WS-SecurityPolicy 1.2

Committee Draft 02

7 March 2007

Specification URIs:

This Version:

> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-cd-02.doc
> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-cd-02.pdf
> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-cd-02.html

Previous Version:

> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2-spec-cd-01.doc
> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2-spec-cd-01.pdf
> http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2-spec-cd-01.html

Latest Version:

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

Artifact Type:

> specification

Technical Committee:

> OASIS Web Services Secure Exchange TC

Chair(s):

> Kelvin Lawrence, IBM
> Chris Kaler, Microsoft

Editor(s):

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

Related work:

> N/A

Declared XML Namespace(s):

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

Abstract:

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

Status:

> This document was last revised or approved by the WS-SX TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.
>
> Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the

"Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/ws-sx.

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

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

Notices

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

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

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

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

1.1 Example

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

Table 1: Example security policy.

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

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

49

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

1.2 Namespaces

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

64   `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702~~512~~`

65

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

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

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

wsc	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512	[WS-SecureConversation]
wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702512	This specification

1.3 Schema Files

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

```
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702512/ws-securitypolicy-1.2.xsd
```

1.4 Terminology

Policy - A collection of policy alternatives.

Policy Alternative - A collection of policy assertions.

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

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

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

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

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

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

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

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

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

Supporting Token - A token used to provide additional claims.

1.4.1 Notational Conventions

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

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

- The syntax appears as an XML instance, but values in italics indicate data types instead of literal
  values.
- Characters are appended to elements and attributes to indicate cardinality:
    - "?" (0 or 1)
    - "*" (0 or more)
    - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "(" and ")" are used to indicate that contained items are to be treated as a group
  with respect to cardinality or choice.
- The characters "[" and "]" are used to call out references and property names.

141 • Ellipses (i.e., "...") indicate points of extensibility. Additional children and/or attributes MAY be
142 added at the indicated extension points but MUST NOT contradict the semantics of the parent
143 and/or owner, respectively. By default, if a receiver does not recognize an extension, the receiver
144 SHOULD ignore the extension; exceptions to this processing rule, if any, are clearly indicated
145 below.

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

148

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

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

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

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

158 In this document reference is made to the wsu:Id attribute and the wsu:Created and wsu:Expires
159 elements in a utility schema (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
160 1.0.xsd). The wsu:Id attribute and the wsu:Created and wsu:Expires elements were added to the
161 utility schema with the intent that other specifications requiring such an ID type attribute or timestamp
162 element could reference it (as is done here).

163

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

1.5 Normative References

170 [RFC2119]   S. Bradner, "Key words for use in RFCs to Indicate Requirement
171 Levels", RFC 2119, Harvard University, March 1997.
172 http://www.ietf.org/rfc/rfc2119.txt

173

174 [SOAP]   W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000.
175 http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

176

177 [SOAP12]   W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24
178 June 2003.
179 http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

180

181 [SOAPNorm]   W3C Working Group Note, "SOAP Version 1.2 Message
182 Normalization", 8 October 2003.
183 http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/

184

185 186 187	[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, Day Software, Adobe Systems, January 2005.
188		http://www.ietf.org/rfc/rfc3986.txt
189		
190 191	[RFC2068]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997
192		http://www.ietf.org/rfc/rfc2068.txt
193		
194	[RFC2246]	IETF Standard, "The TLS Protocol", January 1999.
195		http://www.ietf.org/rfc/rfc2246.txt
196		
197	[SwA]	W3C Note, "SOAP Messages with Attachments", 11 December 2000
198		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211
199		
200 201	[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006.
202		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
203		
204 205	[WS-Policy]	W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006.
206		http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
207 208		W3C Candidate Recommendation "Web Services Policy 1.5 – Framework", 28 February 2007
209		http://www.w3.org/TR/2007/CR-ws-policy-framework-20070228/
210		
211 212	[WS-PolicyAttachment]	W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006.
213 214		http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
215 216		W3C Candidate Recommendation "Web Services Policy 1.5 – Attachment", 28 February 2007
217		http://www.w3.org/TR/2007/CR-ws-policy-attach-20070228/
218		
219	[WS-Trust]	OASIS Committee Draft, "WS-Trust 1.3", September 2006
220		http://docs.oasis-open.org/ws-sx/ws-trust/200512
221		
222 223	[WS-SecureConversation]	OASIS Committee Draft, "WS-SecureConversation 1.3", September 2006
224		http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
225		
226 227	[WSS10]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004.
228 229		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
230		

231	[WSS11]	OASIS Standard, "OASIS Web Services Security: SOAP Message
232		Security 1.1 (WS-Security 2004)", February 2006.
233		http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-
234		spec-os-SOAPMessageSecurity.pdf
235		
236	[WSS:UsernameToken1.0]	OASIS Standard, "Web Services Security: UsernameToken Profile",
237		March 2004
238		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
239		token-profile-1.0.pdf
240		
241	[WSS:UsernameToken1.1]	OASIS Standard, "Web Services Security: UsernameToken Profile
242		1.1", February 2006
243		http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-
244		spec-os-UsernameTokenProfile.pdf
245		
246	[WSS:X509Token1.0]	OASIS Standard, "Web Services Security X.509 Certificate Token
247		Profile", March 2004
248		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
249		profile-1.0.pdf
250		
251	[WSS:X509Token1.1]	OASIS Standard, "Web Services Security X.509 Certificate Token
252		Profile", February 2006
253		http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-
254		spec-os-x509TokenProfile.pdf
255		
256	[WSS:KerberosToken1.1]	OASIS Standard, "Web Services Security Kerberos Token Profile 1.1",
257		February 2006
258		http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-
259		spec-os-KerberosTokenProfile.pdf
260		
261	[WSS:SAMLTokenProfile1.0]	OASIS Standard, "Web Services Security: SAML Token Profile",
262		December 2004
263		http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
264		
265	[WSS:SAMLTokenProfile1.1]	OASIS Standard, "Web Services Security: SAML Token Profile 1.1",
266		February 2006
267		http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-
268		spec-os-SAMLTokenProfile.pdf
269		
270	[WSS:RELTokenProfile1.0]	OASIS Standard, "Web Services Security Rights Expression Language
271		(REL) Token Profile", December 2004
272		http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf
273		
274	[WSS:RELTokenProfile1.1]	OASIS Standard, "Web Services Security Rights Expression Language
275		(REL) Token Profile 1.1", February 2006
276		http://www.oasis-open.org/committees/download.php/16687/oasis-
277		wss-rel-token-profile-1.1.pdf

278 279	[WSS:SwAProfile1.1]	OASIS Standard, "Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1", February 2006
280 281		http://www.oasis-open.org/committees/download.php/16672/wss-v1.1-spec-os-SwAProfile.pdf
282		
283 284	[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002.
285		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
286		
287 288	[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002.
289		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
290		
291 292	[XPATH]	W3C Recommendation "XML Path Language (XPath) Version 1.0", 16 November 1999.
293		http://www.w3.org/TR/1999/REC-xpath-19991116
294		
295 296	[XML-Schema1]	W3C Recommendation, "XML Schema Part 1: Structures Second Edition", 28 October 2004.
297		http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
298		
299 300	[XML-Schema2]	W3C Recommendation, "XML Schema Part 2: Datatypes Second Edition", 28 October 2004.
301		http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
302		
303		

304   ## 1.6 Non-Normative References

305   None.

306

2 Security Policy Model

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

312

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

317

318 It is a goal of the security policy model to leverage the WS-Policy framework's intersection algorithm for
319 selecting policy alternatives and the attachment mechanism for associating policy assertions with web
320 service artifacts. Consequently, wherever possible, the security policy assertions do not use parameters
321 or attributes. This enables first-level, QName based assertion matching without security domain-specific
322 knowledge to be done at the framework level. The first level matching is intended to provide a narrowed
323 set of policy alternatives that are shared by the two parties attempting to establish a secure
324 communication path.

325

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

2.1 Security Assertion Model

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

338

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

341

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

345

346 The security binding assertion is a logical grouping which defines how the general aspects are used to
347 protect the indicated parts. For example, that an asymmetric token is used with a digital signature to
348 provide integrity protection, and that parts are encrypted with a symmetric key which is then encrypted

349 using the public key of the recipient. At its simplest form, the security binding restricts what can be placed
350 in the `wsse:Security` header and the associated processing rules.

351

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

2.2 Nested Policy Assertions

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

2.3 Security Binding Abstraction

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

369

370 A binding defines the following security characteristics:

371 • The minimum set of tokens that will be used and how they are bound to messages. Note that
372 services might accept messages containing more tokens than those specified in policy.

373 • Any necessary key transport mechanisms

374 • Any required message elements (e.g. timestamps) in the `wsse:Security` header.

375 • The content and ordering of elements in the `wsse:Security` header. Elements not specified in
376 the binding are not allowed.

377 • Various parameters, including those describing the algorithms to be used for canonicalization,
378 signing and encryption.

379

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

386

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

390 • TransportBinding (Section 7.3)

391 • SymmetricBinding (Section 7.4)

392   • AsymmetricBinding (Section 7.5)

3 Policy Considerations

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

3.1 Nested Policy

397 This specification makes extensive use of nested policy assertions as described in the Policy Assertion
398 Nesting section of WS-Policy.

399

3.2 Policy Subjects

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

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

405 **[Message Policy Subject]**

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

408

409 wsdl:message

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

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

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

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

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

418 **[Operation Policy Subject]**

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

420 wsdl:portType/wsdl:operation

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

423 wsdl:binding/wsdl:operation

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

426

427

428 **[Endpoint Policy Subject]**

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

431 wsdl:portType

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

wsdl:binding

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

wsdl:port

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

4 Protection Assertions

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

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

4.1 Integrity Assertions

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

4.1.1 SignedParts Assertion

The SignedParts assertion is used to specify the parts of the message outside of security headers that require integrity protection. This assertion can be satisfied using WSS: SOAP Message Security mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the message over a secure transport protocol like HTTPS. The binding specific token properties 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 xmlns:sp="..." ... >
  <sp:Body />?
  <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
  <sp:Attachments />?
  ...
</sp:SignedParts>
```

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

/sp:SignedParts

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

/sp:SignedParts/sp:Body

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

/sp:SignedParts/sp:Header

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

482	single sp:SignedParts element. If multiple SOAP headers with the same local name but different
483	namespace names are to be integrity protected multiple sp:Header elements are needed, either
484	as part of a single sp:SignedParts assertion or as part of separate sp:SignedParts assertions.
485	This element only applies to SOAP header elements targeted to the same actor/role as the
486	Security header impacted by the policy. If it is necessary to specify a requirement to sign specific
487	SOAP Header elements targeted to a different actor/role, that may be accomplished using the
488	sp:SignedElements assertion.
489	/sp:SignedParts/sp:Header/@Name
490	This optional attribute indicates the local name of the SOAP header to be integrity protected. If
491	this attribute is not specified, all SOAP headers whose namespace matches the Namespace
492	attribute are to be protected.
493	/sp:SignedParts/sp:Header/@Namespace
494	This required attribute indicates the namespace of the SOAP header(s) to be integrity protected.
495	/sp:SignedParts/sp:Attachments
496	Presence of this optional empty element indicates that all SwA (SOAP Messages with
497	Attachments) attachments [SwA] are to be integrity protected. When SOAP Message Security is
498	used to accomplish this, all message parts other than the part containing the primary SOAP
499	envelope are to be integrity protected as outlined in WSS: SOAP Message Security
500	[WSS:SwAProfile1.1].

4.1.2 SignedElements Assertion

502	The SignedElements assertion is used to specify arbitrary elements in the message that require integrity
503	protection. This assertion can be satisfied using WSS: SOAP Message Security mechanisms or by
504	mechanisms out of scope of SOAP message security, for example by sending the message over a
505	secure transport protocol like HTTPS. The binding specific token properties details the exact mechanism
506	by which the protection is provided.
507	
508	There MAY be multiple SignedElements assertions present. Multiple SignedElements assertions present
509	within a policy alternative are equivalent to a single SignedElements assertion containing the union of all
510	specified XPath expressions.
511	**Syntax**

```
512   <sp:SignedElements XPathVersion="xs:anyURI"? xmlns:sp="..." ... >
513     <sp:XPath>xs:string</sp:XPath>+
514     ...
515   </sp:SignedElements>
```

516	The following describes the attributes and elements listed in the schema outlined above:
517	/sp:SignedElements
518	This assertion specifies the parts of the message that need integrity protection.
519	/sp:SignedElements/@XPathVersion
520	This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is
521	provided, then XPath 1.0 is assumed.
522	/sp:SignedElements/sp:XPath
523	This element contains a string specifying an XPath expression that identifies the nodes to be
524	integrity protected. The XPath expression is evaluated against the S:Envelope element node of
525	the message. Multiple instances of this element may appear within this assertion and should be
526	treated as separate references in a signature when message security is used.

4.2 Confidentiality Assertions

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

4.2.1 EncryptedParts Assertion

The EncryptedParts assertion is used to specify the parts of the message that require confidentiality. This assertion can be satisfied with WSS: SOAP Message Security mechanisms or by mechanisms out of scope of SOAP message security, for example by sending the message over a secure transport protocol like HTTPS. The binding specific token properties 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 xmlns:sp="..." ... >
  <sp:Body/>?
  <sp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... />*
  <sp:Attachments />?
  ...
</sp:EncryptedParts>
```

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

/sp:EncryptedParts

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

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

/sp:EncryptedParts/sp:Body

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

/sp:EncryptedParts/sp:Header

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

/sp:EncryptedParts/sp:Header/@Name

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

573 /sp:EncryptedParts/sp:Header/@Namespace

| 574 | This required attribute indicates the namespace of the SOAP header(s) to be confidentiality |
| 575 | protected. |

576 /sp:EncryptedParts/sp:Attachments

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

4.2.2 EncryptedElements Assertion

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

588

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

Syntax

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

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

598 /sp:EncryptedElements

| 599 | This assertion specifies the parts of the message that need confidentiality protection. Any such |
| 600 | elements are subject to #Element encryption. |

601 /sp:EncryptedElements/@XPathVersion

| 602 | This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is |
| 603 | provided, then XPath 1.0 is assumed. |

604 /sp:EncryptedElements/sp:XPath

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

4.2.3 ContentEncryptedElements Assertion

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

615

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

619 **Syntax**

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

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

625 /sp:ContentEncryptedElements

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

628 /sp:ContentEncryptedElements/@XPathVersion

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

630 /sp:ContentEncryptedElements/sp:XPath

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

635 4.3 Required Elements Assertion

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

638

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

642 4.3.1 RequiredElements Assertion

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

645

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

649 **Syntax**

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

654

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

656 /sp:RequiredElements

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

658 /sp:RequiredElements/@XPathVersion

| 659 | This optional attribute contains a URI which indicates the version of XPath to use. If no attribute is |
| 660 | provided, then XPath 1.0 is assumed. |

661 /sp:RequiredElements/sp:XPath

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

4.3.2 RequiredParts Assertion

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

670

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

674 **Syntax**

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

678

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

680 /sp:RequiredParts/sp:Header

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

682 /sp:RequiredParts/sp:Header/@Name

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

685 /sp:RequiredParts/sp:Header/@Namespace

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

688 5 Token Assertions

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

693 5.1 Token Inclusion

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

699 5.1.1 Token Inclusion Values

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

http://docs.oasis-open.org/ws-sx/ws-securitypolicy/~~200512~~200702/IncludeToken/Never	The token MUST NOT be included in any messages sent between the initiator and the recipient; rather, an external reference to the token should be used.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/~~200512~~200702/IncludeToken/Once	The token MUST be included in only one message sent from the initiator to the recipient. References to the token MAY use an internal reference mechanism. Subsequent related messages sent between the recipient and the initiator may refer to the token using an external reference mechanism.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/~~200512~~200702/IncludeToken/AlwaysToRecipient	The token MUST be included in all messages sent from initiator to the recipient. The token MUST NOT be included in messages sent from the recipient to the initiator.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/~~200512~~200702/IncludeToken/AlwaysToInitiator	The token MUST be included in all messages sent from the recipient to the initiator. The token MUST NOT be included in messages sent from the initiator to the recipient.
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/~~200512~~200702/IncludeToken/Always	The token MUST be included in all messages sent between the initiator and the recipient. This is the default behavior.

701

702 Note: In examples, the namespace URI is replaced with "..." for brevity. For example,
703 .../IncludeToken/Never is actually http://docs.oasis-open.org/ws-sx/ws-
704 securitypolicy/~~200512~~200702/IncludeToken/Never. Other token inclusion URI values MAY be defined but
705 are out-of-scope of this specification.

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

5.1.2 Token Inclusion and Token References

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

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

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

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

5.2 Token Issuer and Required Claims

5.2.1 Token Issuer

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

5.2.2 Token Issuer Name

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

It is out of scope of this specification how the relationship between the issuer's logical name and the physical manifestation of the issuer in the security token is defined.
While both sp:Issuer and sp:IssuerName elements are optional they are also mutually exclusive and cannot be specified both at the same time.

5.2.3 Required Claims

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

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

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

5.2.4 Processing Rules and Token Matching

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

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

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

5.25.3 Token Properties

5.2.15.3.1 [Derived Keys] Property

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

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

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

5.2.25.3.2 [Explicit Derived Keys] Property

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

5.2.35.3.3 [Implicit Implied Derived Keys] Property

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

5.35.4 Token Assertion Types

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

5.3.15.4.1 UsernameToken Assertion

This element represents a requirement to include a username token.

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

793   1.   When transport security is not used.

794   2.   When a plaintext password is used.

795   3.   When a weak password hash is used.

796   4.   When the username needs to be protected, e.g. for privacy reasons.

