Web Services Security Policy Language (WS-SecurityPolicy)

July 2005

Version 1.1

Authors

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

Copyright Notice

Abstract

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

Composable Architecture

By using the XML, SOAP and WSDL extensibility models, the WS* specifications are designed to be composed with each other to provide a rich Web services environment. WS-SecurityPolicy by itself does not provide a complete security solution for Web services. WS-SecurityPolicy is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models.

Status

This is a public consultation draft release of this specification for community evaluation and review. Feedback on this specification is handled through the WS-* Workshop process.

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, WS-Trust and WS-SecureConversation. 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 3.4, 11, 12, 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.

Table 1: Example security policy.

```
(01)<wsp:Policy>
(02)  <sp:SymmetricBinding>
(03)    <wsp:Policy>
(04)      <sp:ProtectionToken>
(05)        <wsp:Policy>
(06)          <sp:KerberosV5APREQToken
                  sp:IncludeToken=".../IncludeToken/Once" />
(07)        </wsp:Policy>
(08)      </sp:ProtectionToken>
(09)      <sp:SignBeforeEncrypting />
(10)      <sp:EncryptSignature />
```

```
(11)    </wsp:Policy>
(12)  </sp:SymmetricBinding>
(13)  <sp:SignedParts>
(14)    <sp:Body/>
(15)    <sp:Header
           Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
        />
(16)  </sp:SignedParts>
(17)  <sp:EncryptedParts>
(18)    <sp:Body/>
(19)  </sp:EncryptedParts>
(20)</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. Line 6 indicates that a Kerberos V5 APREQ token is to be used by both parties in a message exchange for protection. Line 9 indicates that signatures are generated over plaintext rather than ciphertext. Line 10 indicates that the signature over the signed messages parts is required to be encrypted. Lines 13-16 indicate which message parts are to be covered by the primary signature; in this case the soap:Body element, indicated by Line 14 and any SOAP headers in the WS-Addressing namespace, indicated by line 15. Lines 17-19 indicate which message parts are to be encrypted; in this case just the soap:Body element, indicated by Line 18.

2. Terminology and Notation

2.1 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.

2.2 Namespaces

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

```
http://schemas.xmlsoap.org/ws/2005/07/securitypolicy
```

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/	[SOAP11]
ds	http://www.w3.org/2000/09/xmldsig#	[XMLDSIG]
enc	http://www.w3.org/2001/04/xmlenc#	[XMLENC]
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/2005/xx/oasis-2005xx-wss-wssecurity-secext-1.1.xsd	[WSS11]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	[WS-Policy], [WS-PolicyAttachment]
xsd	http://www.w3.org/2001/XMLSchema	[XMLSchema Part1], [XMLSchema Part2]
wst	http://schemas.xmlsoap.org/ws/2005/02/trust	[WS-Trust]
wsc	http://schemas.xmlsoap.org/ws/2005/02/sc	[WS-SecureConversation]
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	[WS-Addressing]
sp	http://schemas.xmlsoap.org/ws/2005/07/securitypolicy	This specification

2.3 Notational Conventions

The keywords "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 messages:

- 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.

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 or timestamp 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.

2.4 Schema Files

A normative copy of the XML Schema [XML Schema Part 1, Part 2] description for this specification can be retrieved from the following address:

`http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.xsd`

2.5 Compliance

Normative text within this specification takes precedence over outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2], which in turn take precedence over examples.

Sections 3.4, 11, 12, and all Appendices are not normative.

3. 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, 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.

3.1 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.

3.2 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. To enable this set of functionality, this specification introduces a mechanism for nesting policy assertions underneath other assertions. This mechanism is described in Section 4.

3.3 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
- Any necessary key transfer mechanisms
- Any required message elements (e.g. timestamps)
- The content and ordering of elements in the `wsse:Security` header. Elements not specified in the binding are not allowed.
- How correlation of messages is performed securely (if applicable to the message pattern)

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.

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

- TransportBinding
- SymmetricBinding
- AsymmetricBinding

4. Policy Considerations

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

4.1 Nested Policy

The WS-Policy specification defines a mechanism for describing capabilities and requirements as assertions. These assertions are contained within one of wsp:Policy, wsp:All, or wsp:ExactlyOne. The WS-Policy specification defines the nesting semantics associated with the wsp:Policy, wsp:All and wsp:ExactlyOne. However these semantics do not allow individual assertions to specify that the child elements contained within the assertion should also be evaluated as assertions.

The following section is an overview of the nesting semantics of WS-Policy elements.

4.1.1 Nesting Policy Elements

To determine whether two assertions are "compatible", the QName value, that is the Name and Namespace of one assertion element is compared to the QName value of another assertion. If they match, then they are compatible.

A wsp:Policy element may contain one or more assertions. To determine whether two wsp:Policy elements are "compatible", each assertion in one wsp:Policy element is compared, as described above, to the assertions in another wsp:Policy element. If all assertions from each wsp:Policy element are matched exactly, they are compatible.

To enable richer sets of options to be expressed, WS-Policy defines the wsp:All and wsp:ExactlyOne elements. These elements may be placed as immediate children of a wsp:Policy element. In addition, these two elements may also appear under themselves. This allows for a policy to describe alternative options within policy. Let's say that a policy wishes to express requirements for A and (B or C). This could be described as two policy statements:

```
<wsp:Policy>
  <A />
  <B />
</wsp:Policy>
<wsp:Policy>
  <A />
  <C />
</wsp:Policy>
```

Alternatively, we can use the wsp:All and wsp:ExactlyOne elements to describe the alternative policy in a single wsp:Policy element:

```
<wsp:Policy>
  <wsp:All>
    <A />
    <wsp:ExactlyOne>
      <B />
      <C />
    </wsp:ExactlyOne>
  </wsp:All>
</wsp:Policy>
```

This process is described in more detail in the WS-Policy specification.

4.1.2 Nested Policy Assertions

Some assertions may need to declare that additional assertions, scoped only to that assertion, further qualify the behavior and compatibility semantics of that assertion. Whereas the wsp:All and wsp:ExactlyOne elements describe requirements and alternatives of a wsp:Policy element, nested assertions describe requirements and alternatives for the enclosing assertion element. To enable these semantics, this specification defines some assertions such that they have a single wsp:Policy child element which in turn contains assertions which qualify the behavior of the enclosing assertion. Two such assertions are compatible if they have the same QName AND their nested policy expressions (if any) are compatible.

For example, let's say that a policy wishes to express requirements for A and B, and furthermore that B requires C and D. The normalized policy statement would look like:

```
<wsp:Policy>
  <A />
  <B>
    <wsp:Policy>
      <C />
      <D />
    </wsp:Policy>
  </B>
</wsp:Policy>
```

The policy above is fully normalized. Policy normalization DOES NOT promote nested assertions to the outer scope.

The wsp:Policy element allows any assertion as content. However, assertions defined in this specification that allow nested policy will typically constrain the content of that nested policy.

Note: To enable automatic intersection of nested policy assertions, policy engines will need to be modified to scan the contents of assertions to determine whether intersection is required. This approach is being investigated by the WS-Policy working group to formalize the notion of nested policy and to define processing behavior requirements for nested policy. Additionally, an attribute may be defined to advertise to a policy engine that scanning is required on a particular assertion. For example:

