConversation] OASIS Committee Draft, “WS-SecureConversation 1.3", September 2006
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512

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

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

[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile", March 2004 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile 1.1", February 2006 http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf
[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", March 2004 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token Profile", February 2006 http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf
[WSS:KerberosToken1.1]	OASIS Standard, “Web Services Security Kerberos Token Profile 1.1”, February 2006 http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf
[WSS:SAMLTokenProfile1.0]	OASIS Standard, “Web Services Security: SAML Token Profile”, December 2004 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
[WSS:SAMLTokenProfile1.1]	OASIS Standard, “Web Services Security: SAML Token Profile 1.1”, February 2006 http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf
[WSS:RELTokenProfile1.0]	OASIS Standard, “Web Services Security Rights Expression Language (REL) Token Profile”, December 2004 http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
[WSS:RELTokenProfile1.1]	OASIS Standard, “Web Services Security Rights Expression Language (REL) Token Profile 1.1”, February 2006 http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf
[WSS:SwAProfile1.1]	OASIS Standard, “Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1”, February 2006

	http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[XPATH]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116
[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition", 28 October 2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition", 28 October 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.6 Non-Normative References

None.

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.

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.

- TransportBinding (Section 7.3)
- SymmetricBinding (Section 7.4)

- 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

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

wsdl:binding

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

wsdl:port

A policy expression containing one or more of the assertions with Endpoint Policy Subject MAY 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, it's 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

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

/sp:EncryptedParts/sp:Header/@Namespace

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

/sp:EncryptedParts/sp:Attachments

Presence of this optional empty element indicates that all SwA (SOAP Messages with Attachments) attachments [SwA] are to be confidentiality 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 confidentiality protected as outlined in WSS: SOAP Message Security [WSS:SwAProfile1.1].

4.2.2 EncryptedElements Assertion

The EncryptedElements assertion is used to specify arbitrary elements in the message that require confidentiality 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 EncryptedElements assertions present. Multiple EncryptedElements assertions present within a policy alternative are equivalent to a single EncryptedElements assertion containing the union of all specified XPath expressions.

Syntax

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

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

/sp:EncryptedElements

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

/sp:EncryptedElements/@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:EncryptedElements/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.

4.2.3 ContentEncryptedElements Assertion

The ContentEncryptedElements assertion is used to specify arbitrary elements in the message that require confidentiality protection of their content. 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 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.

/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.

/sp:RequiredElements/@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:RequiredElements/sp:XPath

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

4.3.2 RequiredParts Assertion

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

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

Syntax

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

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

/sp:RequiredParts/sp:Header

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

/sp:RequiredParts/sp:Header/@Name

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

/sp:RequiredParts/sp:Header/@Namespace

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

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

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

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

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

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

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

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

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

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

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

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

5.4.2 IssuedToken Assertion

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

Syntax

```
<sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  (
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
  <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
  <wst:Claims Dialect="..."> ... </wst:Claims> ?
  <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
    ...
  </sp:RequestSecurityTokenTemplate>
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:RequireDerivedKeys ... /> |
      <sp:RequireImpliedDerivedKeys ... /> |
      <sp:RequireExplicitDerivedKeys ... />
    ) ?
    <sp:RequireExternalReference ... /> ?
    <sp:RequireInternalReference ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:IssuedToken>
```

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

/sp:IssuedToken

This identifies an IssuedToken assertion.

/sp:IssuedToken/@sp:IncludeToken

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

/sp:IssuedToken/sp:Issuer

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

/sp:IssuedToken/sp:IssuerName

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

/sp:IssuedToken/wst:Claims

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

/sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

/sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

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

/sp:IssuedToken/wsp:Policy

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

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

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

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

This optional element is a policy assertion that indicates whether an internal reference is required when referencing this token.
Note: This reference will be supplied by the issuer of the token.

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

This optional element is a policy assertion that indicates whether an external reference is required when referencing this token.
Note: This reference will be supplied by the issuer of the token.

Note: The IssuedToken may or may not be associated with key material and such key material may be symmetric or asymmetric. The Binding assertion will imply the type of key associated with this token. Services may also include information in the `sp:RequestSecurityTokenTemplate` element to explicitly define the expected key type. See Appendix B for details of the `sp:RequestSecurityTokenTemplate` element.

5.4.3 X509Token Assertion

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

Syntax

```
<sp:X509Token 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:RequireExplicitDerivedKeys ... /> |
      <sp:RequireImpliedDerivedKeys ... />
    ) ?
    <sp:RequireKeyIdentifierReference ... /> ?
    <sp:RequireIssuerSerialReference ... /> ?
    <sp:RequireEmbeddedTokenReference ... /> ?
    <sp:RequireThumbprintReference ... /> ?
    (
      <sp:WssX509V3Token10 ... /> |
      <sp:WssX509Pkcs7Token10 ... /> |
      <sp:WssX509PkiPathV1Token10 ... /> |
      <sp:WssX509V1Token11 ... /> |
      <sp:WssX509V3Token11 ... /> |
      <sp:WssX509Pkcs7Token11 ... /> |
      <sp:WssX509PkiPathV1Token11 ... />
    ) ?
    ...
  </wsp:Policy>
  ...
</sp:X509Token>
```

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

/sp:X509Token

This identifies an X509Token assertion.

/sp:X509Token/@sp:IncludeToken

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

/sp:X509Token/sp:Issuer

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

/sp:X509Token/sp:IssuerName

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

/sp:X509Token/wst:Claims

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

/sp:X509Token/wsp:Policy

This required element identifies additional requirements for use of the sp:X509Token assertion.

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

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

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

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

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

This optional element is a policy assertion that indicates that an issuer serial reference is required when referencing this token.

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

This optional element is a policy assertion that indicates that an embedded token reference is required when referencing this token.

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

This optional element is a policy assertion that indicates that a thumbprint reference is required when referencing this token.

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

This optional element is a policy assertion that indicates that an X509 Version 3 token should be used as defined in [WSS:X509TokenProfile1.0].

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

This optional element is a policy assertion that indicates that an X509 PKCS7 token should be used as defined in [WSS:X509TokenProfile1.0].

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

This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token should be used as defined in [WSS:X509TokenProfile1.0].

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

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

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

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

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

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

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

This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token 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

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

/sp:KerberosToken/wst:Claims

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

/sp:KerberosToken/wsp:Policy

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

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

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

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

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

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

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

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

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

5.4.5 SpnegoContextToken Assertion

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

Syntax

```
<sp:SpnegoContextToken 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:MustNotSendCancel ... /> ?
    <sp:MustNotSendAmend ... /> ?
```

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

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

/sp:SpnegoContextToken

This identifies a SpnegoContextToken assertion.

/sp:SpnegoContextToken/@sp:IncludeToken

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

/sp:SpnegoContextToken/sp:Issuer

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

/sp:SpnegoContextToken/sp:IssuerName

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

/sp:SpnegoContextToken/wst:Claims

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

/sp:SpnegoContextToken/wsp:Policy

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

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

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

sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

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

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

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

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

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

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

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

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

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

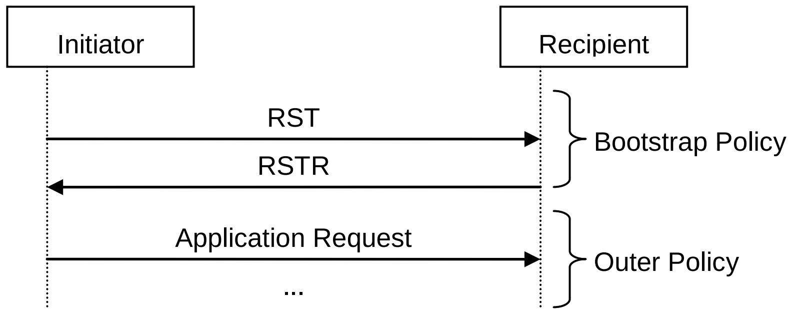
This optional element is a policy assertion that indicates that a Security Context Token should be used as defined in [WS-SecureConversation].