797   When the UsernameToken is to be encrypted it SHOULD be listed as a

798   SignedEncryptedSupportingToken (Section 8.5), EndorsingEncryptedSupportingToken (Section 8.6) or

799   SignedEndorsingEncryptedSupportingToken (Section 8.7).

800

801   **Syntax**

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

826

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

828   /sp:UsernameToken

829   This identifies a UsernameToken assertion.

830   /sp:UsernameToken/@sp:IncludeToken

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

832   /sp:UsernameToken sp:Issuer

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

835   /sp:UsernameToken /sp:IssuerName

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

838   /sp:UsernameToken/wst:Claims

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

841   /sp:UsernameToken/wsp:Policy

842   This ~~optional~~ required element identifies additional requirements for use of the
843   sp:UsernameToken assertion.

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

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

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

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

852 /sp:UsernameToken/wsp:Policy/sp:RequireDerivedKeys

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

855 /sp:UsernameToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

859 /sp:UsernameToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

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

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

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

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

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

869 ## ~~5.3.2~~5.4.2 IssuedToken Assertion

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

874 **Syntax**

875
```
<sp:IssuedToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
876   (
877    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
878    <sp:IssuerName>xs:anyURI</sp:IssuerName>
879   ) ?
```

```
880    <wst:Claims Dialect="..."> ... </wst:Claims> ?
881   <sp:RequestSecurityTokenTemplate TrustVersion="xs:anyURI"? >
882     ...
883   </sp:RequestSecurityTokenTemplate>
884   <wsp:Policy xmlns:wsp="...">
885     (
886       <sp:RequireDerivedKeys ... /> |
887       <sp:RequireImplicitImpliedDerivedKeys ... /> |
888       <sp:RequireExplicitDerivedKeys ... />
889     ) ?
890     <sp:RequireExternalReference ... /> ?
891     <sp:RequireInternalReference ... /> ?
892     ...
893   </wsp:Policy> ?
894   ...
895 </sp:IssuedToken>
```

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

897 /sp:IssuedToken

898   This identifies an IssuedToken assertion.

899 /sp:IssuedToken/@sp:IncludeToken

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

901 /sp:IssuedToken/sp:Issuer

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

904 /sp:IssuedToken/sp:IssuerName

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

906 /sp:IssuedToken/wst:Claims

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

909 /sp:IssuedToken/sp:RequestSecurityTokenTemplate

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

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

914 /sp:IssuedToken/sp:RequestSecurityTokenTemplate/@TrustVersion

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

917 /sp:IssuedToken/wsp:Policy

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

920 /sp:IssuedToken/wsp:Policy/sp:RequireDerivedKeys

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

923 /sp:IssuedToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

927 /sp:IssuedToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

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

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

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

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

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

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

944 ~~5.3.3~~5.4.3 X509Token Assertion

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

946 **Syntax**

947
```
<sp:X509Token sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
  (
    <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
    <sp:IssuerName>xs:anyURI</sp:IssuerName>
  ) ?
  <wst:Claims Dialect="..."> ... </wst:Claims> ?
  <wsp:Policy xmlns:wsp="...">
    (
      <sp:RequireDerivedKeys ... /> |
      <sp:RequireExplicitDerivedKeys ... /> |
      <sp:RequireImplicitImpliedDerivedKeys ... />
    ) ?
    <sp:RequireKeyIdentifierReference ... /> ?
    <sp:RequireIssuerSerialReference ... /> ?
    <sp:RequireEmbeddedTokenReference ... /> ?
    <sp:RequireThumbprintReference ... /> ?
    (
      <sp:WssX509V3Token10 ... /> |
      <sp:WssX509Pkcs7Token10 ... /> |
      <sp:WssX509PkiPathV1Token10 ... /> |
      <sp:WssX509V1Token11 ... /> |
      <sp:WssX509V3Token11 ... /> |
      <sp:WssX509Pkcs7Token11 ... /> |
      <sp:WssX509PkiPathV1Token11 ... />
    ) ?
    ...
  </wsp:Policy> ?
  ...
</sp:X509Token>
```

976

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

978 /sp:X509Token

979   This identifies an X509Token assertion.

980   /sp:X509Token/@sp:IncludeToken

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

982   /sp:X509Token/sp:Issuer

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

985   /sp:X509Token/sp:IssuerName

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

987   /sp:X509Token/wst:Claims

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

990   /sp:X509Token/wsp:Policy

991   This ~~optional~~ required element identifies additional requirements for use of the sp:X509Token
992   assertion.

993   /sp:X509Token/wsp:Policy/sp:RequireDerivedKeys

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

996   /sp:X509Token/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1000   /sp:X509Token/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

1001   This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~ Implied
1002   Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1003   token to 'false'.

1004   /sp:X509Token/wsp:Policy/sp:RequireKeyIdentifierReference

1005   This optional element is a policy assertion that indicates that a key identifier reference is required
1006   when referencing this token.

1007   /sp:X509Token/wsp:Policy/sp:RequireIssuerSerialReference

1008   This optional element is a policy assertion that indicates that an issuer serial reference is required
1009   when referencing this token.

1010   /sp:X509Token/wsp:Policy/sp:RequireEmbeddedTokenReference

1011   This optional element is a policy assertion that indicates that an embedded token reference is
1012   required when referencing this token.

1013   /sp:X509Token/wsp:Policy/sp:RequireThumbprintReference

1014   This optional element is a policy assertion that indicates that a thumbprint reference is required
1015   when referencing this token.

1016   /sp:X509Token/wsp:Policy/sp:WssX509V3Token10

1017   This optional element is a policy assertion that indicates that an X509 Version 3 token should be
1018   used as defined in [WSS:X509TokenProfile1.0].

1019   /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token10

1020   This optional element is a policy assertion that indicates that an X509 PKCS7 token should be
1021   used as defined in [WSS:X509TokenProfile1.0].

1022   /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token10

1023   This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token
1024   should be used as defined in [WSS:X509TokenProfile1.0].

1025   /sp:X509Token/wsp:Policy/sp:WssX509V1Token11

1026   This optional element is a policy assertion that indicates that an X509 Version 1 token should be
1027   used as defined in [WSS:X509TokenProfile1.1].

1028   /sp:X509Token/wsp:Policy/sp:WssX509V3Token11

1029   This optional element is a policy assertion that indicates that an X509 Version 3 token should be
1030   used as defined in [WSS:X509TokenProfile1.1].

1031   /sp:X509Token/wsp:Policy/sp:WssX509Pkcs7Token11

1032   This optional element is a policy assertion that indicates that an X509 PKCS7 token should be
1033   used as defined in [WSS:X509TokenProfile1.1].

1034   /sp:X509Token/wsp:Policy/sp:WssX509PkiPathV1Token11

1035   This optional element is a policy assertion that indicates that an X509 PKI Path Version 1 token
1036   should be used as defined in [WSS:X509TokenProfile1.1].

1037   ## ~~5.3.4~~5.4.4 KerberosToken Assertion

1038   This element represents a requirement for a Kerberos token [WSS:KerberosToken1.1].

1039   **Syntax**

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

    ...
  </wsp:Policy> ?
  ...
</sp:KerberosToken>
```

1063   The following describes the attributes and elements listed in the schema outlined above:

1064   /sp:KerberosToken

1065   This identifies a KerberosV5ApReqToken assertion.

1066   /sp:KerberosToken/@sp:IncludeToken

1067   This optional attribute identifies the token inclusion value for this token assertion.

1068   /sp:KerberosToken/sp:Issuer

1069   This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of
1070   the sp:KerberosToken.

1071 /sp:KerberosToken/sp:IssuerName

1072   This optional element, of type xs:anyURI, contains the logical name of the sp:KerberosToken
1073   issuer.

1074 /sp:KerberosToken/wst:Claims

1075   This optional element identifies the required claims that a security token must contain in order to
1076   satisfy the token assertion requirements.

1077 /sp:KerberosToken/wsp:Policy

1078   This ~~optional~~ required element identifies additional requirements for use of the sp:KerberosToken
1079   assertion.

1080 /sp:KerberosToken/wsp:Policy/sp:RequireDerivedKeys

1081   This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1082   and [~~Implicit~~ Implied Derived Keys] properties for this token to 'true'.

1083 /sp:KerberosToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1084   This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1085   properties for this token to 'true' and the [~~Implicit~~ Implied Derived Keys] property for this token to
1086   'false'.

1087 /sp:KerberosToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

1088   This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~ Implied
1089   Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1090   token to 'false'.

1091 /sp:KerberosToken/wsp:Policy/sp:RequireKeyIdentifierReference

1092   This optional element is a policy assertion that indicates that a key identifier reference is required
1093   when referencing this token.

1094 /sp:KerberosToken/wsp:Policy/sp:WssKerberosV5ApReqToken11

1095   This optional element is a policy assertion that indicates that a Kerberos Version 5 AP-REQ token
1096   should be used as defined in [WSS:KerberosTokenProfile1.1].

1097 /sp:KerberosToken/wsp:Policy/sp:WssGssKerberosV5ApReqToken11

1098   This optional element is a policy assertion that indicates that a GSS Kerberos Version 5 AP-REQ
1099   token should be used as defined in [WSS:KerberosTokenProfile1.1].

1100 ## ~~5.3.5~~5.4.5 SpnegoContextToken Assertion

1101 This element represents a requirement for a SecurityContextToken obtained by executing an n-leg
1102 RST/RSTR SPNEGO binary negotiation protocol with the Web Service, as defined in WS-Trust.

1103 **Syntax**

```
1104    <sp:SpnegoContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1105      (
1106      <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1107      <sp:IssuerName>xs:anyURI</sp:IssuerName>
```

```
1108        ) ?
1109      <wst:Claims Dialect="..."> ... </wst:Claims> ?
1110      <wsp:Policy xmlns:wsp="...">
1111        (
1112          <sp:RequireDerivedKeys ... /> |
1113          <sp:RequireImplicitImpliedDerivedKeys ... /> |
1114          <sp:RequireExplicitDerivedKeys ... />
1115        ) ?
1116        <sp:MustNotSendCancel ... /> ?
1117        <sp:MustNotSendAmend ... /> ?
1118        <sp:MustNotSendRenew ... /> ?
1119        ...
1120      </wsp:Policy> ?
1121      ...
1122    </sp:SpnegoContextToken>
```

1123

1124 The following describes the attributes and elements listed in the schema outlined above:

1125 /sp:SpnegoContextToken

1126   This identifies a SpnegoContextToken assertion.

1127 /sp:SpnegoContextToken/@sp:IncludeToken

1128   This optional attribute identifies the token inclusion value for this token assertion.

1129 /sp:SpnegoContextToken/sp:Issuer

1130   This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for
1131   the Spnego Context Token.

1132 /sp:SpnegoContextToken/sp:IssuerName

1133   This optional element, of type xs:anyURI, contains the logical name of the
1134   sp:SpnegoContextToken issuer.

1135 /sp:SpnegoContextToken/wst:Claims

1136   This optional element identifies the required claims that a security token must contain in order to
1137   satisfy the token assertion requirements.

1138 /sp:SpnegoContextToken/wsp:Policy

1139   This optional required element identifies additional requirements for use of the
1140   sp:SpnegoContextToken assertion.

1141 /sp:SpnegoContextToken/wsp:Policy/sp:RequireDerivedKeys

1142   This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1143   and [ImplicitImplied Derived Keys] properties for this token to 'true'.

1144 /sp:SpnegoContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1145   This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1146   properties for this token to 'true' and the [ImplicitImplied Derived Keys] property for this token to
1147   'false'.

1148 /sp:SpnegoContextToken/wsp:Policy/sp:RequireImplicitImpliedDerivedKeys

1149   This optional element is a policy assertion that sets the [Derived Keys] and [ImplicitImplied
1150   Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1151   token to 'false'.

1152 sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendCancel

1153   This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token
1154   does not support SCT/Cancel RST messages. If this assertion is missing it means that
1155   SCT/Cancel RST messages are supported by the STS.

1156	/sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendAmend
1157	This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token
1158	does not support SCT/Amend RST messages. If this assertion is missing it means that
1159	SCT/Amend RST messages are supported by the STS.
1160	/sp:SpnegoContextToken/wsp:Policy/sp:MustNotSendRenew
1161	This optional element is a policy assertion that indicates that the STS issuing the SP/Nego token
1162	does not support SCT/Renew RST messages. If this assertion is missing it means that
1163	SCT/Renew RST messages are supported by the STS.

1164 ## 5.3.65.4.6 SecurityContextToken Assertion

1165 This element represents a requirement for a SecurityContextToken token.

1166 **Syntax**

```
1167   <sp:SecurityContextToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1168     (
1169       <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1170       <sp:IssuerName>xs:anyURI</sp:IssuerName>
1171     ) ?
1172     <wst:Claims Dialect="..."> ... </wst:Claims> ?
1173     <wsp:Policy xmlns:wsp="...">
1174       (
1175         <sp:RequireDerivedKeys ... /> |
1176         <sp:RequireImplicitImpliedDerivedKeys ... /> |
1177         <sp:RequireExplicitDerivedKeys ... />
1178       ) ?
1179       <sp:RequireExternalUriReference ... /> ?
1180       <sp:SC200502SecurityContextToken SC13SecurityContextToken... /> ?
1181       ...
1182     </wsp:Policy> ?
1183     ...
1184   </sp:SecurityContextToken>
```

1185

1186 The following describes the attributes and elements listed in the schema outlined above:

1187 /sp:SecurityContextToken

1188 This identifies a SecurityContextToken assertion.

1189 /sp:SecurityContextToken/@sp:IncludeToken

1190 This optional attribute identifies the token inclusion value for this token assertion.

1191	/sp:SecurityContextToken/sp:Issuer
1192	This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of
1193	the sp:SecurityContextToken.
1194	/sp:SecurityContextToken/sp:IssuerName
1195	This optional element, of type xs:anyURI, contains the logical name of the
1196	sp:SecurityContextToken issuer.
1197	/sp:SecurityContextToken/wst:Claims
1198	This optional element identifies the required claims that a security token must contain in order to
1199	satisfy the token assertion requirements.

1200 /sp:SecurityContextToken/wsp:Policy

1201 This optional required element identifies additional requirements for use of the
1202 sp:SecurityContextToken assertion.

1203 /sp:SecurityContextToken/wsp:Policy/sp:RequireDerivedKeys

1204 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1205 and [~~Implicit~~Implied Derived Keys] properties for this token to 'true'.

1206 /sp:SecurityContextToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1207 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1208 properties for this token to 'true' and the [~~Implicit~~Implied Derived Keys] property for this token to
1209 'false'.

1210 /sp:SecurityContextToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

1211 This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~Implied
1212 Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1213 token to 'false'.

1214 /sp:SecurityContextToken/wsp:Policy/sp:RequireExternalUriReference

1215 This optional element is a policy assertion that indicates that an external URI reference is
1216 required when referencing this token.

1217 /sp:SecurityContextToken/wsp:Policy/sp:~~SC200502SecurityContextToken~~SC13SecurityContextToken

1218 This optional element is a policy assertion that indicates that a Security Context Token should be
1219 used as defined in [WS-SecureConversation].

1220

1221 Note: This assertion does not describe how to obtain a Security Context Token but rather assumes that
1222 both parties have the token already or have agreed separately on a mechanism for obtaining the token. If
1223 a definition of the mechanism for obtaining the Security Context Token is desired in policy, then either the
1224 sp:SecureConversationToken or the sp:IssuedToken assertion should be used instead.

~~5.3.7~~5.4.7 SecureConversationToken Assertion

1226 This element represents a requirement for a Security Context Token retrieved from the indicated issuer
1227 address. If the sp:Issuer address is absent, the protocol MUST be executed at the same address as the
1228 service endpoint address.

1229

1230 Note: This assertion describes the token accepted by the target service. Because this token is issued by
1231 the target service and may not have a separate port (with separate policy), this assertion SHOULD
1232 contain a bootstrap policy indicating the security binding and policy that is used when requesting this
1233 token from the target service. That is, the bootstrap policy is used to obtain the token and then the
1234 current (outer) policy is used when making requests with the token. This is illustrated in the diagram
1235 below.

1237 **Syntax**

1238
```
<sp:SecureConversationToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1239     (
```

ws-securitypolicy-1.2-spec-cd-02   7 March 2007
Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.   Page 37 of 113
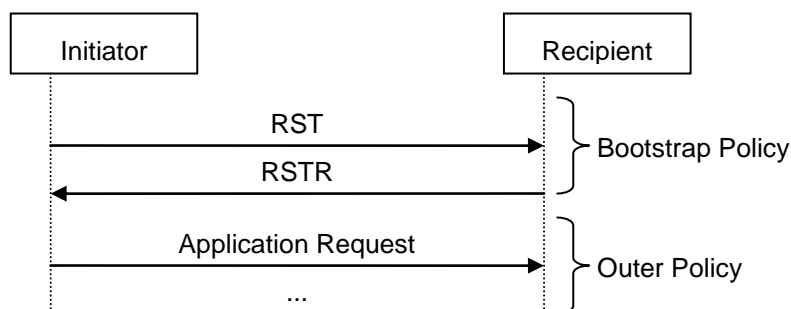
```
1240        <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1241        <sp:IssuerName>xs:anyURI</sp:IssuerName>
1242        ) ?
1243        <wst:Claims Dialect="..."> ... </wst:Claims> ?
1244        <wsp:Policy xmlns:wsp="...">
1245          (
1246            <sp:RequireDerivedKeys ... /> |
1247            <sp:RequireImplicitImpliedDerivedKeys ... /> |
1248            <sp:RequireExplicitDerivedKeys ... />
1249          ) ?
1250          <sp:RequireExternalUriReference ... /> ?
1251          <sp:SC200502SecurityContextToken SC13SecurityContextToken ... /> ?
1252          <sp:MustNotSendCancel ... /> ?
1253          <sp:MustNotSendAmend ... /> ?
1254          <sp:MustNotSendRenew ... /> ?
1255          <sp:BootstrapPolicy ... > ?
1256            <wsp:Policy> ... </wsp:Policy>
1257          </sp:BootstrapPolicy>
1258        </wsp:Policy> ?
1259        ...
1260      </sp:SecureConversationToken>
```

1262 The following describes the attributes and elements listed in the schema outlined above:

1263 /sp:SecureConversationToken

1264   This identifies a SecureConversationToken assertion.

1265 /sp:SecureConversationToken/@sp:IncludeToken

1266   This optional attribute identifies the token inclusion value for this token assertion.

1267 /sp:SecureConversationToken/sp:Issuer

1268   This optional element, of type wsa:EndpointReferenceType, contains a reference to the issuer for
1269   the Security Context Token.

1270 /sp:SecureConversationToken/sp:IssuerName

1271   This optional element, of type xs:anyURI, contains the logical name of the
1272   sp:SecureConversationToken issuer.

1273 /sp:SpnegoContextToken/wst:Claims

1274   This optional element identifies the required claims that a security token must contain in order to
1275   satisfy the token assertion requirements.

1276 /sp:SecureConversationToken/wsp:Policy

1277   This optional required element identifies additional requirements for use of the
1278   sp:SecureConversationToken assertion.

1279 /sp:SecureConversationToken/wsp:Policy/sp:RequireDerivedKeys

1280   This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1281   and [ImplicitImplied Derived Keys] properties for this token to 'true'.

1282 /sp:SecureConversationToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1283   This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1284   properties for this token to 'true' and the [ImplicitImplied Derived Keys] property for this token to
1285   'false'.

1286 /sp:SecureConversationToken/wsp:Policy/sp:RequireImplicitImpliedDerivedKeys

1287   This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~Implied
1288   Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1289   token to 'false'.

1290 /sp:SecureConversationToken/wsp:Policy/sp:RequireExternalUriReference

1291   This optional element is a policy assertion that indicates that an external URI reference is
1292   required when referencing this token.

1293 /sp:SecureConversationToken/wsp:Policy/sp:~~SC200502SecurityContextToken~~SC13SecurityContextToke
1294 n

1295   This optional element is a policy assertion that indicates that a Security Context Token should be
1296   used as obtained using the protocol defined in [WS-SecureConversation].

1297 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendCancel

1298   This optional element is a policy assertion that indicates that the STS issuing the secure
1299   conversation token does not support SCT/Cancel RST messages. If this assertion is missing it
1300   means that SCT/Cancel RST messages are supported by the STS.

1301 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendAmend

1302   This optional element is a policy assertion that indicates that the STS issuing the secure
1303   conversation token does not support SCT/Amend RST messages. If this assertion is missing it
1304   means that SCT/Amend RST messages are supported by the STS.

1305 /sp:SecureConversationToken/wsp:Policy/sp:MustNotSendRenew

1306   This optional element is a policy assertion that indicates that the STS issuing the secure
1307   conversation token does not support SCT/Renew RST messages. If this assertion is missing it
1308   means that SCT/Renew RST messages are supported by the STS.

1309 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy

1310   This optional element is a policy assertion that contains the policy indicating the requirements for
1311   obtaining the Security Context Token.

1312 /sp:SecureConversationToken/wsp:Policy/sp:BootstrapPolicy/wsp:Policy

1313   This element contains the security binding requirements for obtaining the Security Context Token.
1314   It will typically contain a security binding assertion (e.g. sp:SymmetricBinding) along with
1315   protection assertions (e.g. sp:SignedParts) describing the parts of the RST/RSTR messages that
1316   are to be protected.

1317 **Example**