```
<wsp:Policy>
  <A />
  <B x:ContainsPolicy="true">
    <wsp:Policy>
      ...
    </wsp:Policy>
  </B>
</wsp:Policy>
```

Ideally the x:ContainsPolicy attribute will, at some point, be moved in the WS-Policy namespace.

4.1.3 Nesting Policy Processing Rules

This section provides rules for processing nested policy based on the informal description above;

1. Assertions MUST specify whether or not they contain nested policy.
2. Assertions SHOULD specify which other assertions can be present in their nested policy.
3. Nested assertions need to be specifically designed for nesting inside one or more outer assertions. Assertions SHOULD specify which assertions they can be nested within.
4. Assertions from one domain SHOULD NOT be nested inside assertions from another domain. For example, assertions from a transaction domain should not be nested inside an assertion from a security domain.
5. Assertions containing nested policy are normalized recursively such that in the normal form each nested policy contains no choices. Thus each outer assertion that contains nested policy containing choices is duplicated such that there are as many instances of the outer assertion as there are choices in the nested policy, one instance for each nested choice, recursively. See Section 4.1.4 for a worked example of normalization.
6. Nested policies are intersected in their own processing contexts with the corresponding nested policy in a matching outer assertion. Thus two assertions having nested policy intersect if the outer assertion QName matches and the nested policies intersect. Intersection always occurs using the normalized form. See Section 4.1.5 for a worked example of intersection.
7. An assertion with an empty nested policy does not intersect with the same assertion without nested policy.

4.1.4 Nested Policy Normalization Worked Example

This section shows a worked example of normalizing assertions with nested policy.

Policy 1

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <A />
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
           <wsp:All>
              <D/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
```

```
  </wsp:ExactlyOne>
</wsp:Policy>
```

The above policy is normalized by, in this case, creating two alternatives, both containing an A assertion and a B assertion. One alternative contains a B assertion with a nested C assertion while the other contains a B assertion with a nested D assertion;

Normalized form

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <A/>
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
    <wsp:All>
      <A/>
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <D/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

4.1.5 Nested Policy Intersection Worked Example

This section shows a worked example of computing the intersection of two policies that contain assertions with nested policy.

Policy 1

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
```

```
      <A />
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
    <wsp:All>
      <A />
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <D/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Policy 2

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <A />
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
    <wsp:All>
      <A />
```

```
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <E/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

The two policies above, which are already in normal form, are intersected as follows; firstly the QNames of the A and B assertions are intersected then the QNames of the nested assertions inside the B assertions are intersected. In the nested case, only the two B assertions that have nested C assertions match. Thus the intersection of the nested policy is;

Intersected policy

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsp:All>
      <A/>
      <A/>
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
      <B>
        <wsp:Policy>
          <wsp:ExactlyOne>
            <wsp:All>
              <C/>
            </wsp:All>
          </wsp:ExactlyOne>
        </wsp:Policy>
      </B>
    </wsp:All>
```

```
  </wsp:ExactlyOne>
</wsp:Policy>
```

4.2 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.

[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

5. 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]. 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 7.3).

5.1 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.

5.1.1 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 details 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 ... >
  <sp:Body />?
  <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
  ...
</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 (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.

/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.

5.1.2 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 details 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"? ... >
  <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. If no child elements are specified, all message headers targeted at the UltimateReceiver role and the body of the message MUST be integrity protected.

/sp:SignedElements/@XPathVersion

This optional attribute contains a URI which indicates the version of XPath to use.

/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 the signature.

5.2 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.

5.2.1 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 details 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 ... >
  <sp:Body/>?
  <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
  ...
</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 headers cannot be encrypted 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.

5.2.2 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 details 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"? ... >
  <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. If no child elements are specified, the body of the message MUST be confidentiality protected.

/sp:EncryptedElements/@XPathVersion

This optional attribute contains a URI which indicates the version of XPath to use.

/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.

5.3 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.

5.3.1 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"? ... >
  <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.

/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.

6. 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.

6.1 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.

6.1.1 Token Inclusion Values

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

http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/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://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Once	The token MUST be included in only one message sent from initiator to 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://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient	The token MUST be included in all messages sent from initator to recipient. The token MUST NOT be include in messages sent from the

	recipient to the initiator.
http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/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://schemas.xmlsoap.org/ws/2005/07/securitypolicy/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.

6.2 Token Properties

6.2.1 [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'.

6.3 Token Assertions

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

6.3.1 UsernameToken Assertion

This element represents a requirement to include a username token. The default version of this token is the wsse:UsernameToken as defined in [WSS: Username Token Profile 1.0].

Syntax

```
<sp:UsernameToken sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    (
      <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/wsp:Policy
: This optional element identifies additional requirements for use of the sp:UsernameToken assertion.

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

This optional element indicates that a Username token should be used as defined in [WSS: Username Token Profile 1.0].

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

This optional element indicates that a Username token should be used as defined in [WSS: Username Token Profile 1.1].

Note: While Username tokens could be used cryptographically, such usage is discouraged in general because of the relatively low entropy typically associated with passwords. This specification does not define a cryptographic binding for the Username token. A new token assertion could be defined to allow for cryptographic binding.

6.3.2 IssuedToken Assertion

This element represents a requirement for an issued token, that 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"? ... >
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
  <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
    ...
  </sp:RequestSecurityTokenTemplate>
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <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:RequestSecurityTokenTemplate

This required element contains elements which MUST be copied into the 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 URI identifying the version of WS-Trust referenced by the contents of this element.

/sp:IssuedToken/wsp:Policy
This optional element identifies additional requirements for use of the sp:IssuedToken assertion.

/sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys
This optional element sets the [Derived Keys] property for this token to 'true'.

/sp:IssuedToken/wsp:Policy/sp:RequireInternalReference
This optional element 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 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.

6.3.3 X509Token Assertion

This element represents a requirement for a binary security token carrying an X509 token. The default version of this token and associated profile is the X509 Version 3 token as specified in [WSS: X509 Certificate Token Profile 1.0].

Syntax

```
<sp:X509Token sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    <sp:RequireKeyIdentifierReference ... /> ?
    <sp:RequireIssuerSerialReference ... /> ?
    <sp:RequireEmbeddedTokenReference ... /> ?
    <sp:RequireThumbprintReference ... /> ?
    (
      <sp:WssX509V1Token10 ... /> |
      <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/wsp:Policy
: This optional element identifies additional requirements for use of the sp:X509Token assertion.

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

/sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference
: This optional element indicates that an issuer serial reference is required when referencing this token.

/sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference
: This optional element indicates that an embedded token reference is required when referencing this token.

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

/sp:X509Token/wsp:Policy/sp:WssX509V1Token10
: This optional element indicates that an X509 Version 1 token should be used as defined in [WSS: X509 Token Profile 1.0].

/sp:X509Token/wsp:Policy/sp:WssX509V3Token10
: This optional element indicates that an X509 Version 3 token should be used as defined in [WSS: X509 Token Profile 1.0].

/sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10
: This optional element indicates that an X509 PKCS7 token should be used as defined in [WSS: X509 Token Profile 1.0].

/sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10
: This optional element indicates that an X509 PKI Path Version 1 token should be used as defined in [WSS: X509 Token Profile 1.0].

/sp:X509Token/wsp:Policy/sp:WssX509V1Token11
: This optional element indicates that an X509 Version 1 token should be used as defined in [WSS: X509 Token Profile 1.1].

/sp:X509Token/wsp:Policy/sp:WssX509V3Token11
: This optional element indicates that an X509 Version 3 token should be used as defined in [WSS: X509 Token Profile 1.1].

/sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11
: This optional element indicates that an X509 PKCS7 token should be used as defined in [WSS: X509 Token Profile 1.1].

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

This optional element indicates that an X509 PKI Path Version 1 token should be used as defined in [WSS: X509 Token Profile 1.1].

6.3.4 KerberosToken Assertion

This element represents a requirement for a Kerberos token. The default version of this token and associated profile is the Kerberos Version 5 AP-REQ security token as specified in [WSS: Kerberos Token Profile 1.0].

Syntax

```
<sp:KerberosToken sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <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/wsp:Policy
This optional element identifies additional requirements for use of the sp:KerberosToken assertion.

/sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys
This optional element sets the [Derived Keys] property for this token to 'true'.

/sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference
This optional element indicates that a key identifier reference is required when referencing this token.

/sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11
This optional element indicates that a Kerberos Version 5 AP-REQ token should be used as defined in [WSS: Kerberos Token Profile 1.1].

/sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11
This optional element indicates that a GSS Kerberos Version 5 AP-REQ token should be used as defined in [WSS: Kerberos Token Profile 1.1].

6.3.5 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"? ... >
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> ?
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    ...
  </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/wsp:Policy

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

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

This optional element sets the [Derived Keys] property for this token to 'true'.

6.3.6 SecurityContextToken Assertion

This element represents a requirement for a SecurityContextToken token. The default version of this token is the Security Context Token as specified in [WS-SecureConversation 1.0].

Syntax

```
<sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <sp:RequireExternalUriReference ... /> ?
    <sp:SC10SecurityContextToken ... /> ?
    ...
  </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/wsp:Policy

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

/sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys
This optional element sets the [Derived Keys] property for this token to 'true'.

/sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference
This optional element indicates that an external URI reference is required when referencing this token.

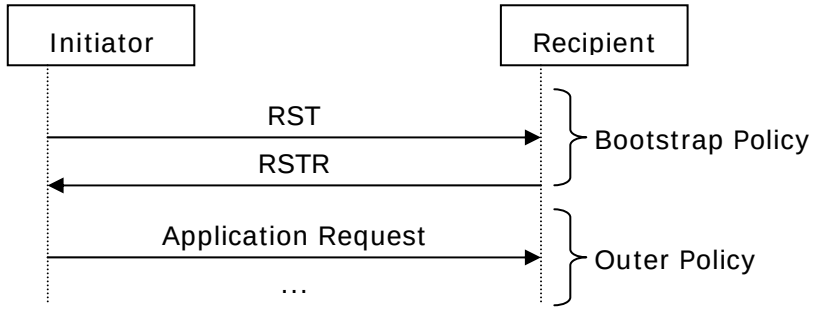
/sp:SecurityContextToken/wsp:Policy/sp:SC10SecurityContextToken
This optional element indicates that a Security Context Token should be used as defined in [WS-SecureConversation 1.0].

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.

6.3.7 SecureConversationToken Assertion

This element represents a requirement for a Security Context Token retrieved from the indicated issuer address. The default version of this token and associated protocol is the Security Context Token as defined in [WS-SecureConversation 1.0]. 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"? ... >
  <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer>?
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <sp:RequireExternalUriReference ... /> ?
    <sp:SC10SecurityContextToken ... /> ?
```

```
    <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/wsp:Policy

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

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

This optional element sets the [Derived Keys] property for this token to 'true'.

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

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

/sp:SecureConversationToken/wsp:Policy/sp:SC10SecurityContextToken

This optional element indicates that a Security Context Token should be used as obtained using the protocol defined in [WS-SecureConversation 1.0].

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

This optional element 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.

Example

```
<wsp:Policy>
  <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>
```

6.3.8 SamlToken Assertion

This element represents a requirement for a SAML token. The default version of this token and associated profile is SAML Version 1.0 token as described in the [WSS: SAML Token Profile].

Syntax

```
<sp:SamlToken sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <sp:RequireKeyIdentifierReference ... /> ?
    (
      <sp:WssSamlV10Token10 ... /> |
```

```
      <sp:WssSamlV11Token10 ... /> |
      <sp:WssSamlV10Token11 ... /> |
      <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/wsp:Policy

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

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

This optional element sets the [Derived Keys] property for this token to 'true'.

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

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

/sp:SamlToken/wsp:Policy/sp:WssSamlV10Token10

This optional element identifies that a SAML Version 1.0 token should be used as defined in [WSS: SAML Token Profile 1.0].

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

This optional element identifies that a SAML Version 1.1 token should be used as defined in [WSS: SAML Token Profile 1.0].

/sp:SamlToken/wsp:Policy/sp:WssSamlV10Token11

This optional element identifies that a SAML Version 1.0 token should be used as defined in [WSS: SAML Token Profile 1.1].

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

This optional element identifies that a SAML Version 1.1 token should be used as defined in [WSS: SAML Token Profile 1.1].

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

This optional element identifies that a SAML Version 2.0 token should be used as defined in [WSS: SAML Token Profile 1.1].

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.

6.3.9 RelToken Assertion

This element represents a requirement for a REL token. The default version of this token and associate profile is the REL Version 1.0 token as described in the [WSS: REL Token Profile].

Syntax

```
<sp:RelToken sp:IncludeToken="xs:anyURI"? ... >
  <wsp:Policy>
    <sp:RequireDerivedKeys ... /> ?
    <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/wsp:Policy

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

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

This optional element sets the [Derived Keys] property for this token to 'true'.

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

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

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

This optional element identifies that a REL Version 1.0 token should be used as defined in [WSS: REL Token Profile 1.0].

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

This optional element identifies that a REL Version 2.0 token should be used as defined in [WSS: REL Token Profile 1.0].

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

This optional element identifies that a REL Version 1.0 token should be used as defined in [WSS: REL Token Profile 1.1].

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

This optional element identifies that a REL Version 2.0 token should be used as defined in [WSS: REL Token Profile 1.1].

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.

6.3.10 HttpsToken Assertion

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

Syntax

```
<sp:HttpsToken RequireClientCertificate="xs:boolean" ... >
  <wsp:Policy>
    ...
  </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/@RequireClientCertificate
: The client MUST provide a certificate when negotiating the HTTPS session.

/sp:HttpsToken/wsp:Policy
: This optional element identifies additional requirements for use of the sp:HttpsToken assertion.

7. 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 8. Assertions that define values for these properties are defined in Section 8. The following properties are used by the security binding assertions.

7.1 [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 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.

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-aes256
KwAes192	http://www.w3.org/2001/04/xmlenc#kw-aes192
KwAes256	http://www.w3.org/2001/04/xmlenc#kw-aes128
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://schemas.xmlsoap.org/ws/2005/02/sc/dk/p_sha1
PSha1L128	http://schemas.xmlsoap.org/ws/2005/02/sc/dk/p_sha1
PSha1L192	http://schemas.xmlsoap.org/ws/2005/02/sc/dk/p_sha1
PSha1L256	http://schemas.xmlsoap.org/ws/2005/02/sc/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

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 KS]	HmacSha1
[Asym KS]	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]	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
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

7.2 [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'.

7.3 [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.
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'.

7.4 [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. 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'.

7.5 [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'.

7.6 [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'.

7.7 [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
LaxTimestampLast	As Lax except that the last item in the security header MUST be a wsse:Timestamp

The default value of this property is 'Lax'.

7.7.1 Strict Layout Rules

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 earlier element in the security header.
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. For example, an encrypted primary signature MUST occur before any supporting token signature per 2c 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). An `xenc:EncryptedKey` element with an internal reference list MUST additionally obey rule (4).

Examples of these layout rules for each security binding are described in Appendix C.

8. 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.

8.1 AlgorithmSuite Assertion

This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite] property described in Section 7.1. The scope of this assertion is defined by its containing assertion.