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

5.4.7 SecureConversationToken Assertion

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

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



Syntax

```
<sp:SecureConversationToken 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 ... /> ?
    <sp:MustNotSendCancel ... /> ?
    <sp:MustNotSendAmend ... /> ?
    <sp:MustNotSendRenew ... /> ?
    <sp:BootstrapPolicy ... >
      <wsp:Policy> ... </wsp:Policy>
    </sp:BootstrapPolicy> ?
  </wsp:Policy>
  ...
</sp:SecureConversationToken>
```

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

/sp:SecureConversationToken

This identifies a SecureConversationToken assertion.

/sp:SecureConversationToken/@sp:IncludeToken

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

/sp:SecureConversationToken/sp:Issuer

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

/sp:SecureConversationToken/sp:IssuerName

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

/sp:SpnegoContextToken/wst:Claims

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

/sp:SecureConversationToken/wsp:Policy

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <wsp:Policy>
          <sp:SecureConversationToken>
            <sp:Issuer>
              <wsa:Address>http://example.org/sts</wsa:Address>
            </sp:Issuer>
            <wsp:Policy>
```

```
              <sp:SC10SecurityContextToken />
              <sp:BootstrapPolicy>
                <wsp:Policy>
                  <sp:AsymmetricBinding>
                    <wsp:Policy>
                      <sp:InitiatorToken>
                        ...
                      </sp:InitiatorToken>
                      <sp:RecipientToken>
                        ...
                      </sp:RecipientToken>
                    </wsp:Policy>
                  </sp:AsymmetricBinding>
                  <sp:SignedParts>
                  ...
                  </sp:SignedParts>
                  ...
                </wsp:Policy>
              </sp:BootstrapPolicy>
            </wsp:Policy>
          </sp:SecureConversationToken>
        </wsp:Policy>
      </sp:ProtectionToken>
      ...
    </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:SignedParts>
  ...
  </sp:SignedParts>
  ...
</wsp:Policy>
```

5.4.8 SamlToken Assertion

This element represents a requirement for a SAML token.

Syntax

```
<sp:SamlToken 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:WssSamlV11Token10 ... /> |
      <sp:WssSamlV11Token11 ... /> |
      <sp:WssSamlV20Token11 ... />
    ) ?
    ...
  </wsp:Policy>
  ...
</sp:SamlToken>
```

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

/sp:SamlToken

This identifies a SamlToken assertion.

/sp:SamlToken/@sp:IncludeToken

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

/sp:SamlToken/sp:Issuer

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

/sp:SamlToken/sp:IssuerName

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

/sp:SamlToken/wst:Claims

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

/sp:SamlToken/wsp:Policy

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

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

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

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

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

/sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10

This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be used as defined in [WSS:SAMLTokenProfile1.0].

/sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11

This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be used as defined in [WSS:SAMLTokenProfile1.1].

/sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11

This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be used as defined in [WSS:SAMLTokenProfile1.1].

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

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

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

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

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

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

> This optional element is a policy assertion that identifies that a REL Version 1.0 token should be used as defined in [WSS:RELTokenProfile1.0].

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

> This optional element is a policy assertion that identifies that a REL Version 2.0 token should be used as defined in [WSS:RELTokenProfile1.0].

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

> This optional element is a policy assertion that identifies that a REL Version 1.0 token should be used as defined in [WSS:RELTokenProfile1.1].

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

> This optional element is a policy assertion that identifies that a REL Version 2.0 token should be used as defined in [WSS:RELTokenProfile1.1].

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

5.4.10 HttpsToken Assertion

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

Syntax

```
<sp:HttpsToken xmlns:sp="..." ... >
  (
    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
    <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
  <wst:Claims Dialect="..."> ... </wst:Claims> ?
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:HttpBasicAuthentication /> |
      <sp:HttpDigestAuthentication /> |
      <sp:RequireClientCertificate /> |
      ...
    )?
    ...
  </wsp:Policy>
  ...
</sp:HttpsToken>
```

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

/sp:HttpsToken

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

/sp:HttpsToken/sp:Issuer

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

/sp:HttpsToken/sp:IssuerName

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

/sp:HttpsToken/wst:Claims

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

/sp:HttpsToken/wsp:Policy

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

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

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

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

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

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

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

5.4.11 KeyValueToken Assertion

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

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

Syntax

```
<sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:RsaKeyValue ... /> ?
      ...
  </wsp:Policy>
  ...
</sp:KeyValueToken>
```

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

/sp:KeyValueToken

This identifies a RsaToken assertion.

/sp:KeyValueToken/@sp:IncludeToken

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

/sp:KeyValueToken/wsp:Policy

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

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

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

5.4.11.1 KeyValue Token

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

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

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

```
<ds:KeyInfo xmlns="http://www.w3/org/2000/09/xmldsig#">
  <ds:KeyValue>
    <ds:RSAKeyValue>
      <ds:Modulus>ds:CryptoBinary</ds:Modulus>
      <ds:Exponent>ds:CryptoBinary</ds:Exponent>
    </ds:RSAKeyValue>
  <ds:KeyValue>
</ds:KeyInfo>
```

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

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#_1">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>...</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <RSAKeyValue>
        <Modulus>...</Modulus>
        <Exponent>...</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

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

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

```
      <KeyValue>
        <RSAKeyValue>
          <Modulus>...</Modulus>
          <Exponent>...</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </t:UseKey>
</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/xmldsig#hmac-sha1
RsaSha1	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Sha1	http://www.w3.org/2000/09/xmldsig#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

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

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

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

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 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 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 wsse:Timestamp. Note that the [Timestamp] property MUST also be set to 'true' in this case.
LaxTimestampLast	As Lax except that the last item in the security header MUST be a wsse:Timestamp. Note that the [Timestamp] property MUST also be set to 'true' in this case.

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.

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

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

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.
    b. A Username token (usually in encrypted form) MUST occur before the signature that signs it.
    c. A primary signature MUST occur before the supporting token signature that signs the primary signature's signature value element.
    d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.
3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element has the same order requirements as the source plain text element, unless requirement 4 indicates otherwise. For example, an encrypted primary signature MUST occur before any supporting token signature per 2.c above and an encrypted token has the same ordering requirements as the unencrypted token.
4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element MUST be present in the security header. The xenc:ReferenceList MUST occur before any xenc:EncryptedData elements in the security header that are referenced from the reference list. However, the xenc:ReferenceList is not required to appear before independently encrypted tokens such as the xenc:EncryptedKey token as defined in WSS.
5. An xenc:EncryptedKey element without an internal reference list [WSS: SOAP Message Security 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.2 Layout Assertion

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

Syntax

```
<sp:Layout xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:Strict ... /> |
    <sp:Lax ... /> |
    <sp:LaxTsFirst ... /> |
    <sp:LaxTsLast ... /> |
    ...
  </wsp:Policy>
  ...
</sp:Layout>
```

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

/sp:Layout

This identifies a Layout assertion.

/sp:Layout/wsp:Policy

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

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

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

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

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

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

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

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

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

7.3 TransportBinding Assertion

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

Syntax

```
<sp:TransportBinding xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:TransportToken ... >
      <wsp:Policy> ... </wsp:Policy>
      ...
    </sp:TransportToken>
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
    <sp:Layout ... > ... </sp:Layout> ?
    <sp:IncludeTimestamp ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:TransportBinding>
```

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

/sp:TransportBinding

This identifies a TransportBinding assertion.