```
1318   <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
1319     <sp:SymmetricBinding>
1320       <wsp:Policy>
1321         <sp:ProtectionToken>
1322           <wsp:Policy>
1323             <sp:SecureConversationToken>
1324               <sp:Issuer>
1325                 <wsa:Address>http://example.org/sts</wsa:Address>
1326               </sp:Issuer>
1327               <wsp:Policy>
```

```
1328                      <sp:SC10SecurityContextToken />
1329                      <sp:BootstrapPolicy>
1330                        <wsp:Policy>
1331                          <sp:AsymmetricBinding>
1332                            <wsp:Policy>
1333                              <sp:InitiatorToken>
1334                                ...
1335                              </sp:InitiatorToken>
1336                              <sp:RecipientToken>
1337                                ...
1338                              </sp:RecipientToken>
1339                            </wsp:Policy>
1340                          </sp:AsymmetricBinding>
1341                          <sp:SignedParts>
1342                            ...
1343                          </sp:SignedParts>
1344                          ...
1345                        </wsp:Policy>
1346                      </sp:BootstrapPolicy>
1347                    </wsp:Policy>
1348                  </sp:SecureConversationToken>
1349              </wsp:Policy>
1350            </sp:ProtectionToken>
1351            ...
1352        </wsp:Policy>
1353      </sp:SymmetricBinding>
1354      <sp:SignedParts>
1355      ...
1356      </sp:SignedParts>
1357      ...
1358    </wsp:Policy>
```

1359 ~~5.3.8~~5.4.8 SamlToken Assertion

1360 This element represents a requirement for a SAML token.

1361 **Syntax**

```
1362    <sp:SamlToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1363      (
1364        <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1365        <sp:IssuerName>xs:anyURI</sp:IssuerName>
1366      ) ?
1367      <wst:Claims Dialect="..."> ... </wst:Claims> ?
1368      <wsp:Policy xmlns:wsp="...">
1369        (
1370          <sp:RequireDerivedKeys ... /> |
1371          <sp:Require~~Implicit~~ImpliedDerivedKeys ... /> |
1372          <sp:RequireExplicitDerivedKeys ... />
1373        ) ?
1374        <sp:RequireKeyIdentifierReference ... /> ?
1375        (
1376          <sp:WssSamlV11Token10 ... /> |
1377          <sp:WssSamlV11Token11 ... /> |
1378          <sp:WssSamlV20Token11 ... />
1379        ) ?
1380        ...
1381      </wsp:Policy> ~~?~~
1382      ...
1383    </sp:SamlToken>
```

1384

1385 The following describes the attributes and elements listed in the schema outlined above:

1386 /sp:SamlToken

This identifies a SamlToken assertion.

1388 /sp:SamlToken/@sp:IncludeToken

1389 This optional attribute identifies the token inclusion value for this token assertion.

1390 /sp:SamlToken/sp:Issuer

1391 This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of
1392 the sp:SamlToken.

1393 /sp:SamlToken/sp:IssuerName

1394 This optional element, of type xs:anyURI, contains the logical name of the sp:SamlToken issuer.

1395 /sp:SamlToken/wst:Claims

1396 This optional element identifies the required claims that a security token must contain in order to
1397 satisfy the token assertion requirements.

1398 /sp:SamlToken/wsp:Policy

1399 This ~~optional~~ required element identifies additional requirements for use of the sp:SamlToken
1400 assertion.

1401 /sp:SamlToken/wsp:Policy/sp:RequireDerivedKeys

1402 This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1403 and [~~Implicit~~Implied Derived Keys] properties for this token to 'true'.

1404 /sp:SamlToken/wsp:Policy/sp:RequireExplicitDerivedKeys

1405 This optional element is a policy assertion that sets the [Derived Keys] and [Explicit Derived Keys]
1406 properties for this token to 'true' and the [~~Implicit~~Implied Derived Keys] property for this token to
1407 'false'.

1408 /sp:SamlToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

1409 This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~Implied
1410 Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1411 token to 'false'.

1412 /sp:SamlToken/wsp:Policy/sp:RequireKeyIdentifierReference

1413 This optional element is a policy assertion that indicates that a key identifier reference is required
1414 when referencing this token.

1415 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token10

1416 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
1417 used as defined in [WSS:SAMLTokenProfile1.0].

1418 /sp:SamlToken/wsp:Policy/sp:WssSamlV11Token11

1419 This optional element is a policy assertion that identifies that a SAML Version 1.1 token should be
1420 used as defined in [WSS:SAMLTokenProfile1.1].

1421 /sp:SamlToken/wsp:Policy/sp:WssSamlV20Token11

1422 This optional element is a policy assertion that identifies that a SAML Version 2.0 token should be
1423 used as defined in [WSS:SAMLTokenProfile1.1].

1424

1425 Note: This assertion does not describe how to obtain a SAML Token but rather assumes that both parties
1426 have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1427 of the mechanism for obtaining the SAML Token is desired in policy, the sp:IssuedToken assertion should
1428 be used instead.

1429 ~~5.3.9~~5.4.9 RelToken Assertion

1430 This element represents a requirement for a REL token.

1431 **Syntax**

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

1456 The following describes the attributes and elements listed in the schema outlined above:

1457 /sp:RelToken

1458   This identifies a RelToken assertion.

1459 /sp:RelToken/@sp:IncludeToken

1460   This optional attribute identifies the token inclusion value for this token assertion.

1461 /sp:RelToken/sp:Issuer

1462   This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of
1463   the sp:RelToken.

1464 /sp:RelToken/sp:IssuerName

1465   This optional element, of type xs:anyURI, contains the logical name of the sp:RelToken issuer.

1466 /sp:RelToken/wst:Claims

1467   This optional element identifies the required claims that a security token must contain in order to
1468   satisfy the token assertion requirements.

1469 /sp:RelToken/wsp:Policy

1470   This ~~optional~~ required element identifies additional requirements for use of the sp:RelToken
1471   assertion.

1472 /sp:RelToken/wsp:Policy/sp:RequireDerivedKeys

1473   This optional element is a policy assertion that sets the [Derived Keys], [Explicit Derived Keys]
1474   and [~~Implicit~~Implied Derived Keys] property for this token to 'true'.

1475 /sp:RelToken/wsp:Policy/sp:RequireExplicitDerivedKeys

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

1479　/sp:RelToken/wsp:Policy/sp:Require~~Implicit~~ImpliedDerivedKeys

1480　　　This optional element is a policy assertion that sets the [Derived Keys] and [~~Implicit~~Implied
1481　　　Derived Keys] properties for this token to 'true' and the [Explicit Derived Keys] property for this
1482　　　token to 'false'.

1483　/sp:RelToken/wsp:Policy/sp:RequireKeyIdentifierReference

1484　　　This optional element is a policy assertion that indicates that a key identifier reference is required
1485　　　when referencing this token.

1486　/sp:RelToken/wsp:Policy/sp:WssRelV10Token10

1487　　　This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1488　　　used as defined in [WSS:RELTokenProfile1.0].

1489　/sp:RelToken/wsp:Policy/sp:WssRelV20Token10

1490　　　This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1491　　　used as defined in [WSS:RELTokenProfile1.0].

1492　/sp:RelToken/wsp:Policy/sp:WssRelV10Token11

1493　　　This optional element is a policy assertion that identifies that a REL Version 1.0 token should be
1494　　　used as defined in [WSS:RELTokenProfile1.1].

1495　/sp:RelToken/wsp:Policy/sp:WssRelV20Token11

1496　　　This optional element is a policy assertion that identifies that a REL Version 2.0 token should be
1497　　　used as defined in [WSS:RELTokenProfile1.1].

1498

1499　Note: This assertion does not describe how to obtain a REL Token but rather assumes that both parties
1500　have the token already or have agreed separately on a mechanism for obtaining the token. If a definition
1501　of the mechanism for obtaining the REL Token is desired in policy, the sp:IssuedToken assertion should
1502　be used instead.

1503　~~5.3.10~~5.4.10 HttpsToken Assertion

1504　This element represents a requirement for a transport binding to support the use of HTTPS.

1505　**Syntax**

```
1506    <sp:HttpsToken xmlns:sp="..." ... >
1507      (
1508        <sp:Issuer>wsa:EndpointReferenceType</sp:Issuer> |
1509        <sp:IssuerName>xs:anyURI</sp:IssuerName>
1510      ) ?
1511      <wst:Claims Dialect="..."> ... </wst:Claims> ?
1512      <wsp:Policy xmlns:wsp="...">
1513        (
1514          <sp:HttpBasicAuthentication /> |
1515          <sp:HttpDigestAuthentication /> |
1516          <sp:RequireClientCertificate /> |
1517          ...
1518        )?
1519        ...
1520      </wsp:Policy> ?
1521      ...
1522    </sp:HttpsToken>
```

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

1524 /sp:HttpsToken

1525 This identifies an Https assertion stating that use of the HTTPS protocol specification is
1526 supported.

1527 /sp:HttpsToken/sp:Issuer

1528 This optional element, of type wsa:EndpointReferenceType, contains reference to the issuer of
1529 the sp:HttpsToken.

1530 /sp:HttpsToken/sp:IssuerName

1531 This optional element, of type xs:anyURI, contains the logical name of the sp:HttpsToken issuer.

1532 /sp:HttpsToken/wst:Claims

1533 This optional element identifies the required claims that a security token must contain in order to
1534 satisfy the token assertion requirements.

1535 /sp:HttpsToken/wsp:Policy

1536 This ~~optional~~ required element identifies additional requirements for use of the sp:HttpsToken
1537 assertion.

1538 /sp:HttpsToken/wsp:Policy/sp:HttpBasicAuthentication

1539 This optional element is a policy assertion that indicates that the client MUST use HTTP Basic
1540 Authentication [RFC2068] to authenticate to the service.

1541 /sp:HttpsToken/wsp:Policy/sp:HttpDigestAuthentication

1542 This optional element is a policy assertion that indicates that the client MUST use HTTP Digest
1543 Authentication [RFC2068] to authenticate to the service.

1544 /sp:HttpsToken/wsp:Policy/sp:RequireClientCertificate

1545 This optional element is a policy assertion that indicates that the client MUST provide a certificate
1546 when negotiating the HTTPS session.

1547 5.4.11 KeyValueToken Assertion

1548 This element represents a requirement for a KeyValue token. The next section defines the KeyValue
1549 security token abstraction for purposes of this token assertion.
1550

1551 This document defines requirements for KeyValue token when used in combination with RSA
1552 cryptographic algorithm. Additional cryptographic algorithms can be introduced by another specifications
1553 by introducing new nested assertions besides *sp:RsaKeyValue*.

1554 **Syntax**

1555 ```
<sp:KeyValueToken sp:IncludeToken="xs:anyURI"? xmlns:sp="..." ... >
1556   <wsp:Policy xmlns:wsp="...">
1557   <sp:RsaKeyValue ... /> ?
1558   ...
1559   </wsp:Policy>
1560   ...
1561 </sp:KeyValueToken>
```

1562 The following describes the attributes listed in the schema outlined above:

1563 /sp:KeyValueToken

1564 This identifies a RsaToken assertion.

1565 /sp:KeyValueToken/@sp:IncludeToken

1566 This optional attribute identifies the token inclusion value for this token assertion.

1567 /sp:KeyValueToken/wsp:Policy

1568     This required element identifies additional requirements for use of the sp:KeyValueToken
1569     assertion.

1570 /sp:KeyValueToken/wsp:Policy/sp:RsaKeyValue

1571     This optional element is a policy assertion that indicates that the ds:RSAKeyValue element must
1572     be present in the KeyValue token. This indicates that an RSA key pair must be used.

## 5.4.11.1 KeyValue Token

1574 XML Signature specification allows reference an arbitrary key pair by using the corresponding public key
1575 value. This allows using an arbitrary key pair to sign or encrypt XML elements. The purpose of this
1576 section is to define the KeyValue token abstraction that represents such key pair referencing mechanism.
1577
1578 Although the *ds:KeyValue* element as defined in the XML Signature specification is generic enough to be
1579 used with any asymmetric cryptographic algorithm this document only profiles the usage of *ds:KeyValue*
1580 element in combination with RSA cryptographic algorithm.
1581
1582 The RSA key pair is represented by the *ds:KeyInfo* element containing the *ds:KeyValue* element with the
1583 RSA public key value in *ds:RSAKeyValue* as defined in the XML Signature specification:

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

1592
1593 When the KeyValue token is used the corresponding public key value appears directly in the signature or
1594 encrypted data *ds:KeyInfo* element like in the following example. There is no KeyValue token
1595 manifestation outside the *ds:KeyInfo* element.

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

1619
1620 Since there is no representation of the KeyValue token outside the *ds:KeyInfo* element and thus no
1621 identifier can be associated with the token, the KeyValue token cannot be referenced by using
1622 *wsse:SecurityTokenReference* element. However the *ds:KeyInfo* element representing the KeyValue
1623 token can be used whenever a security token can be used as illustrated on the following example:

```
1624    <t:RequestSecurityToken xmlns:t="...">
1625      <t:RequestType>...</t:RequestType>
1626      ...
1627      <t:UseKey>
1628        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1629          <KeyValue>
1630            <RSAKeyValue>
1631              <Modulus>...</Modulus>
1632              <Exponent>...</Exponent>
1633            </RSAKeyValue>
1634          </KeyValue>
1635        </KeyInfo>
1636      </t:UseKey>
1637    </t:RequestSecurityToken>
```

6 Security Binding Properties

1638

1639 This section defines the various properties or conditions of a security binding, their semantics, values and
1640 defaults where appropriate. Properties are used by a binding in a manner similar to how variables are
1641 used in code. Assertions populate, (or set) the value of the property (or variable). When an assertion that
1642 populates a value of a property appears in a policy, that property is set to the value indicated by the
1643 assertion. The security binding then uses the value of the property to control its behavior. The properties
1644 listed here are common to the various security bindings described in Section 7. Assertions that define
1645 values for these properties are defined in Section 7. The following properties are used by the security
1646 binding assertions.

6.1 [Algorithm Suite] Property

1647

1648 This property specifies the algorithm suite required for performing cryptographic operations with
1649 symmetric or asymmetric key based security tokens. An algorithm suite specifies actual algorithms and
1650 allowed key lengths. A policy alternative will define what algorithms are used and how they are used. This
1651 property defines the set of available algorithms. The value of this property is typically referenced by a
1652 security binding and is used to specify the algorithms used for all message level cryptographic operations
1653 performed under the security binding.