Syntax

```
<sp:AlgorithmSuite ... >
  <wsp:Policy>
   (<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 ... /> ?
    ...
  </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 element contains one or more policy assertions that indicate the specific algorithm suite to use.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This assertion indicates that the [C14N] property of an algorithm suite is set to 'C14N'.

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

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

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

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

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

This assertion indicates that the [XPath] property is set to 'XPath'.

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

This assertion indicates that the [XPath] property is set to 'XPath20'.

8.2 Layout Assertion

This assertion indicates a requirement for a particular security header layout as defined under the [Security Header Layout] property described in Section 7.7. The scope of this assertion is defined by its containing assertion.

Syntax

```
<sp:Layout ... >
  <wsp:Policy>
    <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 element contains one or more policy assertions that indicate the specific security header layout to use.

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

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

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

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

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

This assertion indicates that the [Security Header Layout] property is set to 'LaxTimestampFirst'.

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

This assertion indicates that the [Security Header Layout] property is set to 'LaxTimestampLast'.

8.3 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 ... >
  <wsp:Policy>
    <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 assertion 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 assertion indicates a value that populates the [Algorithm Suite] property. See Section 8.1 for more details.

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

This assertion indicates a value that populates the [Security Header Layout] property. See Section 8.2 for more details.

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

This assertion indicates that the [Timestamp] property is set to 'true'.

8.4 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 MUST apply to [Endpoint Policy Subject].

Syntax

```
<sp:SymmetricBinding ... >
  <wsp:Policy>
    (
      <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 assertion 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 one or more tokens to use for encryption.

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

This assertion 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 one or more tokens to use for signatures.

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

This assertion 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 assertion indicates a value that populates the [Algorithm Suite] property. See Section 8.1 for more details.

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

This assertion indicates a value that populates the [Security Header Layout] property. See Section 8.1 for more details.

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

This assertion indicates that the [Timestamp] property is set to 'true'.

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

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

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

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

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

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

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

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

8.5 AsymmetricBinding Assertion

The AsymmetricBinding 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; [Initiator Token] and [Recipient Token]. If the message pattern requires multiple messages, this binding defines that the [Initiator Token] is used for the message signature from initiator to the recipient, and for encryption from recipient to initiator. The [Recipient Token] is used for encryption from initiator to recipient, and for the message signature from recipient to initiator. This assertion MUST apply to [Endpoint Policy Subject].

Syntax

```
<sp:AsymmetricBinding ... >
  <wsp:Policy>
    <sp:InitiatorToken>
      <wsp:Policy> ... </wsp:Policy>
    </sp:InitiatorToken>
    <sp:RecipientToken>
      <wsp:Policy> ... </wsp:Policy>
    </sp:RecipientToken>
    <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 assertion indicates a requirement for an Initiator Token. The specified token populates the [Initiator Token] property 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:RecipientToken

This assertion indicates a requirement for a Recipient Token. The specified token populates the [Recipient 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:AlgorithmSuite

This assertion indicates a value that populates the [Algorithm Suite] property. See Section 8.1 for more details.

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

This assertion indicates a value that populates the [Security Header Layout] property. See Section 8.2 for more details.

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

This assertion indicates that the [Timestamp] property is set to 'true'.

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

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

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

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

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

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

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

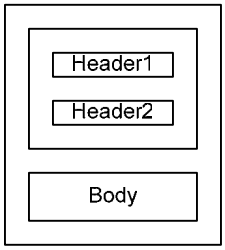
This assertion indicates that the [Entire Header And Body Signatures] property is set to 'true'.

9. Supporting Tokens

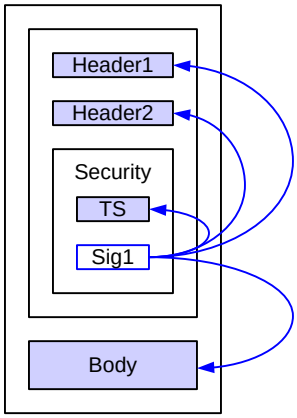
Security Bindings use tokens to secure the message exchange. The Security Binding will require one to create a signature using the token identified in the Security Binding policy. This signature will here-to-fore be referred to as the “message signature”. Additional tokens may be specified to augment the claims provided by the token associated with the “message signature” provided by the Security Binding. This section defines four properties related to supporting token requirements which may be referenced by a Security Binding: [Supporting Tokens], [Signed Supporting Tokens], [Endorsing Supporting Tokens] and [Signed Endorsing Supporting Tokens]. Four assertions are defined to populate those properties: SupportingTokens, SignedSupportingTokens, EndorsingSupportingTokens, and SignedEndorsingSupportingTokens. 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 include all tokens from the outer scope and any additional tokens for a specific message from the inner scope.

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.

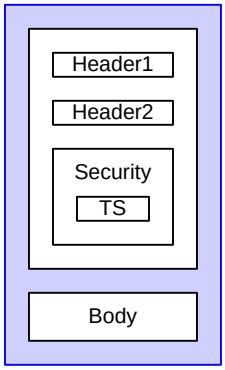


Each binding requires that the message is signed using a token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts of the message including the message timestamp (TS) facilitate replay detection. The signature is then included as part of the Security header 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:



9.1 SupportingTokens Assertion

Supporting tokens are included in the security header and may optionally include additional message parts to sign and/or encrypt.

Syntax

```
<sp:SupportingTokens ... >
  <wsp:Policy>
    [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 follows the schema outlined in Section 8.1 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 follows the schema outlined in Section 5.1.1 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 follows the schema outlined in Section 5.1.2 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

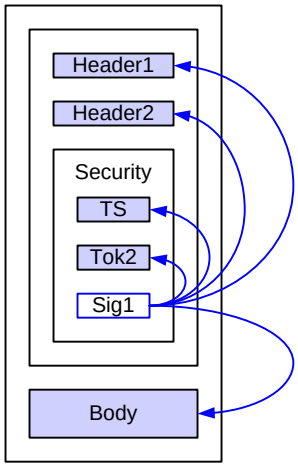
This optional element follows the schema outlined in Section 5.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

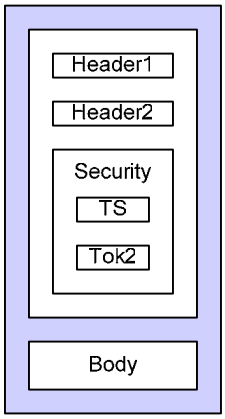
This optional element follows the schema outlined in Section 5.2.1 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

9.2 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 ... >
  <wsp:Policy>
    [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 follows the schema outlined in Section 8.1 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 follows the schema outlined in Section 5.1.1 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 follows the schema outlined in Section 5.1.2 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

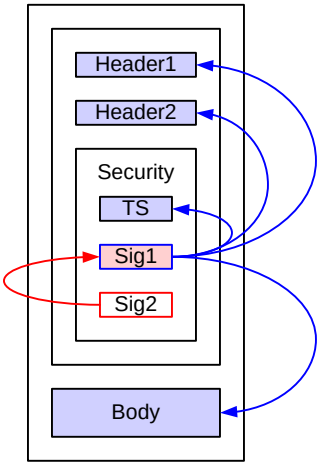
This optional element follows the schema outlined in Section 5.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

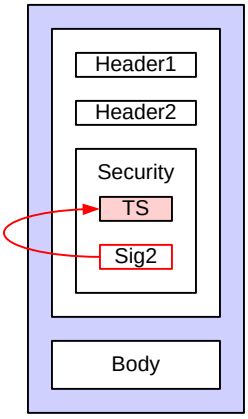
This optional element follows the schema outlined in Section 5.2.1 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

9.3 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) should cover the message timestamp as illustrated below:



Syntax

```
<sp:EndorsingSupportingTokens ... >
  <wsp:Policy>
    [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 follows the schema outlined in Section 8.1 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 follows the schema outlined in Section 5.1.1 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 follows the schema outlined in Section 5.1.2 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

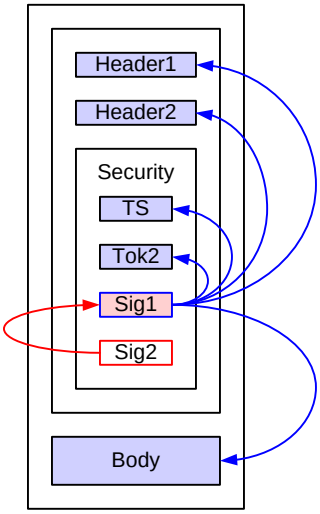
This optional element follows the schema outlined in Section 5.2.1 and describes additional message parts that MUST be encrypted using the token identified by this policy assertion.