/sp:TransportBinding/wsp:Policy

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

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

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

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

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

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

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

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

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

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

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

7.4 SymmetricBinding Assertion

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

Syntax

```
<sp:SymmetricBinding xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:EncryptionToken ... >
        <wsp:Policy> ... </wsp:Policy>
      </sp:EncryptionToken>
      <sp:SignatureToken ... >
        <wsp:Policy> ... </wsp:Policy>
      </sp:SignatureToken>
    ) | (
      <sp:ProtectionToken ... >
        <wsp:Policy> ... </wsp:Policy>
      </sp:ProtectionToken>
    )
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
    <sp:Layout ... > ... </sp:Layout> ?
    <sp:IncludeTimestamp ... /> ?
    <sp:EncryptBeforeSigning ... /> ?
    <sp:EncryptSignature ... /> ?
    <sp:ProtectTokens ... /> ?
    <sp:OnlySignEntireHeadersAndBody ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:SymmetricBinding>
```

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

/sp:SymmetricBinding

This identifies a SymmetricBinding assertion.

/sp:SymmetricBinding/wsp:Policy

This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding assertion.

/sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken

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

/sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy

The policy contained here MUST identify exactly one token to use for encryption.

/sp:SymmetricBinding/wsp:Policy/sp:SignatureToken

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

/sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy

The policy contained here MUST identify exactly one token to use for signatures.

/sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken

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

/sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy

The policy contained here MUST identify exactly one token to use for protection.

/sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite

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

/sp:SymmetricBinding/wsp:Policy/sp:Layout

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

/sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp

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

/sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

This optional element is a policy assertion that indicates that the [Protection Order] property is set to 'EncryptBeforeSigning'.

/sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature

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

/sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens

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

/sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

This optional element is a policy assertion that indicates that the [Entire Header And Body 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>
    )
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
    <sp:Layout ... > ... </sp:Layout> ?
    <sp:IncludeTimestamp ... /> ?
    <sp:EncryptBeforeSigning ... /> ?
    <sp:EncryptSignature ... /> ?
    <sp:ProtectTokens ... /> ?
    <sp:OnlySignEntireHeadersAndBody ... /> ?
    ...
  </wsp:Policy>
  ...
</sp:AsymmetricBinding>
```

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

/sp:AsymmetricBinding

This identifies a AsymmetricBinding assertion.

/sp:AsymmetricBinding/wsp:Policy

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

/sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken

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

/sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy

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

/sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite

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

/sp:AsymmetricBinding/wsp:Policy/sp:Layout

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

/sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp

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

/sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

This optional element is a policy assertion that indicates that the [Protection Order] property is set to 'EncryptBeforeSigning'.

/sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature

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

/sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens

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

/sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

This optional element is a policy assertion that indicates that the [Entire Header And Body Signatures] property is set to 'true'.

8 Supporting Tokens

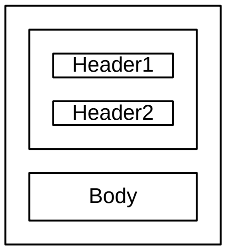
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. In case of Transport Binding the message is signed 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.

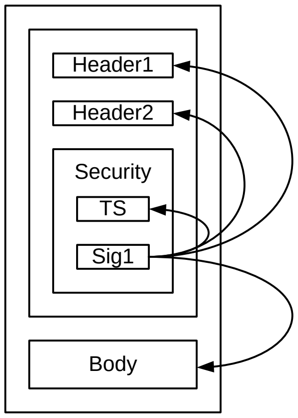
In cases where multiple tokens are specified that sign and/or encrypt overlapping message parts, all the tokens should sign and encrypt the various message parts. In such cases ordering of elements (tokens, signatures, reference lists etc.) in the security header would be used to determine which order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

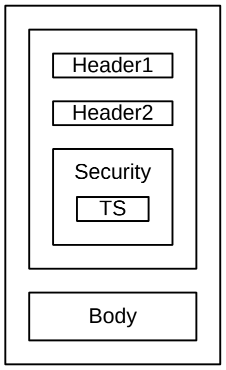


Even before any supporting tokens are added, each binding requires that the message is signed using a token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection. The signature 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> |
      <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
      <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
    ) *
    ...
  </wsp:Policy>
  ...
</sp:SupportingTokens>
```

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

/sp:SupportingTokens

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

/sp:SupportingTokens/wsp:Policy

This describes additional requirements for satisfying the SupportingTokens assertion.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

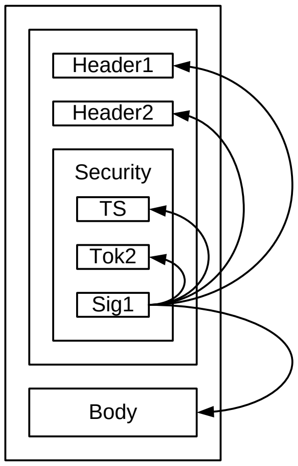
This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

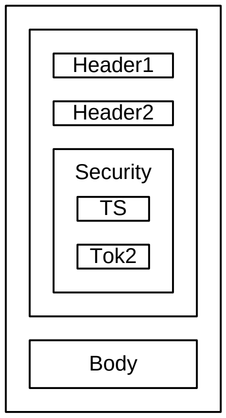
This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

8.2 SignedSupportingTokens Assertion

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



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.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

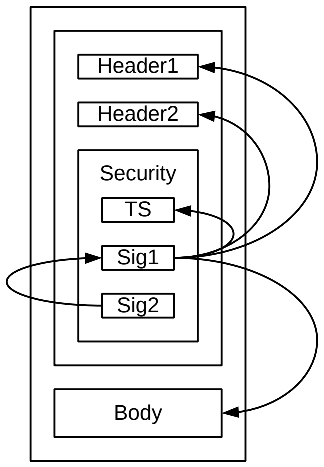
This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

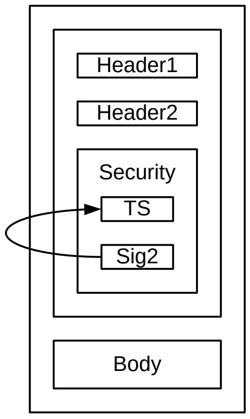
This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

8.3 EndorsingSupportingTokens Assertion

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



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



Syntax

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

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

/sp:EndorsingSupportingTokens

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

/sp:EndorsingSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

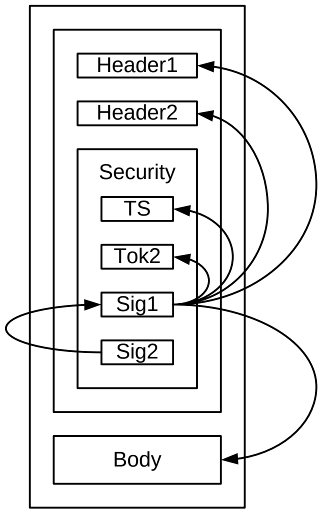
This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

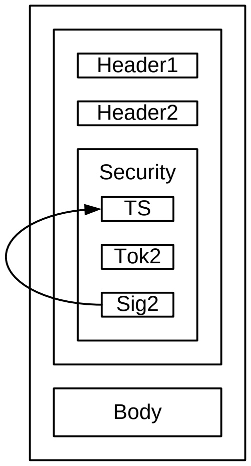
This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

8.4 SignedEndorsingSupportingTokens Assertion

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



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



Syntax

```
<sp:SignedEndorsingSupportingTokens 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:SignedEndorsingSupportingTokens>
```

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

/sp:SignedEndorsingSupportingTokens

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

/sp:SignedEndorsingSupportingTokens/wsp:Policy

> This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

> The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

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

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

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

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.

<u>**WSS: SOAP Message Security 1.0 Properties**</u>

[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 `wsse11:SignatureConfirmation` elements should be used as defined in WSS: Soap Message Security 1.1. If the value is 'true', `wsse11: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:

/sp:Wss10

This identifies a WSS10 assertion.