1654 Note: In some cases, this property MAY be referenced under a context other than a security binding and
1655 used to control the algorithms used under that context. For example, supporting token assertions define
1656 such a context. In such contexts, the specified algorithms still apply to message level cryptographic
1657 operations.

1658 An algorithm suite defines values for each of the following operations and properties:

1659 • [Sym Sig]   Symmetric Key Signature
1660 • [Asym Sig]   Signature with an asymmetric key
1661 • [Dig]   Digest
1662 • [Enc]   Encryption
1663 • [Sym KW]   Symmetric Key Wrap
1664 • [Asym KW]   Asymmetric Key Wrap
1665 • [Comp Key]   Computed key
1666 • [Enc KD]   Encryption key derivation
1667 • [Sig KD]   Signature key derivation
1668 • [Min SKL]   Minimum symmetric key length
1669 • [Max SKL]   Maximum symmetric key length
1670 • [Min AKL]   Minimum asymmetric key length
1671 • [Max AKL]   Maximum asymmetric key length

1672

1673 The following table provides abbreviations for the algorithm URI used in the table below:

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

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

1674

1675 The tables below show all the base algorithm suites defined by this specification. This table defines
1676 values for properties which are common for all suites:

Property	Algorithm / Value
[Sym Sig]	HmacSha1
[Asym Sig]	RsaSha1
[Comp Key]	PSha1
[Max SKL]	256
[Min AKL]	1024
[Max AKL]	4096

1677 This table defines additional properties whose values can be specified along with the default value for that
1678 property.

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

1679 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

6.2 [Timestamp] Property

1680

1681 This boolean property specifies whether a `wsu:Timestamp` element is present in the `wsse:Security`
1682 header. If the value is 'true', the timestamp element MUST be present and MUST be integrity protected
1683 either by transport or message level security. If the value is 'false', the timestamp element MUST NOT be
1684 present. The default value for this property is 'false'.

6.3 [Protection Order] Property

1685

1686 This property indicates the order in which integrity and confidentiality are applied to the message, in
1687 cases where both integrity and confidentiality are required:

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

1688 The default value for this property is 'SignBeforeEncrypting'.

6.4 [Signature Protection] Property

1689

1690 This boolean property specifies whether the signature must be encrypted. If the value is 'true', the primary
1691 signature MUST be encrypted and any signature confirmation elements MUST also be encrypted. The
1692 primary signature element is not required to be encrypted if the value is 'true' when there is nothing else
1693 in the message that is encrypted. If the value is 'false', the primary signature MUST NOT be encrypted
1694 and any signature confirmation elements MUST NOT be encrypted. The default value for this property is
1695 'false'.

6.5 [Token Protection] Property

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

6.6 [Entire Header and Body Signatures] Property

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

6.7 [Security Header Layout] Property

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

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

6.7.1 Strict Layout Rules for WSS 1.0

1. Tokens that are included in the message MUST be declared before use. For example:
   a. A local signing token MUST occur before the signature that uses it.
   b. A local token serving as the source token for a derived key token MUST occur before that derived key token.

1723   c. A local encryption token MUST occur before the reference list that points to
1724   xenc:EncryptedData elements that use it.

1725   d. If the same token is used for both signing and encryption, then it should appear before
1726   the ds:Signature and xenc:ReferenceList elements in the security header that are
1727   generated using the token.

1728   2. Signed elements inside the security header MUST occur before the signature that signs them.
1729   For example:

1730   a. A timestamp MUST occur before the signature that signs it.

1731   b. A Username token (usually in encrypted form) MUST occur before the signature that
1732   signs it.

1733   c. A primary signature MUST occur before the supporting token signature that signs the
1734   primary signature's signature value element.

1735   3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1736   has the same order requirements as the source plain text element, unless requirement 4
1737   indicates otherwise. For example, an encrypted primary signature MUST occur before any
1738   supporting token signature per 2.c above and an encrypted token has the same ordering
1739   requirements as the unencrypted token.

1740 If there are any encrypted elements in the message then a top level xenc:ReferenceList element or a top
1741 level xenc:EncryptedKey element which contains an xenc:ReferenceList element MUST be present in the
1742 security header. The xenc:ReferenceList or xenc:EncryptedKey MUST occur before any
1743 xenc:EncryptedData elements in the security header that are referenced from the reference list. Strict
1744 Layout Rules for WSS 1.1

1745   1. Tokens that are included in the message MUST be declared before use. For example:

1746   a. A local signing token MUST occur before the signature that uses it.

1747   b. A local token serving as the source token for a derived key token MUST occur before that
1748   derived key token.

1749   c. A local encryption token MUST occur before the reference list that points to
1750   xenc:EncryptedData elements that use it.

1751   d. If the same token is used for both signing and encryption, then it should appear before
1752   the ds:Signature and xenc:ReferenceList elements in the security header that are
1753   generated using the token.

1754   2. Signed elements inside the security header MUST occur before the signature that signs them.
1755   For example:

1756   a. A timestamp MUST occur before the signature that signs it.

1757   b. A Username token (usually in encrypted form) MUST occur before the signature that
1758   signs it.

1759   c. A primary signature MUST occur before the supporting token signature that signs the
1760   primary signature's signature value element.

1761   d. A wsse11:SignatureConfirmation element MUST occur before the signature that signs it.

1762   3. When an element in a security header is encrypted, the resulting xenc:EncryptedData element
1763   has the same order requirements as the source plain text element, unless requirement 4
1764   indicates otherwise. For example, an encrypted primary signature MUST occur before any
1765   supporting token signature per 2.c above and an encrypted token has the same ordering
1766   requirements as the unencrypted token.

1767   4. If there are any encrypted elements in the message then a top level xenc:ReferenceList element
1768   MUST be present in the security header. The xenc:ReferenceList MUST occur before any
1769   xenc:EncryptedData elements in the security header that are referenced from the reference list.
1770   However, the xenc:ReferenceList is not required to appear before independently encrypted
1771   tokens such as the xenc:EncryptedKey token as defined in WSS.

1772   5. An xenc:EncryptedKey element without an internal reference list [WSS: SOAP Message Security
1773   1.1] MUST obey rule 1 above.

7 Security Binding Assertions

1774

1775 The appropriate representation of the different facets of security mechanisms requires distilling the
1776 common primitives (to enable reuse) and then combining the primitive elements into patterns. The policy
1777 scope of assertions defined in this section is the policy scope of their containing element.

7.1 AlgorithmSuite Assertion

1778

1779 This assertion indicates a requirement for an algorithm suite as defined under the [Algorithm Suite]
1780 property described in Section 6.1. The scope of this assertion is defined by its containing assertion.

1781 **Syntax**

1782
```
<sp:AlgorithmSuite xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
   (<sp:Basic256 ... /> |
    <sp:Basic192 ... /> |
    <sp:Basic128 ... /> |
    <sp:TripleDes ... /> |
    <sp:Basic256Rsa15 ... /> |
    <sp:Basic192Rsa15 ... /> |
    <sp:Basic128Rsa15 ... /> |
    <sp:TripleDesRsa15 ... /> |
    <sp:Basic256Sha256 ... /> |
    <sp:Basic192Sha256 ... /> |
    <sp:Basic128Sha256 ... /> |
    <sp:TripleDesSha256 ... /> |
    <sp:Basic256Sha256Rsa15 ... /> |
    <sp:Basic192Sha256Rsa15 ... /> |
    <sp:Basic128Sha256Rsa15 ... /> |
    <sp:TripleDesSha256Rsa15 ... /> |
    ...)
    <sp:InclusiveC14N ... /> ?
    <sp:SOAPNormalization10 ... /> ?
    <sp:STRTransform10 ... /> ?
   (<sp:XPath10 ... /> |
    <sp:XPathFilter20 ... /> |
    <sp:AbsXPath ... /> |
    ...)?
    ...
  </wsp:Policy>
  ...
</sp:AlgorithmSuite>
```

1812

1813 The following describes the attributes and elements listed in the schema outlined above:

1814 /sp:AlgorithmSuite

1815   This identifies an AlgorithmSuite assertion.

1816 /sp:AlgorithmSuite/wsp:Policy

1817   This required element contains one or more policy assertions that indicate the specific algorithm
1818   suite to use.

1819 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256

1820   This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1821   to 'Basic256'.

1822 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192

| 1823 | This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set |
| 1824 | to 'Basic192'. |

1825 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128

1826 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1827 to 'Basic128'.

1828 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDes

1829 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1830 to 'TripleDes'.

1831 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Rsa15

1832 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1833 to 'Basic256Rsa15'.

1834 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Rsa15

1835 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1836 to 'Basic192Rsa15'.

1837 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Rsa15

1838 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1839 to 'Basic128Rsa15'.

1840 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesRsa15

1841 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1842 to 'TripleDesRsa15'.

1843 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256

1844 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1845 to 'Basic256Sha256'.

1846 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256

1847 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1848 to 'Basic192Sha256'.

1849 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256

1850 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1851 to 'Basic128Sha256'.

1852 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256

1853 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1854 to 'TripleDesSha256'.

1855 /sp:AlgorithmSuite/wsp:Policy/sp:Basic256Sha256Rsa15

1856 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1857 to 'Basic256Sha256Rsa15'.

1858 /sp:AlgorithmSuite/wsp:Policy/sp:Basic192Sha256Rsa15

1859 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1860 to 'Basic192Sha256Rsa15'.

1861 /sp:AlgorithmSuite/wsp:Policy/sp:Basic128Sha256Rsa15

1862 This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1863 to 'Basic128Sha256Rsa15'.

1864 /sp:AlgorithmSuite/wsp:Policy/sp:TripleDesSha256Rsa15

1865   This optional element is a policy assertion that indicates that the [Algorithm Suite] property is set
1866   to 'TripleDesSha256Rsa15'.

1867 /sp:AlgorithmSuite/wsp:Policy/sp:InclusiveC14N

1868   This optional element is a policy assertion that indicates that the [C14N] property of an algorithm
1869   suite is set to 'C14N'. Note: as indicated in Section 6.1 the default value of the [C14N] property is
1870   'ExcC14N'.

1871 /sp:AlgorithmSuite/wsp:Policy/sp:SoapNormalization10

1872   This optional element is a policy assertion that indicates that the [SOAP Norm] property is set to
1873   'SNT'.

1874 /sp:AlgorithmSuite/wsp:Policy/sp:STRTransform10

1875   This optional element is a policy assertion that indicates that the [STR Transform] property is set
1876   to 'STRT10'.

1877 /sp:AlgorithmSuite/wsp:Policy/sp:XPath10

1878   This optional element is a policy assertion that indicates that the [XPath] property is set to 'XPath'.

1879 /sp:AlgorithmSuite/wsp:Policy/sp:XPathFilter20

1880   This optional element is a policy assertion that indicates that the [XPath] property is set to
1881   'XPath20'.

1882 /sp:AlgorithmSuite/wsp:Policy/sp:AbsXPath

1883   This optional element is a policy assertion that indicates that the [XPath] property is set to
1884   'AbsXPath' (see AbsoluteLocationPath in [XPATH]).

1885

7.2 Layout Assertion

1887 This assertion indicates a requirement for a particular security header layout as defined under the
1888 [Security Header Layout] property described in Section 6.7. The scope of this assertion is defined by its
1889 containing assertion.

1890 **Syntax**

```
1891    <sp:Layout xmlns:sp="..." ... >
1892      <wsp:Policy xmlns:wsp="...">
1893        <sp:Strict ... /> |
1894        <sp:Lax ... /> |
1895        <sp:LaxTsFirst ... /> |
1896        <sp:LaxTsLast ... /> |
1897        ...
1898      </wsp:Policy>
1899      ...
1900    </sp:Layout>
```

1901

1902 The following describes the attributes and elements listed in the schema outlined above:

1903 /sp:Layout

1904   This identifies a Layout assertion.

1905 /sp:Layout/wsp:Policy

1906   This required element contains one or more policy assertions that indicate the specific security
1907   header layout to use.

1908 /sp:Layout/wsp:Policy/sp:Strict

1909   This optional element is a policy assertion that indicates that the [Security Header Layout]
1910   property is set to 'Strict'.

1911 /sp:Layout/wsp:Policy/sp:Lax

1912   This optional element is a policy assertion that indicates that the [Security Header Layout]
1913   property is set to 'Lax'.

1914 /sp:Layout/wsp:Policy/sp:LaxTsFirst

1915   This optional element is a policy assertion that indicates that the [Security Header Layout]
1916   property is set to 'LaxTimestampFirst'. Note that the [Timestamp] property MUST also be set to
1917   'true' by the presence of an sp:IncludeTimestamp assertion.

1918 /sp:Layout/wsp:Policy/sp:LaxTsLast

1919   This optional element is a policy assertion that indicates that the [Security Header Layout]
1920   property is set to 'LaxTimestampLast'. Note that the [Timestamp] property MUST also be set to
1921   'true' by the presence of an sp:IncludeTimestamp assertion.

7.3 TransportBinding Assertion

1922

1923 The TransportBinding assertion is used in scenarios in which message protection and security correlation
1924 is provided by means other than WSS: SOAP Message Security, for example by a secure transport like
1925 HTTPS. Specifically, this assertion indicates that the message is protected using the means provided by
1926 the transport. This binding has one binding specific token property; [Transport Token]. This assertion
1927 MUST apply to [Endpoint Policy Subject].

1928 **Syntax**

```
1929  <sp:TransportBinding xmlns:sp="..." ... >
1930    <wsp:Policy xmlns:wsp="...">
1931      <sp:TransportToken ... >
1932        <wsp:Policy> ... </wsp:Policy>
1933        ...
1934      </sp:TransportToken>
1935      <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1936      <sp:Layout ... > ... </sp:Layout> ?
1937      <sp:IncludeTimestamp ... /> ?
1938      ...
1939    </wsp:Policy>
1940    ...
1941  </sp:TransportBinding>
```

1942

1943 The following describes the attributes and elements listed in the schema outlined above:

1944 /sp:TransportBinding

1945   This identifies a TransportBinding assertion.

1946 /sp:TransportBinding/wsp:Policy

1947   This indicates a nested `wsp:Policy` element that defines the behavior of the TransportBinding
1948   assertion.

1949 /sp:TransportBinding/wsp:Policy/sp:TransportToken

1950   This required element is a policy assertion that indicates a requirement for a Transport Token.
1951   The specified token populates the [Transport Token] property and indicates how the transport is
1952   secured.

1953 /sp:TransportBinding/wsp:Policy/sp:TransportToken/wsp:Policy

1954   This indicates a nested policy that identifies the type of Transport Token to use.

1955 /sp:TransportBinding/wsp:Policy/sp:AlgorithmSuite

1956 This required element is a policy assertion that indicates a value that populates the [Algorithm
1957 Suite] property. See Section 6.1 for more details.

1958 /sp:TransportBinding/wsp:Policy/sp:Layout

1959 This optional element is a policy assertion that indicates a value that populates the [Security
1960 Header Layout] property. See Section 6.7 for more details.

1961 /sp:TransportBinding/wsp:Policy/sp:IncludeTimestamp

1962 This optional element is a policy assertion that indicates that the [Timestamp] property is set to
1963 'true'.

7.4 SymmetricBinding Assertion

1964

1965 The SymmetricBinding assertion is used in scenarios in which message protection is provided by means
1966 defined in WSS: SOAP Message Security. This binding has two binding specific token properties;
1967 [Encryption Token] and [Signature Token]. If the message pattern requires multiple messages, this
1968 binding defines that the [Encryption Token] used from initiator to recipient is also used from recipient to
1969 initiator. Similarly, the [Signature Token] used from initiator to recipient is also use from recipient to
1970 initiator. If a sp:ProtectionToken assertion is specified, the specified token populates both token
1971 properties and is used as the basis for both encryption and signature in both directions. This assertion
1972 SHOULD apply to [Endpoint Policy Subject]. This assertion MAY apply to [Operation Policy Subject].

1973 **Syntax**

```
1974   <sp:SymmetricBinding xmlns:sp="..." ... >
1975     <wsp:Policy xmlns:wsp="...">
1976       (
1977         <sp:EncryptionToken ... >
1978           <wsp:Policy> ... </wsp:Policy>
1979         </sp:EncryptionToken>
1980         <sp:SignatureToken ... >
1981           <wsp:Policy> ... </wsp:Policy>
1982         </sp:SignatureToken>
1983       ) | (
1984         <sp:ProtectionToken ... >
1985           <wsp:Policy> ... </wsp:Policy>
1986         </sp:ProtectionToken>
1987       )
1988       <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
1989       <sp:Layout ... > ... </sp:Layout> ?
1990       <sp:IncludeTimestamp ... /> ?
1991       <sp:EncryptBeforeSigning ... /> ?
1992       <sp:EncryptSignature ... /> ?
1993       <sp:ProtectTokens ... /> ?
1994       <sp:OnlySignEntireHeadersAndBody ... /> ?
1995       ...
1996     </wsp:Policy>
1997     ...
1998   </sp:SymmetricBinding>
```

1999

2000 The following describes the attributes and elements listed in the schema outlined above:

2001 /sp:SymmetricBinding

2002 This identifies a SymmetricBinding assertion.

2003 /sp:SymmetricBinding/wsp:Policy

2004 This indicates a nested wsp:Policy element that defines the behavior of the SymmetricBinding
2005 assertion.

2006 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken

2007   This optional element is a policy assertion that indicates a requirement for an Encryption Token.
2008   The specified token populates the [Encryption Token] property and is used for encryption. It is an
2009   error for both an sp:EncryptionToken and an sp:ProtectionToken assertion to be specified.