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

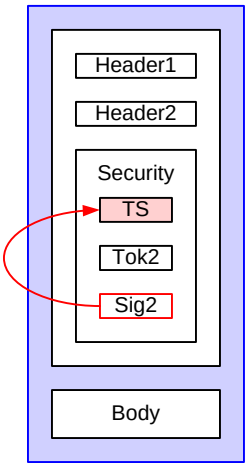
This optional element follows the schema outlined in Section 5.2.1 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

9.4 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 ... >
  <wsp:Policy>
    [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 follows the schema outlined in Section 8.1 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 follows the schema outlined in Section 5.1.1 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 5.1.2 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 follows the schema outlined in Section 5.2.1 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 follows the schema outlined in Section 5.2.1 and describes additional message elements that MUST be encrypted using the token identified by this policy assertion.

9.5 Example

Example policy containing supporting token assertions.

```
<!-- Example Endpoint Policy -->
<wsp:Policy>
  <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:SignedTokens>
        <wsp:Policy>
          <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:SignedTokens>
      <sp:SignedEndorsingTokens>
        <wsp:Policy>
          <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:SignedEndorsingTokens>
    </wsp:Policy>
  </sp:SymmetricBinding>
  ...
</wsp:Policy>
```

The sp:SignedTokens 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:SignedEndorsingTokens assertion indicates that an X509 certificate must be include 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 include in the security header..

10. 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.

WSS: SOAP Message Security 1.0 Properties

[Direct References]

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

[Key Identifier References]

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

[Issuer Serial References]

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

[External URI References]

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

[Embedded Token References]

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

WSS: SOAP Message Security 1.1 Properties

[Thumbprint References]

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

[EncryptedKey References]

This boolean property indicates whether the initiator and recipient MUST be able to process references using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to process such references. A value of 'false'

indicates that the initiator and recipient MUST NOT generate such references and that the initiator and recipient MAY send a fault if such references are encountered. This property has a default value of 'false'.

[Signature Confirmation]

This boolean property specifies whether `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. 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'.

10.1 Wss10 Assertion

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

Syntax

```
<sp:Wss10 ... >
  <wsp:Policy>
    <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 assertion indicates that the [Key Identifier References] property is set to 'true'.

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

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

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

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

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

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

10.2 Wss11 Assertion

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

Syntax

```
< sp:Wss11 ... >
  <wsp:Policy>
    <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 assertion indicates that the [Key Identifier References] property is set to 'true'.

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

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

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

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

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

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

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

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

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

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

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

This assertion indicates that the [Signature Confirmation] property is set to 'true'.

11. 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 7. The assertions defined here MUST apply to [Endpoint Policy Subject].

WS-Trust 1.0 Properties

[Client Challenge]

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

[Server Challenge]

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

[Client Entropy]

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

[Server Entropy]

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

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

[Issued Tokens]

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

11.1 Trust10 Assertion

The Trust10 assertion allows you to specify which WS-Trust 1.0 options are supported.

Syntax

```
<sp:Trust10 ... >
  <wsp:Policy>
    <sp:MustSupportClientChallenge  ... />?
    <sp:MustSupportServerChallenge  ... />?
    <sp:RequireClientEntropy  ... />?
    <sp:RequireServerEntropy  ... />?
    <sp:MustSupportIssuedTokens  ... />?
    ...
```

```
  </wsp:Policy>
  ...
</sp:Trust10 ... >
```

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

/sp:Trust10

This identifies a Trust10 assertion.

/sp:Trust10/wsp:Policy

This indicates a policy that controls WS-Trust 1.0 options.

/sp:Trust10/wsp:Policy/sp:MustSupportClientChallenge

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

/sp:Trust10/wsp:Policy/sp:MustSupportServerChallenge

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

/sp:Trust10/wsp:Policy/sp:RequireClientEntropy

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

/sp:Trust10/wsp:Policy/sp:RequireServerEntropy

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

/sp:Trust10/wsp:Policy/sp:MustSupportIssuedTokens

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

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 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.

13. Acknowledgements

We would like to thank the following people for their contributions towards this specification:

Vaithialingam B. Balayoghan, Microsoft

Francisco Curbera, IBM

Christopher Ferris, IBM

Cedric Fournet, Microsoft

Andy Gordon, Microsoft

Tomasz Janczuk, Microsoft

David Melgar, IBM

Bruce Rich, IBM

Jeffrey Schlimmer, Microsoft

Chris Sharp, IBM

Kent Tamura, IBM

T.R. Vishwanath, Microsoft

Elliot Waingold, Microsoft

14. References

The following are normative references

[KEYWORDS]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

[RFC2068]
IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997

[SOAP11]
W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

[SOAP12]
W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework," 24 June 2003.

[XMLSchema Part1]
W3C Recommendation, "XML Schema Part 1: Structure Second Edition," 28 October 2004.

[XMLSchema Part2]
W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition," 28 October 2004.

[URI]
T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.

The following are non-normative references

[WS-Policy]
S.Bajaj, et.al.,"Web Services Policy Framework (WS-Policy)," September 2004

[WS-PolicyAttachment]
S.Bajaj, et.al.,"Web Services Policy Attachment (WS-PolicyAttachment)," September 2004

[WS-Trust]
S.Anderson, et.al.,"Web Services Trust Language (WS-Trust)," February 2005

[WS-SecureConversation]
S.Anderson, et.al.,"Web Services Secure Conversation Language (WS-SecureConversation)," February 2005

[WS-Addressing]
D.Box, et.al.,"Web Services Addressing Language (WS-Addressing)," August 2004

[WSS10]
A. Nadalin, et.al., "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)," OASIS Standard 200401, March 2004

[WSS:UsernameToken 1.0]
A. Nadalin, et.al., "Web Services Security: UsernameToken Profile 1.0," OASIS Standard 200401, March 2004

[WSS:X509Token]
P. Hallam-Baker, et.al., "Web Services Security X.509 Certificate Token Profile," OASIS Standard 200401, March 2004

[WSS:Kerberos Token 1.0]
TBD - update

[XMLDSIG]
D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. *XML-Signature Syntax and Processing*, W3C Recommendation, 12 February 2002. http://www.w3.org/TR/xmldsig-core/.

[XMLENC]
W3C Recommendation, "XML Encryption Syntax and Processing," 10 December 2002.

Appendix 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.