/sp:Wss10/wsp:Policy

This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

/sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier

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

/sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial

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

/sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI

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

/sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken

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

9.2 Wss11 Assertion

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

Syntax

```
<sp:Wss11 xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    <sp:MustSupportRefKeyIdentifier  ... /> ?
    <sp:MustSupportRefIssuerSerial  ... /> ?
    <sp:MustSupportRefExternalURI  ... /> ?
    <sp:MustSupportRefEmbeddedToken  ... /> ?
    <sp:MustSupportRefThumbprint  ... /> ?
    <sp:MustSupportRefEncryptedKey  ... /> ?
    <sp:RequireSignatureConfirmation  ... /> ?
    ...
  </wsp:Policy>
</sp:Wss11>
```

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

/sp:Wss11

This identifies an WSS11 assertion.

/sp:Wss11/wsp:Policy

This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

/sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier

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

/sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial

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

/sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI

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

/sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken

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

/sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint

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

/sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey

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

/sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation

This optional element is a policy assertion indicates that the [Signature Confirmation] property is 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].

<u>WS-Trust 1.3 Properties</u>

[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'.

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

This optional element is a policy assertion that indicates that the STS requires the requestor to 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 preferably 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 prefered 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 on the various token assertions. Five possible values are currently specified for the sp:IncludeToken attribute and implementations are expected to understand the meaning of all 5 values. If this information was encoded into the assertion QNames, each existing token assertion would require five variants, one for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

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

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

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)

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.3 Message Policy Subject Assertions

A.3.1 Supporting 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.3.2 Protection Assertions

SignedParts Assertion	(Section 4.1.1)
SignedElements Assertion	(Section 4.1.2)
EncryptedParts Assertion	(Section 4.2.1)
EncryptedElements Assertion	(Section 4.2.2)
ContentEncryptedElements Assertion	(Section 4.2.3)
RequiredElements Assertion	(Section 4.3.1)
RequiredParts Assertion	(Section 4.3.2)

A.4 Assertions With Undefined Policy Subject

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

A.4.1 General Assertions

AlgorithmSuite Assertion	(Section 7.1)
Layout Assertion	(Section 7.2)

A.4.2 Token Usage Assertions

See the nested assertions under the TransportBinding, SymmetricBinding and AssymetricBinding assertions.

A.4.3 Token Assertions

UsernameToken Assertion	(Section 5.3.1)

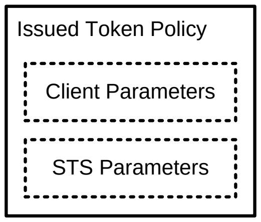
IssuedToken Assertion	(Section 5.3.2)
X509Token Assertion	(Section 5.3.3)
KerberosToken Assertion	(Section 5.3.4)
SpnegoContextToken Assertion	(Section 5.3.5)
SecurityContextToken Assertion	(Section 5.3.6)
SecureConversationToken Assertion	(Section 5.3.7)
SamlToken Assertion	(Section 5.3.8)
RelToken Assertion	(Section 5.3.9)
HttpsToken Assertion	(Section 5.3.10)

B. Issued Token Policy

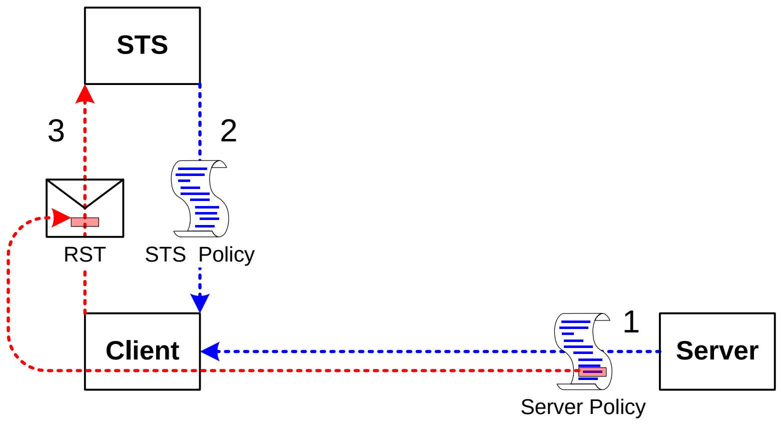
The section provides further detail about behavior associated with the IssuedToken assertion in section 5.3.2.

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.

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

Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [WS-Trust]. All items are optional, since the Server and STS may already have a pre-arranged relationship 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>
```

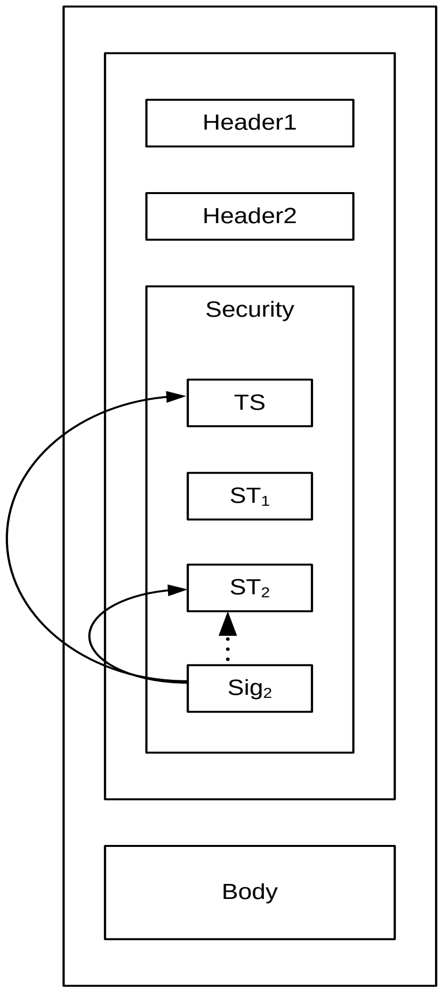
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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 Sig_2 indicate the parts signed by the supporting token labeled ST_2, namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST_2. 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

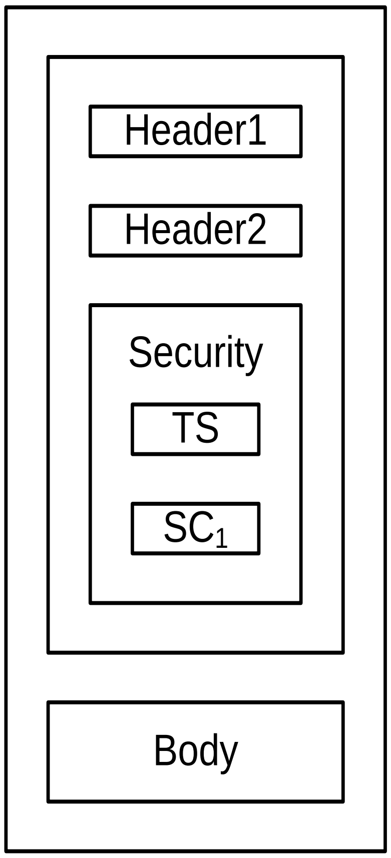
```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken 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:



The outer box shows that the entire message is protected (signed and encrypted) by the transport. One `wsse11:SignatureConfirmation` element labeled SC_1 corresponding to the signature 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.