2010 /sp:SymmetricBinding/wsp:Policy/sp:EncryptionToken/wsp:Policy

2011   The policy contained here MUST identify exactly one token to use for encryption.

2012 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken

2013   This optional element is a policy assertion that indicates a requirement for a Signature Token.
2014   The specified token populates the [Signature Token] property and is used for the message
2015   signature. It is an error for both an sp:SignatureToken and an sp:ProtectionToken assertion to be
2016   specified.

2017 /sp:SymmetricBinding/wsp:Policy/sp:SignatureToken/wsp:Policy

2018   The policy contained here MUST identify exactly one token to use for signatures.

2019 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken

2020   This optional element is a policy assertion that indicates a requirement for a Protection Token.
2021   The specified token populates the [Encryption Token] and [Signature Token properties] and is
2022   used for the message signature and for encryption. It is an error for both an sp:ProtectionToken
2023   assertion and either an sp:EncryptionToken assertion or an sp:SignatureToken assertion to be
2024   specified.

2025 /sp:SymmetricBinding/wsp:Policy/sp:ProtectionToken/wsp:Policy

2026   The policy contained here MUST identify exactly one token to use for protection.

2027 /sp:SymmetricBinding/wsp:Policy/sp:AlgorithmSuite

2028   This required element is a policy assertion that indicates a value that populates the [Algorithm
2029   Suite] property. See Section 6.1 for more details.

2030 /sp:SymmetricBinding/wsp:Policy/sp:Layout

2031   This optional element is a policy assertion that indicates a value that populates the [Security
2032   Header Layout] property. See Section 6.7 for more details.

2033 /sp:SymmetricBinding/wsp:Policy/sp:IncludeTimestamp

2034   This optional element is a policy assertion that indicates that the [Timestamp] property is set to
2035   'true'.

2036 /sp:SymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

2037   This optional element is a policy assertion that indicates that the [Protection Order] property is set
2038   to 'EncryptBeforeSigning'.

2039 /sp:SymmetricBinding/wsp:Policy/sp:EncryptSignature

2040   This optional element is a policy assertion that indicates that the [Signature Protection] property is
2041   set to 'true'.

2042 /sp:SymmetricBinding/wsp:Policy/sp:ProtectTokens

2043   This optional element is a policy assertion that indicates that the [Token Protection] property is
2044   set to 'true'.

2045 /sp:SymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

2046   This optional element is a policy assertion that indicates that the [Entire Header And Body
2047   Signatures] property is set to 'true'.

7.5 AsymmetricBinding Assertion

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

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

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

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

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

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

Syntax

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

```
2100          </sp:RecipientEncryptionToken>
2101        )
2102      <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite>
2103      <sp:Layout ... > ... </sp:Layout> ?
2104      <sp:IncludeTimestamp ... /> ?
2105      <sp:EncryptBeforeSigning ... /> ?
2106      <sp:EncryptSignature ... /> ?
2107      <sp:ProtectTokens ... /> ?
2108      <sp:OnlySignEntireHeadersAndBody ... /> ?
2109      ...
2110    </wsp:Policy>
2111    ...
2112  </sp:AsymmetricBinding>
```

2113

2114   The following describes the attributes and elements listed in the schema outlined above:

2115   /sp:AsymmetricBinding

2116   This identifies a AsymmetricBinding assertion.

2117   /sp:AsymmetricBinding/wsp:Policy

2118   This indicates a nested wsp:Policy element that defines the behavior of the AsymmetricBinding
2119   assertion.

2120   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken

2121   This optional element is a policy assertion that indicates a requirement for an Initiator Token. The
2122   specified token populates the [Initiator Signature Token] and [Initiator Encryption Token]
2123   properties and is used for the message signature from initiator to recipient, and encryption from
2124   recipient to initiator.

2125   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy

2126   The policy contained here MUST identify one or more token assertions.

2127   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken

2128   This optional element is a policy assertion that indicates a requirement for an Initiator Signature
2129   Token. The specified token populates the [Initiator Signature Token] property and is used for the
2130   message signature from initiator to recipient.

2131   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorSignatureToken/wsp:Policy

2132   The policy contained here MUST identify one or more token assertions.

2133   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken

2134   This optional element is a policy assertion that indicates a requirement for an Initiator Encryption
2135   Token. The specified token populates the [Initiator Encryption Token] property and is used for the
2136   message encryption from recipient to initiator.

2137   /sp:AsymmetricBinding/wsp:Policy/sp:InitiatorToken/wsp:Policy/sp:InitiatorEncryptionToken/wsp:Policy

2138   The policy contained here MUST identify one or more token assertions.

2139   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken

2140   This optional element is a policy assertion that indicates a requirement for a Recipient Token. The
2141   specified token populates the [Recipient Signature Token] and [Recipient Encryption Token]
2142   property and is used for encryption from initiator to recipient, and for the message signature from
2143   recipient to initiator.

2144   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy

2145   The policy contained here MUST identify one or more token assertions.

2146   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken

2147   This optional element is a policy assertion that indicates a requirement for a Recipient Signature
2148   Token. The specified token populates the [Recipient Signature Token] property and is used for
2149   the message signature from Recipient to recipient.

2150   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientSignatureToken/wsp:Policy

2151   The policy contained here MUST identify one or more token assertions.

2152   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken

2153   This optional element is a policy assertion that indicates a requirement for a Recipient Encryption
2154   Token. The specified token populates the [Recipient Encryption Token] property and is used for
2155   the message encryption from recipient to Recipient.

2156   /sp:AsymmetricBinding/wsp:Policy/sp:RecipientToken/wsp:Policy/sp:RecipientEncryptionToken/wsp:Policy

2157   The policy contained here MUST identify one or more token assertions.

2158   /sp:AsymmetricBinding/wsp:Policy/sp:AlgorithmSuite

2159   This required element is a policy assertion that indicates a value that populates the [Algorithm
2160   Suite] property. See Section 6.1 for more details.

2161   /sp:AsymmetricBinding/wsp:Policy/sp:Layout

2162   This optional element is a policy assertion that indicates a value that populates the [Security
2163   Header Layout] property. See Section 6.7 for more details.

2164   /sp:AsymmetricBinding/wsp:Policy/sp:IncludeTimestamp

2165   This optional element is a policy assertion that indicates that the [Timestamp] property is set to
2166   'true'.

2167   /sp:AsymmetricBinding/wsp:Policy/sp:EncryptBeforeSigning

2168   This optional element is a policy assertion that indicates that the [Protection Order] property is set
2169   to 'EncryptBeforeSigning'.

2170   /sp:AsymmetricBinding/wsp:Policy/sp:EncryptSignature

2171   This optional element is a policy assertion that indicates that the [Signature Protection] property is
2172   set to 'true'.

2173   /sp:AsymmetricBinding/wsp:Policy/sp:ProtectTokens

2174   This optional element is a policy assertion that indicates that the [Token Protection] property is
2175   set to 'true'.

2176   /sp:AsymmetricBinding/wsp:Policy/sp:OnlySignEntireHeadersAndBody

2177   This optional element is a policy assertion that indicates that the [Entire Header And Body
2178   Signatures] property is set to 'true'.

8 Supporting Tokens

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

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

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

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

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

2216 Even before any supporting tokens are added, each binding requires that the message is signed using a
2217 token satisfying the required usage for that binding, and that the signature (Sig1) covers important parts
2218 of the message including the message timestamp (TS) facilitate replay detection. The signature is then
2219 included as part of the Security header as illustrated below:

2220

2221

2222 Note: if required, the initiator may also include in the Security header the token used as the basis for the
2223 message signature (Sig1), not shown in the diagram.

2224 If transport security is used, only the message timestamp (TS) is included in the Security header as
2225 illustrated below.. The "message signature" is provided by the underlying transport protocol and is not
2226 part of the message XML.

2227

8.1 SupportingTokens Assertion

2229 Supporting tokens are included in the security header and may optionally include additional message
2230 parts to sign and/or encrypt. The supporting tokens can be added to any SOAP message and do not
2231 require any protection (signature or encryption) to be applied to the message before they are added.
2232 More specifically there is no requirement on "message signature" being present before the supporting
2233 tokens are added. However it is RECOMMENDED to employ underlying protection mechanism to ensure
2234 that the supporting tokens are cryptographically bound to the message during the transmission.

2235 **Syntax**

```
2236   <sp:SupportingTokens xmlns:sp="..." ... >
2237     <wsp:Policy xmlns:wsp="...">
2238       [Token Assertion]+
2239       <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2240       (
2241         <sp:SignedParts ... > ... </sp:SignedParts> |
2242         <sp:SignedElements ... > ... </sp:SignedElements> |
```

```
2243          <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2244          <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2245        ) *
2246        ...
2247      </wsp:Policy>
2248      ...
2249    </sp:SupportingTokens>
```

2250

2251   The following describes the attributes and elements listed in the schema outlined above:

2252   /sp:SupportingTokens

2253   This identifies a SupportingTokens assertion. The specified tokens populate the [Supporting
2254   Tokens] property.

2255   /sp:SupportingTokens/wsp:Policy

2256   This describes additional requirements for satisfying the SupportingTokens assertion.

2257   /sp:SupportingTokens/wsp:Policy/[Token Assertion]

2258   The policy MUST identify one or more token assertions.

2259   /sp:SupportingTokens/wsp:Policy/sp:AlgorithmSuite

2260   This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2261   describes the algorithms to use for cryptographic operations performed with the tokens identified
2262   by this policy assertion.

2263   /sp:SupportingTokens/wsp:Policy/sp:SignedParts

2264   This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2265   describes additional message parts that MUST be included in the signature generated with the
2266   token identified by this policy assertion.

2267   /sp:SupportingTokens/wsp:Policy/sp:SignedElements

2268   This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
2269   describes additional message elements that MUST be included in the signature generated with
2270   the token identified by this policy assertion.

2271   /sp:SupportingTokens/wsp:Policy/sp:EncryptedParts

2272   This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2273   describes additional message parts that MUST be encrypted using the token identified by this
2274   policy assertion.

2275   /sp:SupportingTokens/wsp:Policy/sp:EncryptedElements

2276   This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2277   describes additional message elements that MUST be encrypted using the token identified by this
2278   policy assertion.

8.2 SignedSupportingTokens Assertion

2280   Signed tokens are included in the "message signature" as defined above and may optionally include
2281   additional message parts to sign and/or encrypt. The diagram below illustrates how the attached token
2282   (Tok2) is signed by the message signature (Sig1):

2283

2284

Header1
Header2
Security
  TS
  Tok2
  Sig1
Body

2285 If transport security is used, the token (Tok2) is included in the Security header as illustrated below:

2286

Header1
Header2
Security
  TS
  Tok2
Body

2287

2288 **Syntax**

```
2289  <sp:SignedSupportingTokens xmlns:sp="..." ... >
2290    <wsp:Policy xmlns:wsp="...">
2291      [Token Assertion]+
2292      <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
2293      (
2294        <sp:SignedParts ... > ... </sp:SignedParts> |
2295        <sp:SignedElements ... > ... </sp:SignedElements> |
2296        <sp:EncryptedParts ... > ... </sp:EncryptedParts> |
2297        <sp:EncryptedElements ... > ... </sp:EncryptedElements> |
2298      ) *
2299      ...
2300    </wsp:Policy>
2301    ...
2302  </sp:SignedSupportingTokens>
```

2303

2304 The following describes the attributes and elements listed in the schema outlined above:

2305 /sp:SignedSupportingTokens

2306   This identifies a SignedSupportingTokens assertion. The specified tokens populate the [Signed
2307   Supporting Tokens] property.

2308 /sp:SignedSupportingTokens/wsp:Policy

2309   This describes additional requirements for satisfying the SignedSupportingTokens assertion.

2310   /sp:SignedSupportingTokens/wsp:Policy/[Token Assertion]

2311   The policy MUST identify one or more token assertions.

2312   /sp:SignedSupportingTokens/wsp:Policy/sp:AlgorithmSuite

2313   This optional element is a policy assertion that follows the schema outlined in Section 7.1 and
2314   describes the algorithms to use for cryptographic operations performed with the tokens identified
2315   by this policy assertion.

2316   /sp:SignedSupportingTokens/wsp:Policy/sp:SignedParts

2317   This optional element is a policy assertion that follows the schema outlined in Section 4.1.1 and
2318   describes additional message parts that MUST be included in the signature generated with the
2319   token identified by this policy assertion.

2320   /sp:SignedSupportingTokens/wsp:Policy/sp:SignedElements

2321   This optional element is a policy assertion that follows the schema outlined in Section 4.1.2 and
2322   describes additional message elements that MUST be included in the signature generated with
2323   the token identified by this policy assertion.

2324   /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedParts

2325   This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2326   describes additional message parts that MUST be encrypted using the token identified by this
2327   policy assertion.

2328   /sp:SignedSupportingTokens/wsp:Policy/sp:EncryptedElements

2329   This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2330   describes additional message elements that MUST be encrypted using the token identified by this
2331   policy assertion.

8.3 EndorsingSupportingTokens Assertion

2333   Endorsing tokens sign the message signature, that is they sign the entire `ds:Signature` element
2334   produced from the message signature and may optionally include additional message parts to sign and/or
2335   encrypt. The diagram below illustrates how the endorsing signature (Sig2) signs the message signature
2336   (Sig1):

2337

Header1

Header2

Security

TS

Sig1

Sig2

Body

2338

2339   If transport security is used, the signature (Sig2) MUST cover the message timestamp as illustrated
2340   below:

2341

```
                    ┌─────────────────┐
                    │  ┌───────────┐  │
                    │  │  Header1  │  │
                    │  └───────────┘  │
                    │  ┌───────────┐  │
                    │  │  Header2  │  │
                    │  └───────────┘  │
                    │  ┌───────────┐  │
                    │  │ Security  │  │
                    │  │ ┌───────┐ │  │
                    │→ │ │  TS   │ │  │
                    │  │ └───────┘ │  │
                    │  │ ┌───────┐ │  │
                    │  │ │  Sig2 │ │  │
                    │  │ └───────┘ │  │
                    │  └───────────┘  │
                    │  ┌───────────┐  │
                    │  │   Body    │  │
                    │  └───────────┘  │
                    └─────────────────┘
```

2342

Syntax

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

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

/sp:EndorsingSupportingTokens

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

/sp:EndorsingSupportingTokens/wsp:Policy

This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

The policy MUST identify one or more token assertions.

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

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

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

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

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

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

2379 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedParts

2380   This optional element is a policy assertion that follows the schema outlined in Section 4.2.1 and
2381   describes additional message parts that MUST be encrypted using the token identified by this
2382   policy assertion.

2383 /sp:EndorsingSupportingTokens/wsp:Policy/sp:EncryptedElements

2384   This optional element is a policy assertion that follows the schema outlined in Section 4.2.2 and
2385   describes additional message elements that MUST be encrypted using the token identified by this
2386   policy assertion.

8.4 SignedEndorsingSupportingTokens Assertion

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

2394

2395

2396 If transport security is used, the token (Tok2) is included in the Security header and the signature (Sig2)
2397 should cover the message timestamp as illustrated below:

2398

2399

Syntax

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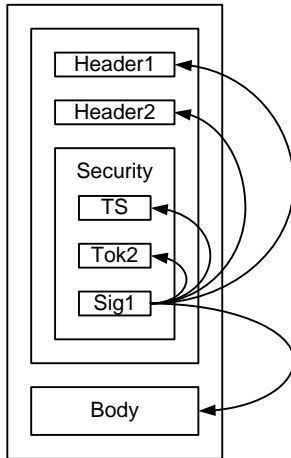<sp:SignedEndorsingSupportingTokens xmlns:sp="..." ... >
  <wsp:Policy xmlns:wsp="...">
    [Token Assertion]+
    <sp:AlgorithmSuite ... > ... </sp:AlgorithmSuite> ?
    (
      <sp:SignedParts ... > ... </sp:SignedParts> |
      <sp:SignedElements ... > ... </sp:SignedElements> |
      <sp:EncryptedParts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

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

/sp:SignedEndorsingSupportingTokens

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

/sp:SignedEndorsingSupportingTokens/wsp:Policy

> This describes additional requirements for satisfying the EndorsingSupportingTokens assertion.

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

> The policy MUST identify one or more token assertions.

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

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

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

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

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

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

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

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

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

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

8.5 SignedEncryptedSupportingTokens Assertion

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

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

8.6 EncryptedSupportingTokens Assertion

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

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

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

8.68.7 EndorsingEncryptedSupportingTokens Assertion

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

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

8.78.8 SignedEndorsingEncryptedSupportingTokens Assertion

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

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

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

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

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

2486 • For signed, endorsing supporting tokens, the supporting token is signed twice, once by the
2487   message signature and once by the endorsing signature.

2488 In addition, signed supporting tokens are covered by the message signature, although this is independent
2489 of [Token Protection].