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

TransportBinding Assertion	(Section 8.3)
SymmetricBinding Assertion	(Section 8.4)
AsymmetricBinding Assertion	(Section 8.5)

A.1.3 Token Assertions

SupportingTokens Assertion	(Section 9.1)
SignedSupportingTokens Assertion	(Section 9.2)
EndorsingSupportingTokens Assertion	(Section 9.3)
SignedEndorsingSupportingTokens Assertion	(Section 9.4)

A.1.4 WSS: SOAP Message Security 1.0 Assertions

Wss10 Assertion	(Section 10.1)

A.1.5 WSS: SOAP Message Security 1.1 Assertions

Wss11 Assertion	(Section 10.2)

A.1.6 Trust 1.0 Assertions

Trust10 Assertion	(Section 11.1)

A.2 Operation Policy Subject Assertions

A.2.1 Supporting Token Assertions

SupportingTokens Assertion	(Section 9.1)
SignedSupportingTokens Assertion	(Section 9.2)
EndorsingSupportingTokens Assertion	(Section 9.3)
SignedEndorsingSupportingTokens Assertion	(Section 9.4)

A.3 Message Policy Subject Assertions

A.3.1 Supporting Token Assertions

SupportingTokens Assertion	(Section 9.1)
SignedSupportingTokens Assertion	(Section 9.2)
EndorsingSupportingTokens Assertion	(Section 9.3)
SignedEndorsingSupportingTokens Assertion	(Section 9.4)

A.3.2 Protection Assertions

SignedParts Assertion	(Section 5.1.1)
SignedElements Assertion	(Section 5.1.2)
EncryptedParts Assertion	(Section 5.2.1)
EncryptedElements Assertion	(Section 5.2.2)
RequiredElements Assertion	(Section 5.3.1)

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.

A.4.1 General Assertions

AlgorithmSuite Assertion	(Section 8.1)
Layout Assertion	(Section 8.2)
IncludeTimestamp Assertion	(Section 8.3)
IncludeTimestamp Assertion	(Section 8.4)
EncryptBeforeSigning Assertion	(Section 8.4)
EncryptSignature Assertion	(Section 8.4)
ProtectTokens Assertion	(Section 8.4)
OnlySignEntireHeadersAndBody Assertion	(Section 8.4)
IncludeTimestamp Assertion	(Section 8.5)
EncryptBeforeSigning Assertion	(Section 8.5)
EncryptSignature Assertion	(Section 8.5)
ProtectTokens Assertion	(Section 8.5)
OnlySignEntireHeadersAndBody Assertion	(Section 8.5)

A.4.2 Token Usage Assertions

TransportToken Assertion	(Section 8.3)
EncryptionToken Assertion	(Section 8.4)

SignatureToken Assertion	(Section 8.4)
ProtectionToken Assertion	(Section 8.4)
InitiatorToken Assertion	(Section 8.5)
RecipientToken Assertion	(Section 8.5)

A.4.3 Token Assertions

UsernameToken Assertion	(Section 6.3.1)
IssuedToken Assertion	(Section 6.3.2)
X509Token Assertion	(Section 6.3.3)
KerberosToken Assertion	(Section 6.3.4)
SpnegoContextToken Assertion	(Section 6.3.5)
SecurityContextToken Assertion	(Section 6.3.6)
SecureConversationToken Assertion	(Section 6.3.7)
SamlToken Assertion	(Section 6.3.8)
RelToken Assertion	(Section 6.3.9)
HttpsToken Assertion	(Section 6.3.10)

A.4.4 WSS: SOAP Message Security 1.0 Assertions

MustSupportRefKeyIdentifier Assertion	(Section 10.1)
MustSupportRefIssuerSerial Assertion	(Section 10.1)
MustSupportRefExternalUri Assertion	(Section 10.1)
MustSupportRefEmbeddedToken Assertion	(Section 10.1)

A.4.5 WSS: SOAP Message Security 1.1 Assertions

MustSupportRefKeyIdentifier Assertion	(Section 10.2)
MustSupportRefIssuerSerial Assertion	(Section 10.2)
MustSupportRefExternalUri Assertion	(Section 10.2)
MustSupportRefEmbeddedToken Assertion	(Section 10.2)
MustSupportRefThumbprint Assertion	(Section 10.2)
RequireSignatureConfirmation Assertion	(Section 10.2)

A.4.6 Trust 1.0 Assertions

MustSupportClientChallenge Assertion	(Section 11.1)
MustSupportServerChallenge Assertion	(Section 11.1)
RequireClientEntropy Assertion	(Section 11.1)
RequireServerEntropy Assertion	(Section 11.1)
MustSupportIssuedTokens Assertion	(Section 11.1)

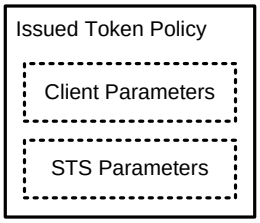
Appendix B – Issued Token Policy

The section provides further detail about behavior associated with the IssuedToken assertion in section 6.2.2.

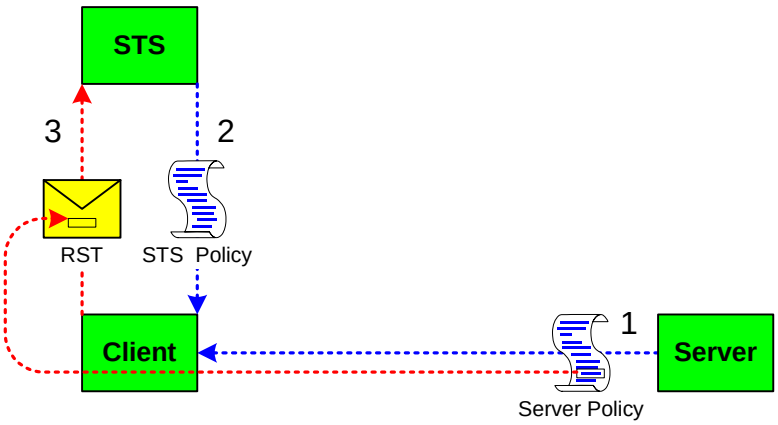
The issued token security model involves a three-party setup. There's a target Server, a Client, and a trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from STS to Client. Policy may be embedded inside an Issued

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 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 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.

Appendix 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 7.7.

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>
  <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 />
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <sp:SignedEndorsingSupportingTokens>
        <wsp:Policy>
          <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
```

```
      </sp:SignedEndorsingSupportingTokens>
    </wsp:Policy>
  </sp:TransportBinding>
  <sp:Wss11>
    <sp:RequireSignatureConfirmation />
  </sp:Wss11>
</wsp:Policy>
```

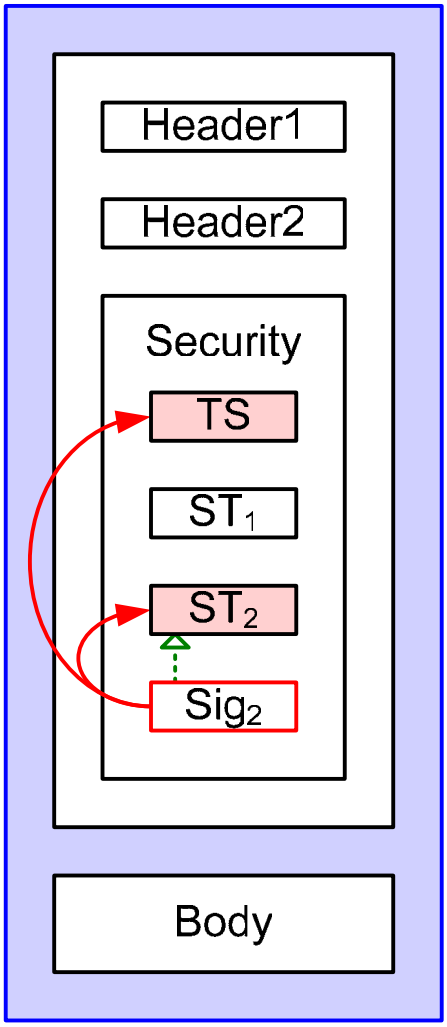
This policy is used as the basis for the examples shown in the subsequent section describing the security header layout for this binding.

C.1.2 Initiator to Recipient Messages

Messages sent from initator 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 blue outer box shows that the entire message is protected (signed and encrypted) by the transport. The red arrows (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 green 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>
  <S:Header>
    ...
    <wsse:Security>
      <wsu:Timestamp wsu:Id="timestamp">
        <wsu:Created>[datetime]</wsu:Created>
        <wsu:Expires>[datetime]</wsu:Expires>
      </wsu:Timestamp>
```