Example:

Recipient to initiator message

```
<S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
      <wsse11:SignatureConfirmation Value="..." />
      ...
    </wsse:Security>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

C.2 Symmetric Binding

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>
```

```

<!-- Example Message Policy -->
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
  <sp:SignedParts>
    <sp:Header Name="Header1" Namespace="..." />
    <sp:Header Name="Header2" Namespace="..." />
    <sp:Body/>
  </sp:SignedParts>
  <sp:EncryptedParts>
    <sp:Header Name="Header2" Namespace="..." />
    <sp:Body/>
  </sp:EncryptedParts>
</wsp:Policy>
```

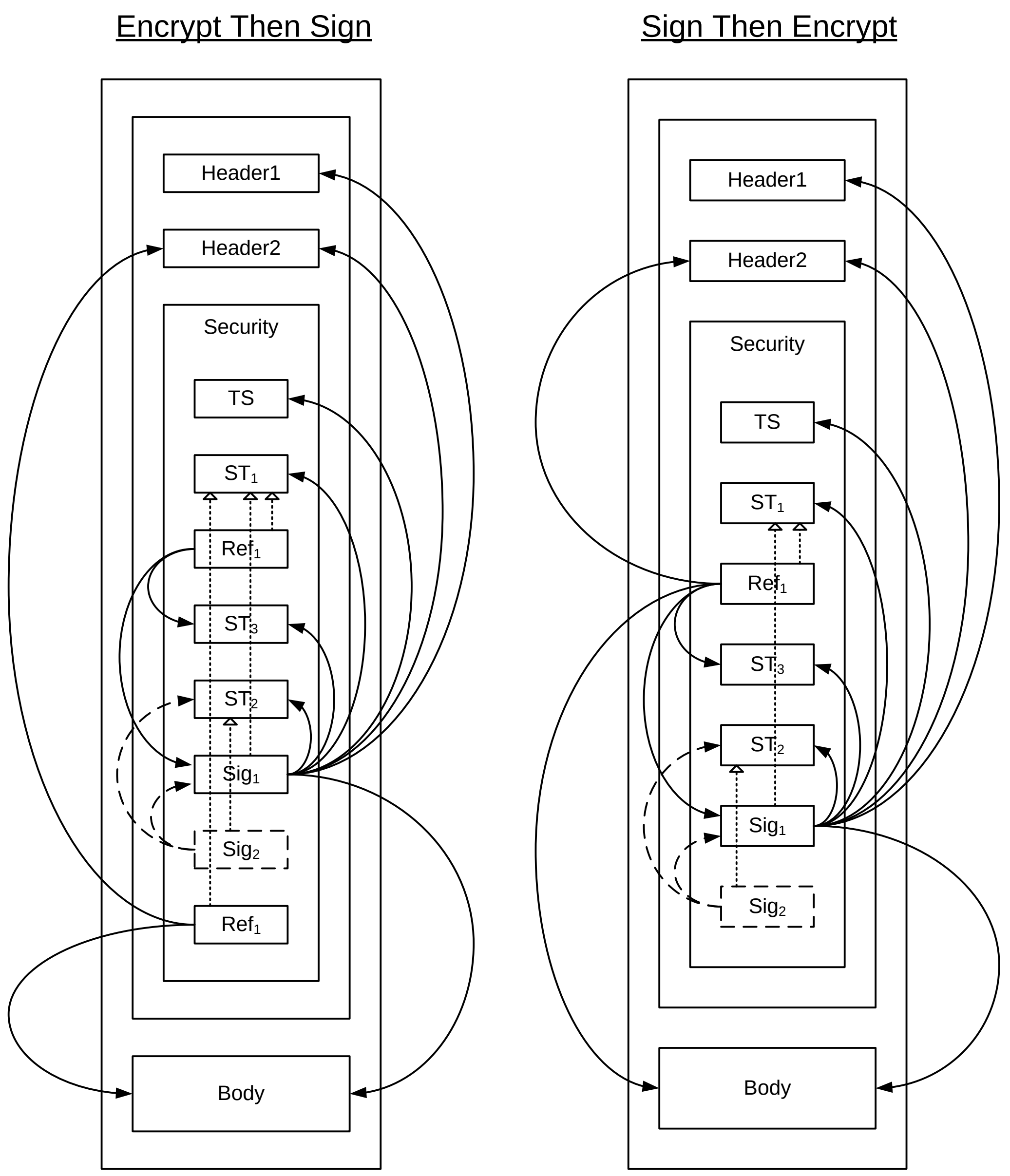
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.2.2 Initiator to Recipient Messages

Messages sent from initiator to recipient 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/Once or .../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. 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 3 above MUST be used, otherwise the key in the [Encryption Token].
5. Any tokens from the [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties whose `sp:IncludeToken` attribute is .../IncludeToken/Once or .../IncludeToken/Always.
6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken` attribute on the [Signature Token] is .../IncludeToken/Once or .../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 tokens from 5 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 cover the [Signature Token] regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token from 7 above MUST be used, otherwise the key in the [Signature Token] from 6 above.
9. Signatures covering the main signature from 8 above for any tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties. If [Token Protection] is 'true', the signature MUST also cover the endorsing token. If [Derived Keys] is 'true' and the endorsing token is associated with a symmetric key, then a Derived Key Token, based on the endorsing token, appears before the signature.
10. 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 token from 3 above MUST be used, otherwise the key in the [Encryption Token] from 2 above.