2490 8.98.10 Example

2491 Example policy containing supporting token assertions:

```
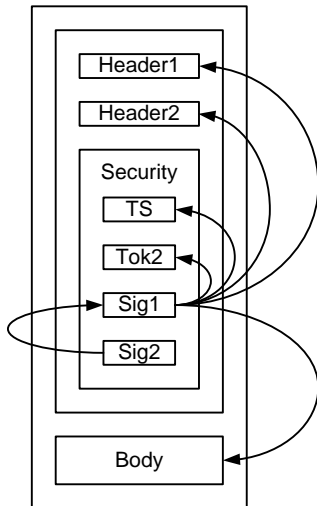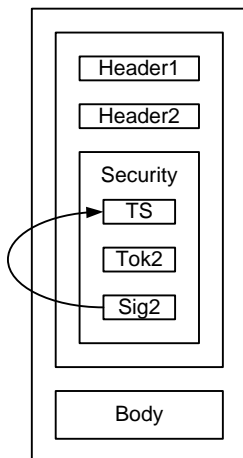2492  <!-- Example Endpoint Policy -->
2493  <wsp:Policy xmlns:wsp="...">
2494    <sp:SymmetricBinding xmlns:sp="...">
2495      <wsp:Policy>
2496        <sp:ProtectionToken>
2497          <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
2498            <sp:Issuer>...</sp:Issuer>
2499            <sp:RequestSecurityTokenTemplate>
2500            ...
2501            </sp:RequestSecurityTokenTemplate>
2502          </sp:IssuedToken>
2503        </sp:ProtectionToken>
2504        <sp:AlgorithmSuite>
2505          <wsp:Policy>
2506            <sp:Basic256 />
2507          </wsp:Policy>
2508        </sp:AlgorithmSuite>
2509        ...
2510      </wsp:Policy>
2511    </sp:SymmetricBinding>
2512    ...
2513    <sp:SignedSupportingTokens>
2514      <wsp:Policy>
2515        <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
2516      </wsp:Policy>
2517    </sp:SignedSupportingTokens>
2518    <sp:SignedEndorsingSupportingTokens>
2519      <wsp:Policy>
2520        <sp:X509Token sp:IncludeToken=".../IncludeToken/Once" >
2521          <wsp:Policy>
2522            <sp:WssX509v3Token10 />
2523          </wsp:Policy>
2524        </sp:X509Token>
2525      </wsp:Policy>
2526    </sp:SignedEndorsingSupportingTokens>
2527    ...
2528  </wsp:Policy>
```

2529 The sp:SignedSupportingTokens assertion in the above policy indicates that a Username Token must be
2530 included in the security header and covered by the message signature. The
2531 sp:SignedEndorsingSupportingTokens assertion indicates that an X509 certificate must be included in the
2532 security header and covered by the message signature. In addition, a signature over the message
2533 signature based on the key material associated with the X509 certificate must be included in the security
2534 header.

9 WSS: SOAP Message Security Options

2535

2536 There are several optional aspects to the WSS: SOAP Message Security specification that are
2537 independent of the trust and token taxonomies. This section describes another class of properties and
2538 associated assertions that indicate the supported aspects of WSS: SOAP Message Security. The
2539 assertions defined here MUST apply to [Endpoint Policy Subject].

2540 The properties and assertions dealing with token references defined in this section indicate whether the
2541 initiator and recipient MUST be able to process a given reference mechanism, or whether the initiator and
2542 recipient MAY send a fault if such references are encountered.

2543

2544 Note: This approach is chosen because:

2545   A) [WSS: SOAP Message Security] allows for multiple equivalent reference mechanisms to be used
2546   in a single reference.
2547   B) In a multi-message exchange, a token may be referenced using different mechanisms depending
2548   on which of a series of messages is being secured.

2549

2550 If a message sent to a recipient does not adhere to the recipient's policy the recipient MAY raise a
2551 `wsse:InvalidSecurity` fault.

2552

2553 **WSS: SOAP Message Security 1.0 Properties**

2554 **[Direct References]**

2555 This property indicates whether the initiator and recipient MUST be able to process direct token
2556 references (by ID or URI reference). This property always has a value of 'true'. i.e. All implementations
2557 MUST be able to process such references.

2558

2559 **[Key Identifier References]**

2560 This boolean property indicates whether the initiator and recipient MUST be able to process key-specific
2561 identifier token references. A value of 'true' indicates that the initiator and recipient MUST be able to
2562 generate and process such references. A value of 'false' indicates that the initiator and recipient MUST
2563 NOT generate such references and that the initiator and recipient MAY send a fault if such references are
2564 encountered. This property has a default value of 'false'.

2565

2566 **[Issuer Serial References]**

2567 This boolean property indicates whether the initiator and recipient MUST be able to process references
2568 using the issuer and token serial number. A value of 'true' indicates that the initiator and recipient MUST
2569 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT
2570 generate such references and that the initiator and recipient MAY send a fault if such references are
2571 encountered. This property has a default value of 'false'.

2572

2573 **[External URI References]**

2574 This boolean property indicates whether the initiator and recipient MUST be able to process references to
2575 tokens outside the message using URIs. A value of 'true' indicates that the initiator and recipient MUST
2576 be able to process such references. A value of 'false' indicates that the initiator and recipient MUST NOT

2577 generate such references and that the initiator and recipient MAY send a fault if such references are
2578 encountered. This property has a default value of 'false'.

2579 **[Embedded Token References]**

2580 This boolean property indicates whether the initiator and recipient MUST be able to process references
2581 that contain embedded tokens. A value of 'true' indicates that the initiator and recipient MUST be able to
2582 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2583 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2584 This property has a default value of 'false'.

2585

2586 <u>**WSS: SOAP Message Security 1.1 Properties**</u>

2587 **[Thumbprint References]**

2588 This boolean property indicates whether the initiator and recipient MUST be able to process references
2589 using token thumbprints. A value of 'true' indicates that the initiator and recipient MUST be able to
2590 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2591 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2592 This property has a default value of 'false'.

2593

2594 **[EncryptedKey References]**

2595 This boolean property indicates whether the initiator and recipient MUST be able to process references
2596 using EncryptedKey references. A value of 'true' indicates that the initiator and recipient MUST be able to
2597 process such references. A value of 'false' indicates that the initiator and recipient MUST NOT generate
2598 such references and that the initiator and recipient MAY send a fault if such references are encountered.
2599 This property has a default value of 'false'.

2600

2601 **[Signature Confirmation]**

2602 This boolean property specifies whether `wsse11:SignatureConfirmation` elements should be used
2603 as defined in WSS: Soap Message Security 1.1. If the value is 'true',
2604 `wsse11:SignatureConfirmation` elements MUST be used and signed by the message signature. If
2605 the value is 'false', signature confirmation elements MUST NOT be used. The value of this property
2606 applies to all signatures that are included in the security header. This property has a default value of
2607 'false'.

9.1 Wss10 Assertion

2609 The Wss10 assertion allows you to specify which WSS: SOAP Message Security 1.0 options are
2610 supported.

2611 **Syntax**

```
2612 <sp:Wss10 xmlns:sp="..." ... >
2613   <wsp:Policy xmlns:wsp="...">
2614     <sp:MustSupportRefKeyIdentifier  ... /> ?
2615     <sp:MustSupportRefIssuerSerial  ... /> ?
2616     <sp:MustSupportRefExternalURI  ... /> ?
2617     <sp:MustSupportRefEmbeddedToken  ... /> ?
2618     ...
2619   </wsp:Policy>
2620   ...
2621 </sp:Wss10>
```

2622

2623 The following describes the attributes and elements listed in the schema outlined above:

2624 /sp:Wss10

2625   This identifies a WSS10 assertion.

2626 /sp:Wss10/wsp:Policy

2627

2628   This indicates a policy that controls WSS: SOAP Message Security 1.0 options.

2629 /sp:Wss10/wsp:Policy/sp:MustSupportRefKeyIdentifier

2630   This optional element is a policy assertion indicates that the [Key Identifier References] property
2631   is set to 'true'.

2632 /sp:Wss10/wsp:Policy/sp:MustSupportRefIssuerSerial

2633   This optional element is a policy assertion indicates that the [Issuer Serial References] property is
2634   set to 'true'.

2635 /sp:Wss10/wsp:Policy/sp:MustSupportRefExternalURI

2636   This optional element is a policy assertion indicates that the [External URI References] property is
2637   set to 'true'.

2638 /sp:Wss10/wsp:Policy/sp:MustSupportRefEmbeddedToken

2639   This optional element is a policy assertion indicates that the [Embedded Token References]
2640   property is set to 'true'.

2641 9.2 Wss11 Assertion

2642 The Wss11 assertion allows you to specify which WSS: SOAP Message Security 1.1 options are
2643 supported.

2644 **Syntax**

```
2645  <sp:Wss11 xmlns:sp="..." ... >
2646    <wsp:Policy xmlns:wsp="...">
2647      <sp:MustSupportRefKeyIdentifier  ... /> ?
2648      <sp:MustSupportRefIssuerSerial   ... /> ?
2649      <sp:MustSupportRefExternalURI   ... /> ?
2650      <sp:MustSupportRefEmbeddedToken   ... /> ?
2651      <sp:MustSupportRefThumbprint  ... /> ?
2652      <sp:MustSupportRefEncryptedKey   ... /> ?
2653      <sp:RequireSignatureConfirmation   ... /> ?
2654      ...
2655    </wsp:Policy>
2656  </sp:Wss11>
```

2657

2658 The following describes the attributes and elements listed in the schema outlined above:

2659 /sp:Wss11

2660   This identifies an WSS11 assertion.

2661 /sp:Wss11/wsp:Policy

2662   This indicates a policy that controls WSS: SOAP Message Security 1.1 options.

2663 /sp:Wss11/wsp:Policy/sp:MustSupportRefKeyIdentifier

2664   This optional element is a policy assertion indicates that the [Key Identifier References] property
2665   is set to 'true'.

2666 /sp:Wss11/wsp:Policy/sp:MustSupportRefIssuerSerial

2667   This optional element is a policy assertion indicates that the [Issuer Serial References] property is
2668   set to 'true'.

2669   /sp:Wss11/wsp:Policy/sp:MustSupportRefExternalURI

2670   This optional element is a policy assertion indicates that the [External URI References] property is
2671   set to 'true'.

2672   /sp:Wss11/wsp:Policy/sp:MustSupportRefEmbeddedToken

2673   This optional element is a policy assertion indicates that the [Embedded Token References]
2674   property is set to 'true'.

2675   /sp:Wss11/wsp:Policy/sp:MustSupportRefThumbprint

2676   This optional element is a policy assertion indicates that the [Thumbprint References] property is
2677   set to 'true'.

2678   /sp:Wss11/wsp:Policy/sp:MustSupportRefEncryptedKey

2679   This optional element is a policy assertion indicates that the [EncryptedKey References] property
2680   is set to 'true'.

2681   /sp:Wss11/wsp:Policy/sp:RequireSignatureConfirmation

2682   This optional element is a policy assertion indicates that the [Signature Confirmation] property is
2683   set to 'true'.

2684 10 WS-Trust Options

2685 This section defines the various policy assertions related to exchanges based on WS-Trust, specifically
2686 with client and server challenges and entropy behaviors. These assertions relate to interactions with a
2687 Security Token Service and may augment the behaviors defined by the Binding Property Assertions
2688 defined in Section 6. The assertions defined here MUST apply to [Endpoint Policy Subject].

2689

2690 **WS-Trust 1.~~3~~0 Properties**

2691 **[Client Challenge]**

2692 This boolean property indicates whether client challenges are supported. A value of 'true' indicates that a
2693 `wst:SignChallenge` element is supported inside of an RST sent by the client to the server. A value of
2694 'false' indicates that a `wst:SignChallenge` is not supported. There is no change in the number of
2695 messages exchanged by the client and service in satisfying the RST. This property has a default value of
2696 'false'.

2697

2698 **[Server Challenge]**

2699 This boolean property indicates whether server challenges are supported. A value of 'true' indicates that a
2700 `wst:SignChallenge` element is supported inside of an RSTR sent by the server to the client. A value of
2701 'false' indicates that a `wst:SignChallenge` is not supported. A challenge issued by the server may
2702 increase the number of messages exchanged by the client and service in order to accommodate the
2703 `wst:SignChallengeResponse` element sent by the client to the server in response to the
2704 `wst:SignChallenge` element. A final RSTR containing the issued token will follow subsequent to the
2705 server receiving the `wst:SignChallengeResponse` element. This property has a default value of 'false'.

2706

2707 **[Client Entropy]**

2708 This boolean property indicates whether client entropy is required to be used as key material for a
2709 requested proof token. A value of 'true' indicates that client entropy is required. A value of 'false' indicates
2710 that client entropy is not required. This property has a default value of 'false'.

2711

2712 **[Server Entropy]**

2713 This boolean property indicates whether server entropy is required to be used as key material for a
2714 requested proof token. A value of 'true' indicates that server entropy is required. A value of 'false'
2715 indicates that server entropy is not required. This property has a default value of 'false'.

2716 Note: If both the [Client Entropy] and [Server Entropy] properties are set to true, Client and server entropy
2717 are combined to produce a computed key using the Computed Key algorithm defined by the [Algorithm
2718 Suite] property.

2719

2720 **[Issued Tokens]**

2721 This boolean property indicates whether the `wst:IssuedTokens` header is supported as described in
2722 WS-Trust. A value of 'true' indicates that the wst:IssuedTokens header is supported. A value of 'false'
2723 indicates that the `wst:IssuedTokens` header is not supported. This property has a default value of
2724 'false'.

2725 **[Collection]**

2726   This boolean property specifies whether a wst:RequestSecurityTokenCollection element is present. A
2727   value of 'true' indicates that the wst:RequestSecurityTokenCollection element MUST be present and
2728   MUST be integrity protected either by transport or message level security. A value of 'false' indicates that
2729   the wst:RequestSecurityTokenCollection element MUST NOT be present. This property has a default
2730   value of 'false'.

2731

2732   10.1 Trust1~~3~~0 Assertion

2733   The Trust1~~3~~0 assertion allows you to specify which WS-Trust 1.~~3~~0 options are supported.

2734   **Syntax**

```
2735    <sp:Trust13ø xmlns:sp="..." ... >
2736      <wsp:Policy xmlns:wsp="...">
2737        <sp:MustSupportClientChallenge  ... />?
2738        <sp:MustSupportServerChallenge  ... />?
2739        <sp:RequireClientEntropy  ... />?
2740        <sp:RequireServerEntropy  ... />?
2741        <sp:MustSupportIssuedTokens  ... />?
2742        <sp:RequireRequestSecurityTokenCollection />?
2743        <sp:RequireAppliesTo />?
2744        ...
2745      </wsp:Policy>
2746      ...
2747    </sp:Trust13ø ... >
```

2748

2749   The following describes the attributes and elements listed in the schema outlined above:

2750   /sp:Trust1~~3~~0

2751   This identifies a Trust1~~3~~0 assertion.

2752   /sp:Trust1~~3~~0/wsp:Policy

2753   This indicates a policy that controls WS-Trust 1.~~3~~0 options.

2754   /sp:Trust1~~3~~0/wsp:Policy/sp:MustSupportClientChallenge

2755   2756   This optional element is a policy assertion indicates that the [Client Challenge] property is set to
            'true'.

2757   /sp:Trust1~~3~~0/wsp:Policy/sp:MustSupportServerChallenge

2758   2759   This optional element is a policy assertion indicates that the [Server Challenge] property is set to
            'true'.

2760   /sp:Trust1~~3~~0/wsp:Policy/sp:RequireClientEntropy

2761   2762   This optional element is a policy assertion indicates that the [Client Entropy] property is set to
            'true'.

2763   /sp:Trust1~~3~~0/wsp:Policy/sp:RequireServerEntropy

2764   2765   This optional element is a policy assertion indicates that the [Server Entropy] property is set to
            'true'.

2766   /sp:Trust1~~3~~0/wsp:Policy/sp:MustSupportIssuedTokens

2767   2768   This optional element is a policy assertion indicates that the [Issued Tokens] property is set to
            'true'.

2769   /sp:Trust1~~3~~0/wsp:Policy/sp:RequireRequestSecurityTokenCollection

2770   2771   This optional element is a policy assertion that indicates that the [Collection] property is set to
            'true'.

2772 /sp:Trust10/wsp:Policy/sp:RequireAppliesTo

2773 This optional element is a policy assertion indicates that the STS requires the requestor to specify
2774 the scope for the issued token using wsp:AppliesTo in the RST.