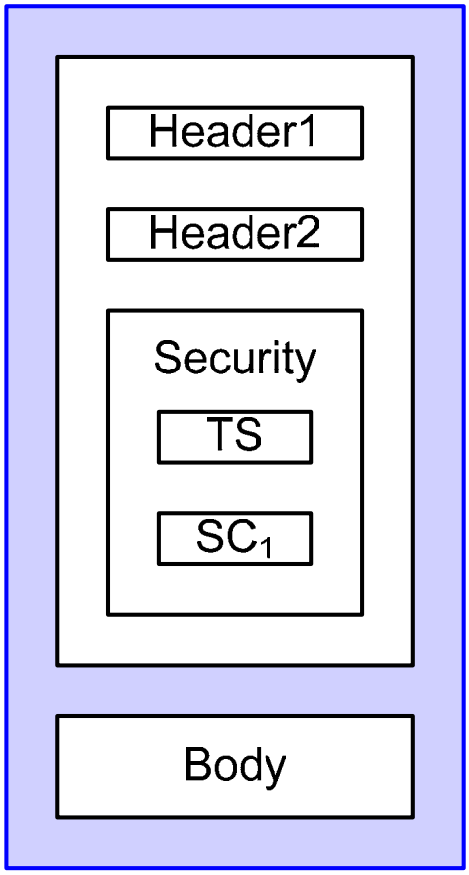
```
      <wsse:UsernameToken wsu:Id='SomeSignedToken' >
      ...
      </wsse:UsernameToken>
      <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
      ...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:References>
            <ds:Reference URI="#timestamp" />
            <ds:Reference URI="#SomeSignedEndorsingToken" />
          </ds:References>
        </ds:SignedInfo>
        <ds:Signature>...</ds:Signature>
        <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 blue 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>
  <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>
  <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 />
      <sp:SignedSupportingTokens>
```

```
        <wsp:Policy>
          <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <sp:SignedEndorsingSupportingTokens>
        <wsp:Policy>
          <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:SignedEndorsingSupportingTokens>
    </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:Wss11>
    <wsp:Policy>
      <sp:RequireSignatureConfirmation />
    </wsp:Policy>
  </sp:Wss11>
</wsp:Policy>

<!-- Example Message Policy -->
<wsp:Policy>
  <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

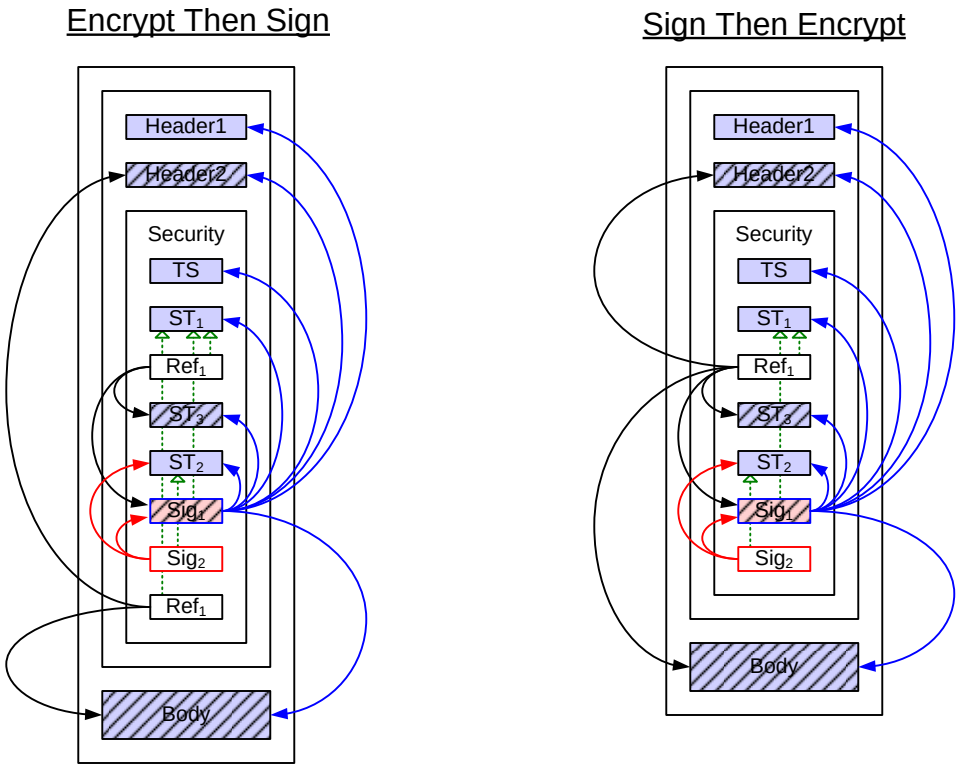
Messages sent from initiator to recipient have the following layout for the security header:

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 blue arrows (right) indicate parts that were signed as part of the message signature labeled Sig_1. The red arrows (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 black arrows (left) from boxes labeled Ref_1 indicate references to parts encrypted using a key based on the Shared Secret Token labeled ST_1. The green dotted arrows 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>
  <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:Signature>...</ds:Signature>
        <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:Signature>...</ds:Signature>
      <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>
  <xenc:EncryptedData Id="enc_Body">
    ...
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#_SharedSecretToken" />
```

```
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

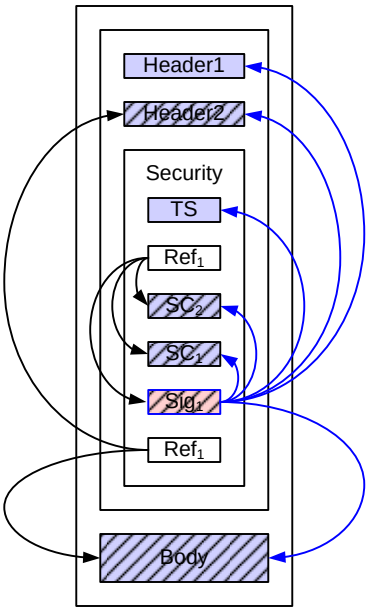
C.2.3 Recipient to Initiator Messages

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

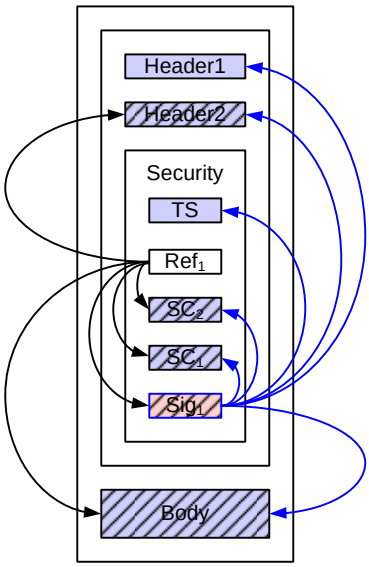
1. A `wsu:Timestamp` element if [Timestamp] is 'true'.
2. If the `sp:IncludeToken` attribute on the [Encryption Token] is .../IncludeToken/Always, then the [Encryption Token].
3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This Derived Key Token is used for 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:

Encrypt Then Sign



Sign Then Encrypt



The blue arrows (right) indicate parts that were signed as part of the message signature labeled Sig_1. The black arrows (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:Signature>...</ds:Signature>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_SomeIssuedToken" />
          </wsse:SecurityTokenReference>
```

```
          </ds:KeyInfo>
        </ds:Signature>
        -->
        ...
        <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>
    </wsse:Security>
  </S:Header>
  <S: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>
```

```
<sp:AsymmetricBinding>
  <wsp:Policy>
    <sp:RecipientToken>
      <wsp:Policy>
        <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Always" />
      </wsp:Policy>
   </sp:RecipientToken>
   <sp:InitiatorToken>
     <wsp:Policy>
       <sp:X509V3Token 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 />
   <sp:SignedSupportingTokens>
     <wsp:Policy>
       <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
     </wsp:Policy>
   </sp:SignedSupportingTokens>
   <sp:SignedEndorsingSupportingTokens>
     <wsp:Policy>
       <sp:X509V3Token sp:IncludeToken=".../IncludeToken/Once" />
     </wsp:Policy>
   </sp:SignedEndorsingSupportingTokens>
  </wsp:Policy>
</sp:AsymmetricBinding>
<sp:Wss11>
  <wsp:Policy>
    <sp:RequireSignatureConfirmation />
  </wsp:Policy>
</sp:Wss11>
```

```
</wsp:Policy>

<!-- Example Message Policy -->
<wsp:All>
  <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

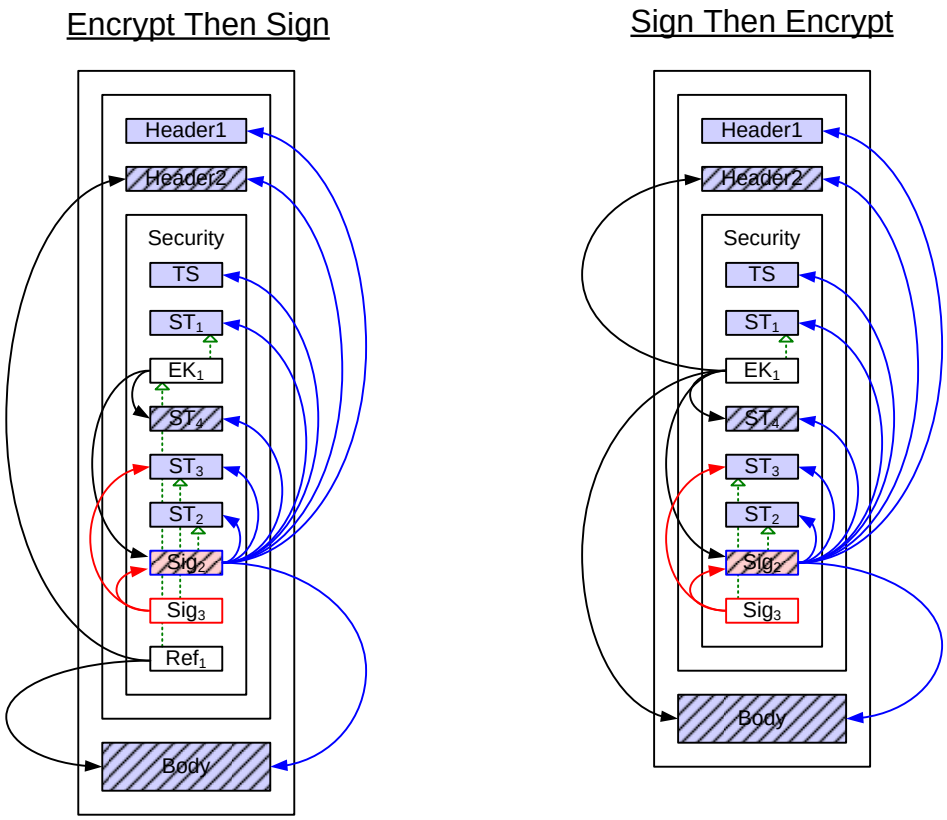
Messages sent from initiator to recipient have the following layout:

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 then an xenc:EncryptedKey element, containing a key encrypted for the recipient. If [Protection Order] is 'SignBeforeEncrypting' then 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 [Signed Supporting Tokens] and [Signed Endorsing Supporting Tokens] properties 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 a reference list including 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 blue arrows (right) indicate parts that were signed as part of the message signature labeled Sig_2 using the [Initiator Token] labeled ST_2. The red arrows (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 black arrows (left) from boxes labeled EK_1 indicate references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The black arrows (left) from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the encrypted key labeled EK_1. The green dotted arrows 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.

Example

Initiator to recipient message