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



The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_1. The dashed arrows on the left from the box labeled Sig_2 indicate the parts signed by the supporting token labeled ST_2, namely the message signature labeled Sig_1 and the token used as the basis for the signature labeled ST_2. The arrows on the left from boxes labeled Ref_1 indicate references to parts encrypted using a key based on the Shared Secret Token labeled ST_1. The dotted arrows inside the box labeled Security indicate the token that was used as the basis for each cryptographic operation. 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 using EncryptBeforeSigning:

```
<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
  xmlns:wsse11="..." xmlns:wsse="..." xmlns:saml="..."
  xmlns:xenc="..." xmlns:ds="...">
  <S:Header>
    <x:Header1 wsu:Id="Header1" >
    ...
    </x:Header1>

    <wsse11:EncryptedHeader wsu:Id="enc_Header2">
      <!-- Plaintext Header2
      <x:Header2 wsu:Id="Header2" >
      ...
      </x:Header2>
      -->
      ...
    </wsse11:EncryptedHeader>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp">
        <wsu:Created>...</wsu:Created>
        <wsu:Expires>...</wsu:Expires>
      </wsu:Timestamp>
      <saml:Assertion AssertionId="_SharedSecretToken" ...>
      ...
      </saml:Assertion>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Signature" />
        <xenc:DataReference URI="#enc_SomeUsernameToken" />
        ...
      </xenc:ReferenceList>
      <xenc:EncryptedData ID="enc_SomeUsernameToken" >
        <!-- Plaintext UsernameToken
        <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
        ...
        </wsse:UsernameToken>
        -->
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_SharedSecretToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </xenc:EncryptedData>
      <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
      ...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedData ID="enc_Signature">
        <!-- Plaintext Signature
        <ds:Signature Id="Signature">
          <ds:SignedInfo>
            <ds:References>
              <ds:Reference URI="#Timestamp" >...</ds:Reference>
              <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
              <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
              <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
              <ds:Reference URI="#Header1" >...</ds:Reference>
              <ds:Reference URI="#Header2" >...</ds:Reference>
              <ds:Reference URI="#Body" >...</ds:Reference>
            </ds:References>
          </ds:SignedInfo>
          <ds:SignatureValue>...</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI="#_SharedSecretToken" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        -->
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_SharedSecretToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </xenc:EncryptedData>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:References>
            <ds:Reference URI="#Signature" >...</ds:Reference>
            <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
          </ds:References>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SomeSupportingToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Body" />
        <xenc:DataReference URI="#enc_Header2" />
        ...
      </xenc:ReferenceList>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="Body">
    <xenc:EncryptedData Id="enc_Body">
      ...
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#_SharedSecretToken" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

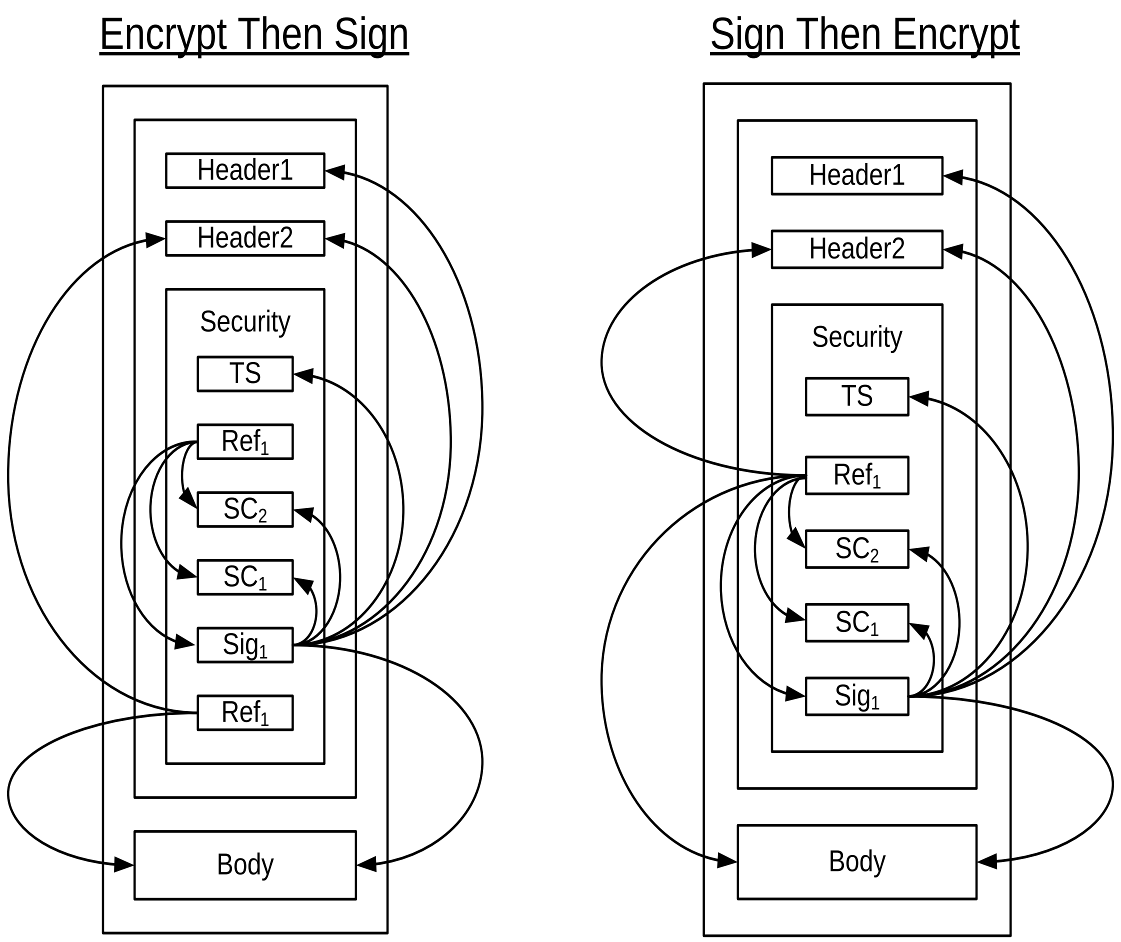
C.2.3 Recipient to Initiator Messages

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 `wsse11: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 `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.
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 `wsse11: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 Sig_1. The arrows on the left from boxes labeled Ref_1 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 `wsse11:SignatureConfirmation` elements labeled SC_1 and SC_2 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:

Recipient to initiator message using EncryptBeforeSigning:

```
<S:Envelope>
  <S:Header>
    <x:Header1 wsu:Id="Header1" >
    ...
    </x:Header1>
    <wsse11:EncryptedHeader wsu:Id="enc_Header2">
      <!-- Plaintext Header2
      <x:Header2 wsu:Id="Header2" >
      ...
      </x:Header2>
      -->
      ...
    </wsse11:EncryptedHeader>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp">
        <wsu:Created>...</wsu:Created>
        <wsu:Expires>...</wsu:Expires>
      </wsu:Timestamp>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Signature" />
        <xenc:DataReference URI="#enc_SigConf1" />
        <xenc:DataReference URI="#enc_SigConf2" />
        ...
      </xenc:ReferenceList>
      <xenc:EncryptedData ID="enc_SigConf1" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
        ...
        </wsse11:SignatureConfirmation>
        -->
      ...
      </xenc:EncryptedData>
      <xenc:EncryptedData ID="enc_SigConf2" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
        ...
        </wsse11:SignatureConfirmation>
        -->
      ...
      </xenc:EncryptedData>

      <xenc:EncryptedData Id="enc_Signature">
        <!-- Plaintext Signature
        <ds:Signature Id="Signature">
          <ds:SignedInfo>
            <ds:References>
              <ds:Reference URI="#Timestamp" >...</ds:Reference>
              <ds:Reference URI="#SigConf1" >...</ds:Reference>
              <ds:Reference URI="#SigConf2" >...</ds:Reference>
              <ds:Reference URI="#Header1" >...</ds:Reference>
              <ds:Reference URI="#Header2" >...</ds:Reference>
              <ds:Reference URI="#Body" >...</ds:Reference>
            </ds:References>
          </ds:SignedInfo>
          <ds:SignatureValue>...</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI="#_SomeIssuedToken" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        -->
        </xenc:EncryptedData>
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_SomeIssuedToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      <xenc:EncryptedData>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Body" />
        <xenc:DataReference URI="#enc_Header2" />
        ...
      </xenc:ReferenceList>
   </xenc:EncryptedData>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="Body">
    <xenc:EncryptedData Id="enc_Body">
      ...
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#_SomeIssuedToken" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