11 Guidance on creating new assertions and assertion extensibility

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

11.1 General Design Points

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

11.2 Detailed Design Guidance

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

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

```
Design 1

   <A1/>
   <A2/>
   <A3/>

Design 2.

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

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

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

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

2819   on the various token assertions. Five possible values are currently specified for the sp:IncludeToken
2820   attribute and implementations are expected to understand the meaning of all 5 values. If this information
2821   was encoded into the assertion QNames, each existing token assertion would require five variants, one
2822   for each Uri value which would result in 45 assertions just for the tokens defined in Section 5.

2823

2824   Nested policy is ideal for encoding parameters that can be usefully matched using policy matching. For
2825   example, the token version assertions defined in Section 5 use such an approach. The overall token type
2826   assertion is parameterized by the nested token version assertions. Policy matching can use these
2827   parameters to find matches between policies where the broad token type is support by both parties but
2828   they might not support the same specific versions.

2829

2830   Note, when designing assertions for new token types such assertions SHOULD allow the
2831   sp:IncludeToken attribute and SHOULD allow nested policy.

2832

12 Security Considerations

2834 It is strongly recommended that policies and assertions be signed to prevent tampering.

2835 It is recommended that policies should not be accepted unless they are signed and have an associated
2836 security token to specify the signer has proper claims for the given policy. That is, a party shouldn't rely
2837 on a policy unless the policy is signed and presented with sufficient claims. It is further recommended that
2838 the entire policy exchange mechanism be protected to prevent man-in-the-middle downgrade attacks.

2839

2840 It should be noted that the mechanisms described in this document could be secured as part of a SOAP
2841 message using WSS: SOAP Message Security [WSS10, WSS11] or embedded within other objects using
2842 object-specific security mechanisms.

2843

2844 It is recommended that policies not specify two (or more) SignedSupportingTokens or
2845 SignedEndorsingSupportingTokens of the same token type. Messages conforming to such policies are
2846 subject to modification which may be undetectable.

2847

2848 It is recommended that policies specify the OnlySignEntireHeadersAndBody assertion along with the rest
2849 of the policy in order to combat certain XML substitution attacks.

A. Assertions and WS-PolicyAttachment

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

A.1 Endpoint Policy Subject Assertions

A.1.1 Security Binding Assertions

TransportBinding Assertion   (Section 7.3)

SymmetricBinding Assertion   (Section 7.4)

AsymmetricBinding Assertion   (Section 7.5)

A.1.2 Token Assertions

SupportingTokens Assertion   (Section 8.1)

SignedSupportingTokens Assertion   (Section 8.2)

EndorsingSupportingTokens Assertion   (Section 8.3)

SignedEndorsingSupportingTokens Assertion   (Section 8.4)

SignedEncryptedSupportingTokens Assertion   (Section 8.5)

EndorsingEncryptedSupportingTokens Assertion   (Section 8.6)

SignedEndorsingEncryptedSupportingTokens Assertion (Section 8.7)

A.1.3 WSS: SOAP Message Security 1.0 Assertions

Wss10 Assertion   (Section 9.1)

A.1.4 WSS: SOAP Message Security 1.1 Assertions

Wss11 Assertion   (Section 9.2)

A.1.5 Trust 1.0 Assertions

Trust13~~0~~ Assertion   (Section 10.1)

A.2 Operation Policy Subject Assertions

A.2.1 Security Binding Assertions

SymmetricBinding Assertion   (Section 7.4)

AsymmetricBinding Assertion   (Section 7.5)

A.2.2 Supporting Token Assertions

SupportingTokens Assertion   (Section 8.1)

SignedSupportingTokens Assertion   (Section 8.2)

2887 A.3 Message Policy Subject Assertions

2888 A.3.1 Supporting Token Assertions

2896 A.3.2 Protection Assertions

2904 A.4 Assertions With Undefined Policy Subject

2905   The assertions listed in this section do not have a defined policy subject because they appear nested
2906   inside some other assertion which does have a defined policy subject. This list is derived from nested
2907   assertions in the specification that have independent sections. It is not a complete list of nested
2908   assertions. Many of the assertions previously listed in this appendix as well as the ones below have
2909   additional nested assertions.

2910 A.4.1 General Assertions

2913 A.4.2 Token Usage Assertions

2914   See the nested assertions under the TransportBinding, SymmetricBinding and AssymetricBinding
2915   assertions.

2916 A.4.3 Token Assertions

B. Issued Token Policy

2927

2928 The section provides further detail about behavior associated with the IssuedToken assertion in section
2929 5.3.2.

2930

2931 The issued token security model involves a three-party setup. There's a target Server, a Client, and a
2932 trusted third party called a Security Token Service or STS. Policy flows from Server to Client, and from
2933 STS to Client. Policy may be embedded inside an Issued Token assertion, or acquired out-of-band. There
2934 may be an explicit trust relationship between the Server and the STS. There must be a trust relationship
2935 between the Client and the STS.

2936

2937 The Issued Token policy assertion includes two parts: 1) client-specific parameters that must be
2938 understood and processed by the client and 2) STS specific parameters which are to be processed by the
2939 STS. The format of the Issued Token policy assertion is illustrated in the figure below.

Issued Token Policy

Client Parameters

STS Parameters

2940

2941 The client-specific parameters of the Issued Token policy assertion along with the remainder of the server
2942 policy are consumed by the client. The STS specific parameters of the Issued Token policy assertion are
2943 passed on to the STS by copying the parameters directly into the `wst:SecondaryParameters` of the
2944 RST request sent by the Client to the STS as illustrated in the figure below.

2945

STS

3   2

RST   STS Policy

1

Client   Server

Server Policy

2946

2947 Before the Client sends the RST to the STS, it will need to obtain the policy for the STS. This will help to
2948 formulate the RST request and will include any security-specific requirements of the STS.

2949

2950 The Client may augment or replace the contents of the RST made to the STS based on the Client-specific
2951 parameters received from the Issued Token policy assertion contained in the Server policy, from policy it
2952 received for the STS, or any other local parameters.

2953

2954   The Issued Token Policy Assertion contains elements which must be understood by the Client. The
2955   assertion contains one element which contains a list of arbitrary elements which should be sent along to
2956   the STS by copying the elements as-is directly into the `wst:SecondaryParameters` of the RST
2957   request sent by the Client to the STS following the protocol defined in WS-Trust.

2958

2959   Elements inside the `sp:RequestSecurityTokenTemplate` element MUST conform to WS-Trust [WS-
2960   Trust]. All items are optional, since the Server and STS may already have a pre-arranged relationship
2961   which specifies some or all of the conditions and constraints for issued tokens.

C. Strict Security Header Layout Examples

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

C.1 Transport Binding

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

C.1.1 Policy

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

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

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

3014 C.1.2 Initiator to Recipient Messages

3015 Messages sent from initiator to recipient have the following layout for the security header:

3016   1. A `wsu:Timestamp` element.

3017   2. Any tokens contained in the [Signed Supporting Tokens] property.

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

3025   4. Any signatures for tokens contained in the [Endorsing Supporting Tokens] property. Each
3026   signature MUST cover the `wsu:Timestamp` element from 1 above and SHOULD cover at least
3027   some other unique identifier for the message in order to prevent replays. If [Token Protection] is
3028   'true', the signature MUST also cover the supporting token. If [Derived Keys] is 'true' and the
3029   supporting token is associated with a symmetric key, then a Derived Key Token, based on the
3030   supporting token, appears before the signature.

3031 The following diagram illustrates the security header layout for the initiator to recipient message:

3032

3033   The outer box shows that the entire message is protected (signed and encrypted) by the transport. The
3034   arrows on the left from the box labeled Sig$_2$ indicate the parts signed by the supporting token labeled ST$_2$,
3035   namely the message timestamp labeled TS and the token used as the basis for the signature labeled ST$_2$.
3036   The dotted arrow indicates the token that was used as the basis for the signature. In general, the ordering
3037   of the items in the security header follows the most optimal layout for a receiver to process its contents.

3038   *Example:*

3039   Initiator to recipient message

```
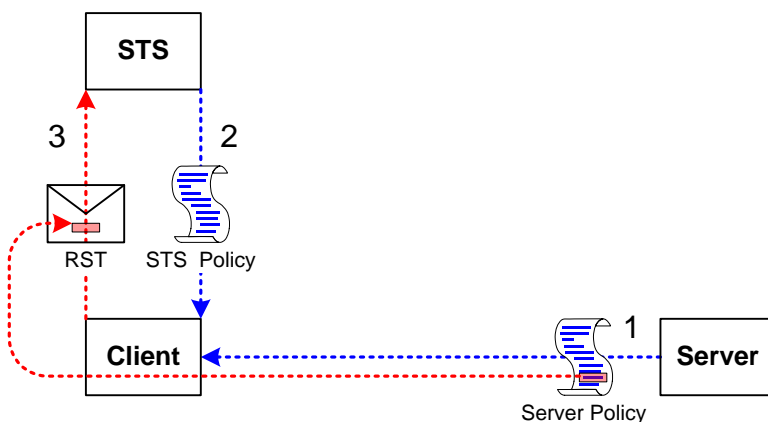3040   <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
3041     <S:Header>
3042       ...
3043       <wsse:Security>
3044         <wsu:Timestamp wsu:Id="timestamp">
3045           <wsu:Created>[datetime]</wsu:Created>
3046           <wsu:Expires>[datetime]</wsu:Expires>
3047         </wsu:Timestamp>
3048         <wsse:UsernameToken wsu:Id='SomeSignedToken' >
3049         ...
3050         </wsse:UsernameToken>
3051         <wsse:BinarySecurityToken wsu:Id="SomeSignedEndorsingToken" >
3052         ...
3053         </wsse:BinarySecurityToken>
3054         <ds:Signature>
3055           <ds:SignedInfo>
3056             <ds:References>
3057               <ds:Reference URI="#timestamp" />
3058               <ds:Reference URI="#SomeSignedEndorsingToken" />
3059             </ds:References>
3060           </ds:SignedInfo>
3061           <ds:SignatureValue>...</ds:SignatureValue>
3062           <ds:KeyInfo>
3063             <wsse:SecurityTokenReference>
3064               <wsse:Reference URI="#SomeSignedEndorsingToken" />
3065             </wsse:SecurityTokenReference>
3066           </ds:KeyInfo>
3067         </ds:Signature>
3068         ...
3069       </wsse:Security>
3070       ...
3071     </S:Header>
3072     <S:Body>
3073       ...
3074     </S:Body>
3075   </S:Envelope>
```

3076 C.1.3 Recipient to Initiator Messages

3077   Messages sent from recipient to initiator have the following layout for the security header:

3078   1. A `wsu:Timestamp` element.

3079   2. If the [Signature Confirmation] property has a value of 'true', then a
3080   `wsse11:SignatureConfirmation` element for each signature in the corresponding message
3081   sent from initiator to recipient. If there are no signatures in the corresponding message from the
3082   initiator to the recipient, then a `wsse11:SignatureConfirmation` element with no Value
3083   attribute.

3084   The following diagram illustrates the security header layout for the recipient to initiator message:

3085

3086 The outer box shows that the entire message is protected (signed and encrypted) by the transport. One
3087 `wsse11:SignatureConfirmation` element labeled SC$_1$ corresponding to the signature in the initial
3088 message illustrated previously is included. In general, the ordering of the items in the security header
3089 follows the most optimal layout for a receiver to process its contents.

3090 *Example:*

3091 Recipient to initiator message

```
3092  <S:Envelope xmlns:S="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
3093    <S:Header>
3094      ...
3095      <wsse:Security>
3096        <wsu:Timestamp wsu:Id="timestamp">
3097          <wsu:Created>[datetime]</wsu:Created>
3098          <wsu:Expires>[datetime]</wsu:Expires>
3099        </wsu:Timestamp>
3100        <wsse11:SignatureConfirmation Value="..." />
3101        ...
3102      </wsse:Security>
3103      ...
3104    </S:Header>
3105    <S:Body>
3106      ...
3107    </S:Body>
3108  </S:Envelope>
```

3109 C.2 Symmetric Binding

3110 This section describes how the 'Strict' security header layout rules apply to the Symmetric Binding.

3111 C.2.1 Policy

3112 The following example shows a policy indicating a Symmetric Binding, a symmetric key based
3113 IssuedToken provided as the Protection Token, an algorithm suite, a requirement to encrypt the message
3114 parts before signing, a requirement to encrypt the message signature, a requirement to include tokens in
3115 the message signature and the supporting signatures, a username token attached to the message, and
3116 finally an X509 token attached to the message and endorsing the message signature. Minimum message
3117 protection requirements are described as well.

```
3118    <!-- Example Endpoint Policy -->
3119    <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3120      <sp:SymmetricBinding>
3121        <wsp:Policy>
3122          <sp:ProtectionToken>
3123            <sp:IssuedToken sp:IncludeToken=".../IncludeToken/Once" >
3124              <sp:Issuer>...</sp:Issuer>
3125              <sp:RequestSecurityTokenTemplate>
3126              ...
3127              </sp:RequestSecurityTokenTemplate>
3128            </sp:IssuedToken>
3129          </sp:ProtectionToken>
3130          <sp:AlgorithmSuite>
3131            <wsp:Policy>
3132              <sp:Basic256 />
3133            </wsp:Policy>
3134          </sp:AlgorithmSuite>
3135          <sp:Layout>
3136            <wsp:Policy>
3137              <sp:Strict />
3138            </wsp:Policy>
3139          </sp:Layout>
3140          <sp:IncludeTimestamp />
3141          <sp:EncryptBeforeSigning />
3142          <sp:EncryptSignature />
3143          <sp:ProtectTokens />
3144        </wsp:Policy>
3145      </sp:SymmetricBinding>
3146      <sp:SignedSupportingTokens>
3147        <wsp:Policy>
3148          <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3149        </wsp:Policy>
3150      </sp:SignedSupportingTokens>
3151      <sp:SignedEndorsingSupportingTokens>
3152        <wsp:Policy>
3153          <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3154            <wsp:Policy>
3155              <sp:WssX509v3Token10 />
3156            </wsp:Policy>
3157          </sp:X509Token>
3158        </wsp:Policy>
3159      </sp:SignedEndorsingSupportingTokens>
3160      <sp:Wss11>
3161        <wsp:Policy>
3162          <sp:RequireSignatureConfirmation />
3163        </wsp:Policy>
3164      </sp:Wss11>
3165    </wsp:Policy>
3166
```

```
3167
3168    <!-- Example Message Policy -->
3169    <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3170      <sp:SignedParts>
3171        <sp:Header Name="Header1" Namespace="..." />
3172        <sp:Header Name="Header2" Namespace="..." />
3173        <sp:Body/>
3174      </sp:SignedParts>
3175      <sp:EncryptedParts>
3176        <sp:Header Name="Header2" Namespace="..." />
3177        <sp:Body/>
3178      </sp:EncryptedParts>
3179    </wsp:Policy>
```

3180   This policy is used as the basis for the examples shown in the subsequent section describing the security
3181   header layout for this binding.

C.2.2 Initiator to Recipient Messages

3183   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.

3216

3217   The following diagram illustrates the security header layout for the initiator to recipient message:

Encrypt Then Sign   Sign Then Encrypt

Header1   Header1
Header2   Header2
Security   Security
TS   TS
ST_1 ST_1
Ref_1 Ref_1
ST_3 ST_3
ST_2 ST_2
Sig_1 Sig_1
Sig_2 Sig_2
Ref_1
Body   Body

3218

3219   The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_1.
3220   The dashed arrows on the left from the box labeled Sig_2 indicate the parts signed by the supporting token
3221   labeled ST_2, namely the message signature labeled Sig_1 and the token used as the basis for the
3222   signature labeled ST_2. The arrows on the left from boxes labeled Ref_1 indicate references to parts
3223   encrypted using a key based on the Shared Secret Token labeled ST_1. The dotted arrows inside the box
3224   labeled Security indicate the token that was used as the basis for each cryptographic operation. In
3225   general, the ordering of the items in the security header follows the most optimal layout for a receiver to
3226   process its contents.

3227   *Example:*

3228   Initiator to recipient message using EncryptBeforeSigning:

```
3229    <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3230      xmlns:wsse11="..." xmlns:wsse="..." xmlns:saml="..."
3231      xmlns:xenc="..." xmlns:ds="...">
3232      <S:Header>
3233        <x:Header1 wsu:Id="Header1" >
3234        ...
3235        </x:Header1>
3236
```

```
3237            <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3238              <!-- Plaintext Header2
3239              <x:Header2 wsu:Id="Header2" >
3240              ...
3241              </x:Header2>
3242              -->
3243              ...
3244            </wsse11:EncryptedHeader>
3245            ...
3246            <wsse:Security>
3247              <wsu:Timestamp wsu:Id="Timestamp">
3248                <wsu:Created>...</wsu:Created>
3249                <wsu:Expires>...</wsu:Expires>
3250              </wsu:Timestamp>
3251              <saml:Assertion AssertionId="_SharedSecretToken" ...>
3252              ...
3253              </saml:Assertion>
3254              <xenc:ReferenceList>
3255                <xenc:DataReference URI="#enc_Signature" />
3256                <xenc:DataReference URI="#enc_SomeUsernameToken" />
3257                ...
3258              </xenc:ReferenceList>
3259              <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3260                <!-- Plaintext UsernameToken
3261                <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3262                ...
3263                </wsse:UsernameToken>
3264                -->
3265                ...
3266                <ds:KeyInfo>
3267                  <wsse:SecurityTokenReference>
3268                    <wsse:Reference URI="#_SharedSecretToken" />
3269                  </wsse:SecurityTokenReference>
3270                </ds:KeyInfo>
3271              </xenc:EncryptedData>
3272              <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3273              ...
3274              </wsse:BinarySecurityToken>
3275              <xenc:EncryptedData ID="enc_Signature">
3276                <!-- Plaintext Signature
3277                <ds:Signature Id="Signature">
3278                  <ds:SignedInfo>
3279                    <ds:References>
3280                      <ds:Reference URI="#Timestamp" >...</ds:Reference>
3281                      <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3282                      <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3283                      <ds:Reference URI="#_SharedSecretToken" >...</ds:Reference>
3284                      <ds:Reference URI="#Header1" >...</ds:Reference>
3285                      <ds:Reference URI="#Header2" >...</ds:Reference>
3286                      <ds:Reference URI="#Body" >...</ds:Reference>
3287                    </ds:References>
3288                  </ds:SignedInfo>
3289                  <ds:SignatureValue>...</ds:SignatureValue>
3290                  <ds:KeyInfo>
3291                    <wsse:SecurityTokenReference>
3292                      <wsse:Reference URI="#_SharedSecretToken" />
3293                    </wsse:SecurityTokenReference>
3294                  </ds:KeyInfo>
3295                </ds:Signature>
3296                -->
3297                ...
3298                <ds:KeyInfo>
3299                  <wsse:SecurityTokenReference>
3300                    <wsse:Reference URI="#_SharedSecretToken" />
```