```
<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>
      <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="#RecipientToken" >...</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:Signature>...</ds:Signature>
        <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:Signature>...</ds:Signature>
      <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>
    <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

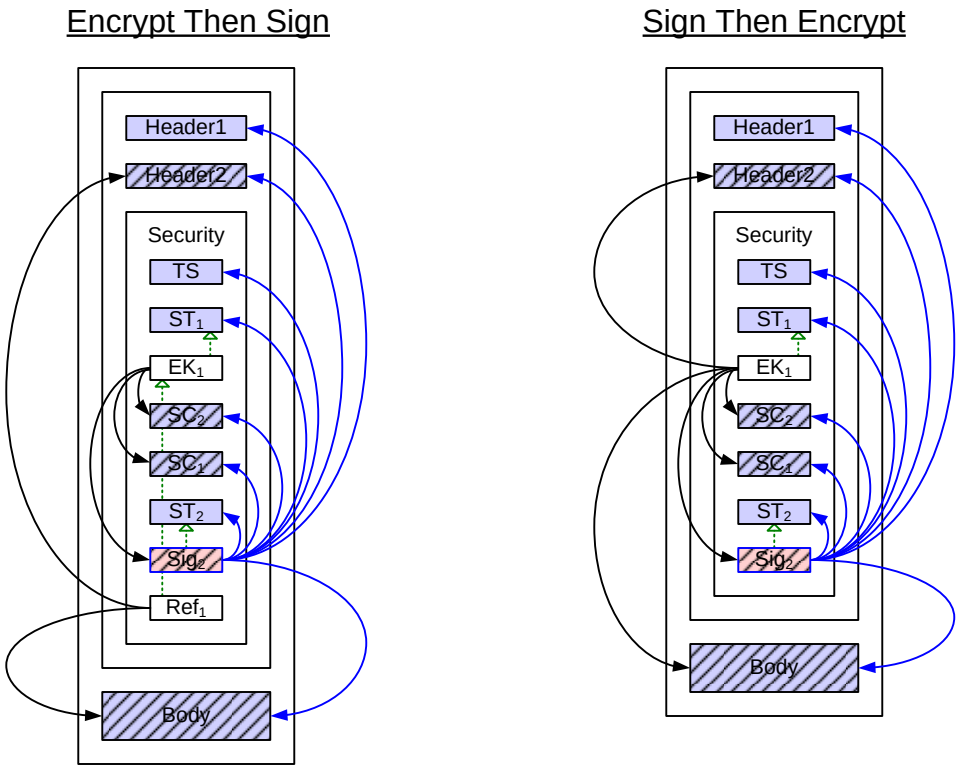
Messages sent from recipient to initiator have the following layout:

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 then an xenc:EncryptedKey element, containing a key encrypted for the initiator. If [Protection Order] is 'SignBeforeEncrypting' then 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' and

the [Initiator Token] is specified, then the signature MUST also cover the [Initiator Token].

7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then a reference list including 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 blue arrows (right) indicate parts that were signed as part of the message signature labeled Sig_2 using the [Recipient Token] labeled ST_2. The black arrows (left) from boxes labeled EK_1 indicate references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The black arrows (left) from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the encrypted key labeled EK_1. The green dotted arrows 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.

Example:

Recipient to initiator message

```
<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>
  <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="#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:Signature>...</ds:Signature>
        <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>
  <xenc:EncryptedData Id="enc_Body">
    ...
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#InitiatorEncryptedKey" />
```

```
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```