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>
```

```
<!-- Example Message Policy -->
<wsp:All xmlns:wsp="..." xmlns:sp="...">
  <sp:SignedParts>
    <sp:Header Name="Header1" Namespace="..." />
    <sp:Header Name="Header2" Namespace="..." />
    <sp:Body/>
  </sp:SignedParts>
  <sp:EncryptedParts>
    <sp:Header Name="Header2" Namespace="..." />
    <sp:Body/>
  </sp:EncryptedParts>
</wsp:All>
```

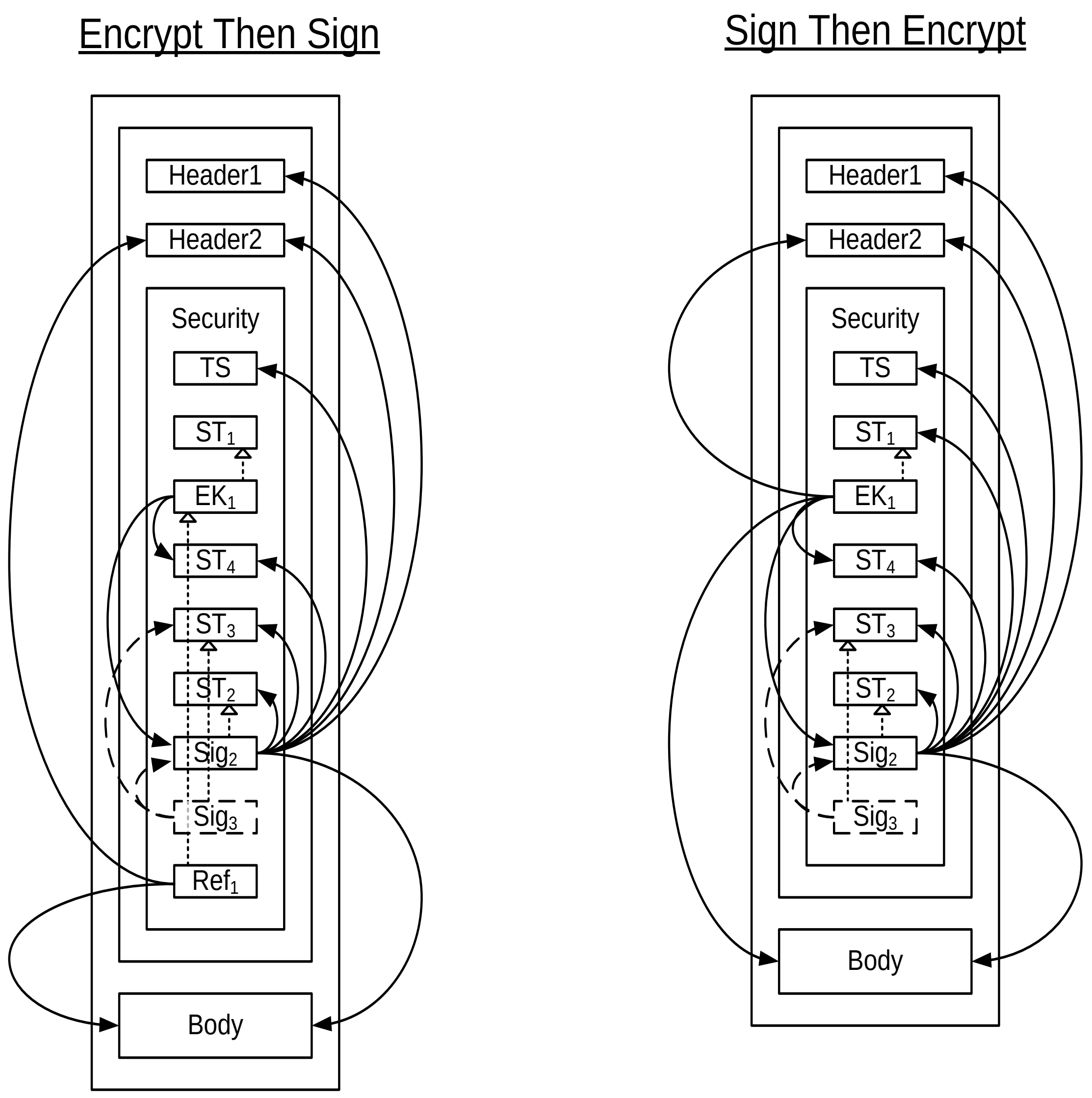
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

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.

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



The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_2 using the [Initiator Token] labeled ST_2. The dashed arrows on the left from the box labeled Sig_3 indicate the parts signed by the supporting token ST_3, namely the message signature Sig_2 and the token used as the basis for the signature labeled ST_3. The arrows on the left from boxes labeled EK_1 indicate references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on the left from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token used as the basis for each cryptographic operation. 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.

Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted key contains an external reference to the token containing the encryption key. The diagram illustrates how one might attach a security token related to the encrypted key for completeness. One possible use-case for this approach might be a stack which does not support the STR Dereferencing Transform, but wishes to include the encryption token in the message signature.

Initiator to recipient message *Example*

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

```
   xmlns:wsse11="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
  <S:Header>
    <x:Header1 wsu:Id="Header1" >
    ...
    </x:Header1>
    <wsse11:EncryptedHeader wsu:Id="enc_Header2">
      <!-- Plaintext Header2
      <x:Header2 wsu:Id="Header2" >
      ...
      </x:Header2>
      -->
      ...
    </wsse11:EncryptedHeader>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp">
        <wsu:Created>...</wsu:Created>
        <wsu:Expires>...</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
      ...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
        ...
        <xenc:ReferenceList>
          <xenc:DataReference URI="#enc_Signature" />
          <xenc:DataReference URI="#enc_SomeUsernameToken" />
          ...
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <xenc:EncryptedData ID="enc_SomeUsernameToken" >
        <!-- Plaintext UsernameToken
        <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
        ...
        </wsse:UsernameToken>
        -->
        ...
      </xenc:EncryptedData>
      <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
      ...
      </wsse:BinarySecurityToken>
      <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
      ...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedData ID="enc_Signature">
        <!-- Plaintext Signature
        <ds:Signature Id="Signature">
          <ds:SignedInfo>
            <ds:References>
              <ds:Reference URI="#Timestamp" >...</ds:Reference>
              <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
              <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
              <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
              <ds:Reference URI="#Header1" >...</ds:Reference>
              <ds:Reference URI="#Header2" >...</ds:Reference>
              <ds:Reference URI="#Body" >...</ds:Reference>
            </ds:References>
          </ds:SignedInfo>
          <ds:SignatureValue>...</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI="#InitiatorToken" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        -->
        ...
      </xenc:EncryptedData>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:References>
            <ds:Reference URI="#Signature" >...</ds:Reference>
            <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
          </ds:References>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SomeSupportingToken" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Body" />
        <xenc:DataReference URI="#enc_Header2" />
        ...
      </xenc:ReferenceList>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="Body">
    <xenc:EncryptedData Id="enc_Body">
      ...
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#RecipientEncryptedKey" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

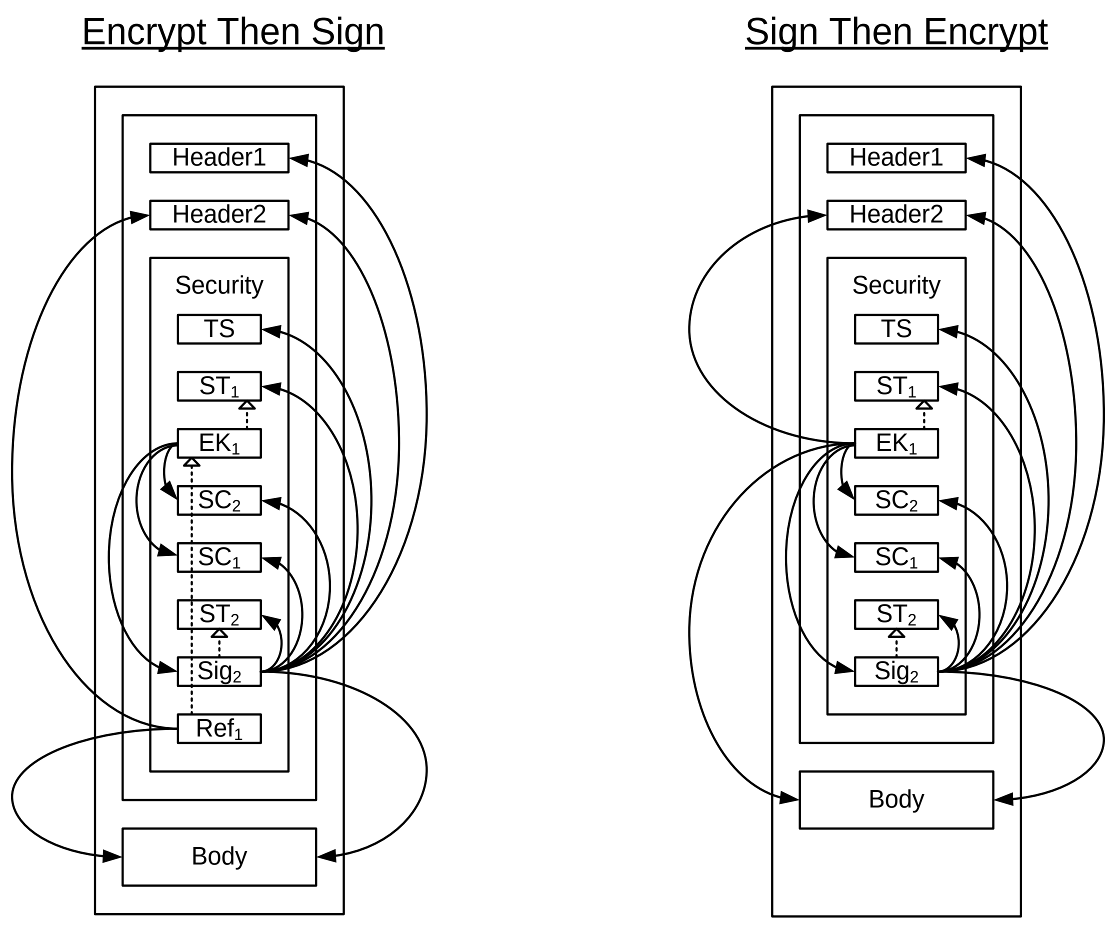
C.3.3 Recipient to Initiator Messages

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 `wsse11:SignatureConfirmation` elements from 4 below, if any.
4. If [Signature Confirmation] is '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.
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 `wsse11: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 Sig_2 using the [Recipient Token] labeled ST_2. The arrows on the left from boxes labeled EK_1 indicate references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on the left from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token used as the basis for each cryptographic operation. Two `wsse11:SignatureConfirmation` elements labeled SC_1 and SC_2 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:*

```
<S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
  xmlns:wsse11="..." xmlns:wsse="..."
  xmlns:xenc="..." xmlns:ds="...">
  <S:Header>
    <x:Header1 wsu:Id="Header1" >
    ...
    </x:Header1>
    <wsse11:EncryptedHeader wsu:Id="enc_Header2">
      <!-- Plaintext Header2
      <x:Header2 wsu:Id="Header2" >
      ...
      </x:Header2>
      -->
      ...
    </wsse11:EncryptedHeader>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp">
        <wsu:Created>...</wsu:Created>
        <wsu:Expires>...</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
      ...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
        ...
        <xenc:ReferenceList>
          <xenc:DataReference URI="#enc_Signature" />
          <xenc:DataReference URI="#enc_SigConf1" />
          <xenc:DataReference URI="#enc_SigConf2" />
          ...
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <xenc:EncryptedData ID="enc_SigConf2" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf2" ...>
        ...
        </wsse11:SignatureConfirmation>
        -->
        ...
      </xenc:EncryptedData>
      <xenc:EncryptedData ID="enc_SigConf1" >
        <!-- Plaintext SignatureConfirmation
        <wsse11:SignatureConfirmation wsu:Id="SigConf1" ...>
        ...
        </wsse11:SignatureConfirmation>
        -->
        ...
      </xenc:EncryptedData>
      <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
      ...
      </wsse:BinarySecurityToken>

      <xenc:EncryptedData ID="enc_Signature">
        <!-- Plaintext Signature
        <ds:Signature Id="Signature">
          <ds:SignedInfo>
            <ds:References>
              <ds:Reference URI="#Timestamp" >...</ds:Reference>
              <ds:Reference URI="#SigConf1" >...</ds:Reference>
              <ds:Reference URI="#SigConf2" >...</ds:Reference>
              <ds:Reference URI="#RecipientToken" >...</ds:Reference>
              <ds:Reference URI="#Header1" >...</ds:Reference>
              <ds:Reference URI="#Header2" >...</ds:Reference>
              <ds:Reference URI="#Body" >...</ds:Reference>
            </ds:References>
          </ds:SignedInfo>
          <ds:SignatureValue>...</ds:SignatureValue>
          <ds:KeyInfo>
            <wsse:SecurityTokenReference>
              <wsse:Reference URI="#RecipientToken" />
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        </ds:Signature>
        -->
        ...
      </xenc:EncryptedData>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#enc_Body" />
        <xenc:DataReference URI="#enc_Header2" />
        ...
      </xenc:ReferenceList>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="Body">
    <xenc:EncryptedData Id="enc_Body">
      ...
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#InitiatorEncryptedKey" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</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 `wsse11: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 `wsse11: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 intial 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

Lloyd Burch, Novell
Scott Cantor, Internet2
Greg Carpenter, Microsoft Corporation
Steve Carter, Novell
Symon Chang, BEA Systems, Inc.
Ching-Yun (C.Y.) Chao, IBM
Martin Chapman, Oracle Corporation
Kate Cherry, Lockheed Martin
Henry (Hyenvui) Chung, IBM
Luc Clement, Systinet Corp.
Paul Cotton, Microsoft Corporation
Glen Daniels, Sonic Software Corp.
Peter Davis, Neustar, Inc.
Martijn de Boer, SAP AG
Werner Dittmann, Siemens AG
Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
Fred Dushin, IONA Technologies
Petr Dvorak, Systinet Corp.
Colleen Evans, Microsoft Corporation
Ruchith Fernando, WSO2
Mark Fussell, Microsoft Corporation
Vijay Gajjala, Microsoft Corporation
Marc Goodner, Microsoft Corporation
Hans Granqvist, VeriSign
Martin Gudgin, Microsoft Corporation
Tony Gullotta, SOA Software Inc.
Jiandong Guo, Sun Microsystems
Phillip Hallam-Baker, VeriSign
Patrick Harding, Ping Identity Corporation
Heather Hinton, IBM
Frederick Hirsch, Nokia Corporation
Jeff Hodges, Neustar, Inc.
Will Hopkins, BEA Systems, Inc.
Alex Hristov, Otecia Incorporated
John Hughes, PA Consulting
Diane Jordan, IBM
Venugopal K, Sun Microsystems
Chris Kaler, Microsoft Corporation
Dana Kaufman, Forum Systems, Inc.
Paul Knight, Nortel Networks Limited
Ramanathan Krishnamurthy, IONA Technologies
Christopher Kurt, Microsoft Corporation
Kelvin Lawrence, IBM
Hubert Le Van Gong, Sun Microsystems
Jong Lee, BEA Systems, Inc.
Rich Levinson, Oracle Corporation
Tommy Lindberg, Dajeil Ltd.
Mark Little, JBoss Inc.
Hal Lockhart, BEA Systems, Inc.
Mike Lyons, Layer 7 Technologies Inc.
Eve Maler, Sun Microsystems
Ashok Malhotra, Oracle Corporation
Anand Mani, CrimsonLogic Pte Ltd
Jonathan Marsh, Microsoft Corporation
Robin Martherus, Oracle Corporation
Miko Matsumura, Infravio, Inc.
Gary McAfee, IBM

Michael McIntosh, IBM
John Merrells, Sxip Networks SRL
Jeff Mischkinsky, Oracle Corporation
Prateek Mishra, Oracle Corporation
Bob Morgan, Internet2
Vamsi Motukuru, Oracle Corporation
Raajmohan Na, EDS
Anthony Nadalin, IBM
Andrew Nash, Reactivity, Inc.
Eric Newcomer, IONA Technologies
Duane Nickull, Adobe Systems
Toshihiro Nishimura, Fujitsu Limited
Rob Philpott, RSA Security
Denis Pilipchuk, BEA Systems, Inc.
Darren Platt, Ping Identity Corporation
Martin Raepple, SAP AG
Nick Ragouzis, Enosis Group LLC
Prakash Reddy, CA
Alain Regnier, Ricoh Company, Ltd.
Irving Reid, Hewlett-Packard
Bruce Rich, IBM
Tom Rutt, Fujitsu Limited
Maneesh Sahu, Actional Corporation
Frank Siebenlist, Argonne National Laboratory
Joe Smith, Apani Networks
Davanum Srinivas, WSO2
Yakov Sverdlov, CA
Gene Thurston, AmberPoint
Victor Valle, IBM
Asir Vedamuthu, Microsoft Corporation
Greg Whitehead, Hewlett-Packard
Ron Williams, IBM
Corinna Witt, BEA Systems, Inc.
Kyle Young, Microsoft Corporation