```
3301              </wsse:SecurityTokenReference>
3302            </ds:KeyInfo>
3303          </xenc:EncryptedData>
3304          <ds:Signature>
3305            <ds:SignedInfo>
3306              <ds:References>
3307                <ds:Reference URI="#Signature" >...</ds:Reference>
3308                <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3309              </ds:References>
3310            </ds:SignedInfo>
3311            <ds:SignatureValue>...</ds:SignatureValue>
3312            <ds:KeyInfo>
3313              <wsse:SecurityTokenReference>
3314                <wsse:Reference URI="#SomeSupportingToken" />
3315              </wsse:SecurityTokenReference>
3316            </ds:KeyInfo>
3317          </ds:Signature>
3318          <xenc:ReferenceList>
3319            <xenc:DataReference URI="#enc_Body" />
3320            <xenc:DataReference URI="#enc_Header2" />
3321            ...
3322          </xenc:ReferenceList>
3323        </wsse:Security>
3324      </S:Header>
3325      <S:Body wsu:Id="Body">
3326        <xenc:EncryptedData Id="enc_Body">
3327          ...
3328          <ds:KeyInfo>
3329            <wsse:SecurityTokenReference>
3330              <wsse:Reference URI="#_SharedSecretToken" />
3331            </wsse:SecurityTokenReference>
3332          </ds:KeyInfo>
3333        </xenc:EncryptedData>
3334      </S:Body>
3335    </S:Envelope>
```

C.2.3 Recipient to Initiator Messages

3337 Messages send from recipient to initiator have the following layout for the security header:

3338 1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3339 2. If the `sp:IncludeToken` attribute on the [Encryption Token] is .../IncludeToken/Always, then the
3340 [Encryption Token].

3341 3. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Encryption Token]. This
3342 Derived Key Token is used for encryption.

3343 4. A reference list including references to encrypted items. If [Signature Protection] is 'true', then the
3344 reference list MUST include a reference to the message signature from 6 below, and the
3345 `wsse11:SignatureConfirmation` elements from 5 below if any. If [Protection Order] is
3346 'SignBeforeEncrypting', then the reference list MUST include a reference to all the message parts
3347 specified in the EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key in
3348 the token from 2 above MUST be used, otherwise the key in the [Encryption Token] from 2
3349 above.

3350 5. If [Signature Confirmation] is 'true' then a `wsse11:SignatureConfirmation` element for each
3351 signature in the corresponding message sent from initiator to recipient. If there are no signatures
3352 in the corresponding message from the initiator to the recipient, then a
3353 `wsse11:SignatureConfirmation` element with no Value attribute.

3354 6. If the [Signature Token] is not the same as the [Encryption Token], and the `sp:IncludeToken`
3355 attribute on the [Signature Token] is .../IncludeToken/Always, then the [Signature Token].

3356   7. If [Derived Keys] is 'true', then a Derived Key Token, based on the [Signature Token]. This
3357   Derived Key Token is used for signature.

3358   8. A signature over the wsu:Timestamp from 1 above, any `wsse11:SignatureConfirmation`
3359   elements from 5 above, and all the message parts specified in SignedParts assertions in the
3360   policy. If [Token Protection] is 'true', the signature MUST also cover the [Signature Token]
3361   regardless of whether it is included in the message. If [Derived Keys] is 'true', the key in the token
3362   from 6 above MUST be used, otherwise the key in the [Signature Token].

3363   9. If [Protection Order] is 'EncryptBeforeSigning' then a reference list referencing all the message
3364   parts specified in EncryptedParts assertions in the policy. If [Derived Keys] is 'true', then the key
3365   in the Derived Key Token from 3 above MUST be used, otherwise the key in the [Encryption
3366   Token].

3367   The following diagram illustrates the security header layout for the recipient to initiator message:

3368

3369   The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_1.
3370   The arrows on the left from boxes labeled Ref_1 indicate references to parts encrypted using a key based
3371   on the [SharedSecret Token] (not shown in these diagrams as it is referenced as an external token). Two
3372   `wsse11:SignatureConfirmation` elements labeled SC_1 and SC_2 corresponding to the two signatures
3373   in the initial message illustrated previously is included. In general, the ordering of the items in the security
3374   header follows the most optimal layout for a receiver to process its contents. The rules used to determine
3375   this ordering are described in Appendix C.

3376   *Example:*

3377   Recipient to initiator message using EncryptBeforeSigning:

```
3378    <S:Envelope>
3379      <S:Header>
3380        <x:Header1 wsu:Id="Header1" >
3381        ...
3382        </x:Header1>
3383        <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3384          <!-- Plaintext Header2
3385          <x:Header2 wsu:Id="Header2" >
3386          ...
3387          </x:Header2>
3388          -->
3389          ...
3390        </wsse11:EncryptedHeader>
3391        ...
3392        <wsse:Security>
3393          <wsu:Timestamp wsu:Id="Timestamp">
3394            <wsu:Created>...</wsu:Created>
3395            <wsu:Expires>...</wsu:Expires>
3396          </wsu:Timestamp>
3397          <xenc:ReferenceList>
3398            <xenc:DataReference URI="#enc_Signature" />
3399            <xenc:DataReference URI="#enc_SigConf1" />
3400            <xenc:DataReference URI="#enc_SigConf2" />
3401            ...
3402          </xenc:ReferenceList>
3403          <xenc:EncryptedData ID="enc_SigConf1" >
3404            <!-- Plaintext SignatureConfirmation
3405            <wsse11:SignatureConfirmation wsu:Id="SigConf1" >
3406            ...
3407            </wsse11:SignatureConfirmation>
3408            -->
3409          ...
3410          </xenc:EncryptedData>
3411          <xenc:EncryptedData ID="enc_SigConf2" >
3412            <!-- Plaintext SignatureConfirmation
3413            <wsse11:SignatureConfirmation wsu:Id="SigConf2" >
3414            ...
3415            </wsse11:SignatureConfirmation>
3416            -->
3417          ...
3418          </xenc:EncryptedData>
```

```
3419
3420            <xenc:EncryptedData Id="enc_Signature">
3421              <!-- Plaintext Signature
3422              <ds:Signature Id="Signature">
3423                <ds:SignedInfo>
3424                  <ds:References>
3425                    <ds:Reference URI="#Timestamp" >...</ds:Reference>
3426                    <ds:Reference URI="#SigConf1" >...</ds:Reference>
3427                    <ds:Reference URI="#SigConf2" >...</ds:Reference>
3428                    <ds:Reference URI="#Header1" >...</ds:Reference>
3429                    <ds:Reference URI="#Header2" >...</ds:Reference>
3430                    <ds:Reference URI="#Body" >...</ds:Reference>
3431                  </ds:References>
3432                </ds:SignedInfo>
3433                <ds:SignatureValue>...</ds:SignatureValue>
3434                <ds:KeyInfo>
3435                  <wsse:SecurityTokenReference>
3436                    <wsse:Reference URI="#_SomeIssuedToken" />
3437                  </wsse:SecurityTokenReference>
3438                </ds:KeyInfo>
3439              </ds:Signature>
3440              -->
3441            </xenc:EncryptedData>
3442            ...
3443            <ds:KeyInfo>
3444              <wsse:SecurityTokenReference>
3445                <wsse:Reference URI="#_SomeIssuedToken" />
3446              </wsse:SecurityTokenReference>
3447            </ds:KeyInfo>
3448          <xenc:EncryptedData>
3449          <xenc:ReferenceList>
3450            <xenc:DataReference URI="#enc_Body" />
3451            <xenc:DataReference URI="#enc_Header2" />
3452            ...
3453          </xenc:ReferenceList>
3454       </xenc:EncryptedData>
3455        </wsse:Security>
3456      </S:Header>
3457      <S:Body wsu:Id="Body">
3458        <xenc:EncryptedData Id="enc_Body">
3459          ...
3460          <ds:KeyInfo>
3461            <wsse:SecurityTokenReference>
3462              <wsse:Reference URI="#_SomeIssuedToken" />
3463            </wsse:SecurityTokenReference>
3464          </ds:KeyInfo>
3465        </xenc:EncryptedData>
3466      </S:Body>
3467    </S:Envelope>
```

3468 C.3 Asymmetric Binding

3469 This section describes how the 'Strict' security header layout rules apply to the Asymmetric Binding.

3470 C.3.1 Policy

3471 The following example shows a policy indicating an Asymmetric Binding, an X509 token as the [Initiator

3472 Token], an X509 token as the [Recipient Token], an algorithm suite, a requirement to encrypt the

3473 message parts before signing, a requirement to encrypt the message signature, a requirement to include

3474 tokens in the message signature and the supporting signatures, a requirement to include

3475 wsse11:SignatureConfirmation elements, a username token attached to the message, and finally

3476 an X509 token attached to the message and endorsing the message signature. Minimum message
3477 protection requirements are described as well.

```
3478    <!-- Example Endpoint Policy -->
3479    <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
3480      <sp:AsymmetricBinding>
3481        <wsp:Policy>
3482          <sp:RecipientToken>
3483            <wsp:Policy>
3484              <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3485            </wsp:Policy>
3486          </sp:RecipientToken>
3487          <sp:InitiatorToken>
3488            <wsp:Policy>
3489              <sp:X509Token sp:IncludeToken=".../IncludeToken/Always" />
3490            </wsp:Policy>
3491          </sp:InitiatorToken>
3492          <sp:AlgorithmSuite>
3493            <wsp:Policy>
3494              <sp:Basic256 />
3495            </wsp:Policy>
3496          </sp:AlgorithmSuite>
3497          <sp:Layout>
3498            <wsp:Policy>
3499              <sp:Strict />
3500            </wsp:Policy>
3501          </sp:Layout>
3502          <sp:IncludeTimestamp />
3503          <sp:EncryptBeforeSigning />
3504          <sp:EncryptSignature />
3505          <sp:ProtectTokens />
3506        </wsp:Policy>
3507      </sp:AsymmetricBinding>
3508      <sp:SignedEncryptedSupportingTokens>
3509        <wsp:Policy>
3510          <sp:UsernameToken sp:IncludeToken=".../IncludeToken/Once" />
3511        </wsp:Policy>
3512      </sp:SignedEncryptedSupportingTokens>
3513      <sp:SignedEndorsingSupportingTokens>
3514        <wsp:Policy>
3515          <sp:X509Token sp:IncludeToken=".../IncludeToken/Once">
3516            <wsp:Policy>
3517              <sp:WssX509v3Token10 />
3518            </wsp:Policy>
3519          </sp:X509Token>
3520        </wsp:Policy>
3521      </sp:SignedEndorsingSupportingTokens>
3522      <sp:Wss11>
3523        <wsp:Policy>
3524          <sp:RequireSignatureConfirmation />
3525        </wsp:Policy>
3526      </sp:Wss11>
3527    </wsp:Policy>
3528
```

3529

```
3530    <!-- Example Message Policy -->
3531    <wsp:All xmlns:wsp="..." xmlns:sp="...">
3532      <sp:SignedParts>
3533        <sp:Header Name="Header1" Namespace="..." />
3534        <sp:Header Name="Header2" Namespace="..." />
3535        <sp:Body/>
3536      </sp:SignedParts>
3537      <sp:EncryptedParts>
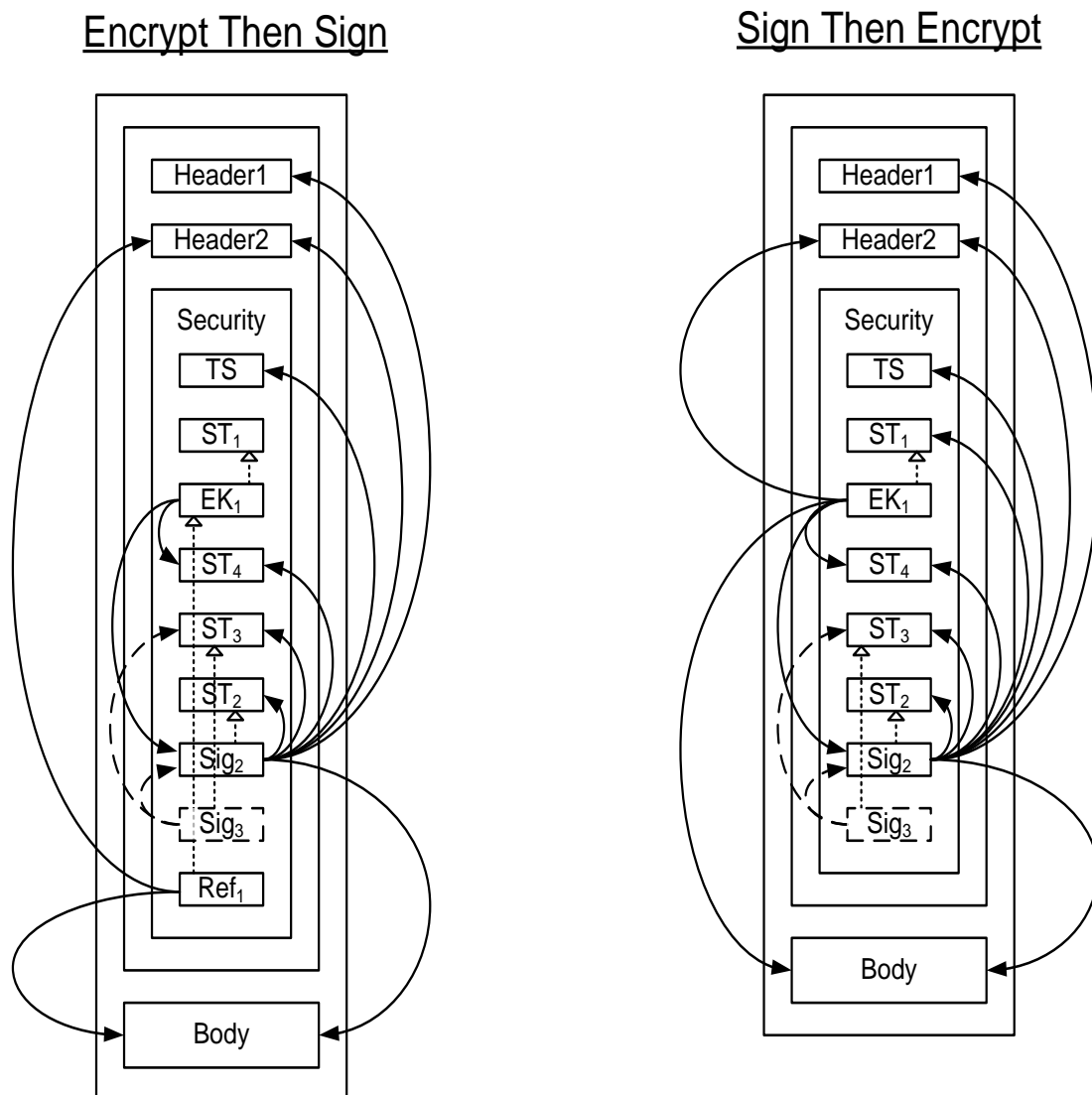3538        <sp:Header Name="Header2" Namespace="..." />
3539        <sp:Body/>
3540      </sp:EncryptedParts>
3541    </wsp:All>
```

3542

3543   This policy is used as the basis for the examples shown in the subsequent section describing the security
3544   header layout for this binding.

C.3.2 Initiator to Recipient Messages

3546   Messages sent from initiator to recipient have the following layout:

3547   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3548   2. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3549   .../IncludeToken/Once or .../IncludeToken/Always, then the [Recipient Token].

3550   3. If a [Recipient Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3551   [SignatureProtection] is 'true' then an xenc:EncryptedKey element, containing a key encrypted for
3552   the recipient. The xenc:EncryptedKey element MUST include an xenc:ReferenceList containing a
3553   reference to all the message parts specified in EncryptedParts assertions in the policy. If
3554   [Signature Protection] is 'true' then the reference list MUST contain a reference to the message
3555   signature from 6 below. It is an error if [Signature Protection] is 'true' and there is not a message
3556   signature.

3557   4. Any tokens from the supporting tokens properties (as defined in section 8) whose
3558   `sp:IncludeToken` attribute is .../IncludeToken/Once or .../IncludeToken/Always.

3559   5. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3560   .../IncludeToken/Once or .../IncludeToken/Always, then the [Initiator Token].

3561   6. A signature based on the key in the [Initiator Token] if specified, over the `wsu:Timestamp` from
3562   1 above, any tokens from 4 above regardless of whether they are included in the message, and
3563   any message parts specified in SignedParts assertions in the policy. If [Token Protection] is 'true',
3564   the signature MUST also cover the [Initiator Token] regardless of whether it is included in the
3565   message.

3566   7. Signatures for tokens from the [Endorsing Supporting Tokens] and [Signed Endorsing Supporting
3567   Tokens] properties. If [Derived Keys] is 'true' and the supporting token is associated with a
3568   symmetric key, then a Derived Key Token, based on the supporting token, appears before the
3569   signature. If [Token Protection] is 'true', the signature MUST also cover the supporting token
3570   regardless of whether it is included in the message.

3571   8. If a [Recipient Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
3572   [Signature Protection] is 'false' then an xenc:EncryptedKey element, containing a key encrypted
3573   for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
3574   reference list includes a reference to all the message parts specified in EncryptedParts assertions
3575   in the policy. The encrypted parts MUST reference the key contained in the xenc:EncryptedKey
3576   element from 3 above.

3577

3578 The following diagram illustrates the security header layout for the initiator to recipient messages:

Encrypt Then Sign   Sign Then Encrypt

3579

3580 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_2
3581 using the [Initiator Token] labeled ST_2. The dashed arrows on the left from the box labeled Sig_3 indicate
3582 the parts signed by the supporting token ST_3, namely the message signature Sig_2 and the token used as
3583 the basis for the signature labeled ST_3. The arrows on the left from boxes labeled EK_1 indicate references
3584 to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on the left
3585 from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained in the
3586 encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token used as
3587 the basis for each cryptographic operation. In general, the ordering of the items in the security header
3588 follows the most optimal layout for a receiver to process its contents. The rules used to determine this
3589 ordering are described in Appendix C.

3590

3591 Note: In most typical scenarios, the recipient key is not included in the message, but rather the encrypted
3592 key contains an external reference to the token containing the encryption key. The diagram illustrates
3593 how one might attach a security token related to the encrypted key for completeness. One possible use-

3594   case for this approach might be a stack which does not support the STR Dereferencing Transform, but
3595   wishes to include the encryption token in the message signature.

3596   Initiator to recipient message *Example*

3597
```
<S:Envelope  xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
```

```
3598           xmlns:wsse11="..." xmlns:wsse="..." xmlns:xenc="..." xmlns:ds="...">
3599         <S:Header>
3600           <x:Header1 wsu:Id="Header1" >
3601           ...
3602           </x:Header1>
3603           <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3604             <!-- Plaintext Header2
3605             <x:Header2 wsu:Id="Header2" >
3606             ...
3607             </x:Header2>
3608             -->
3609             ...
3610           </wsse11:EncryptedHeader>
3611           ...
3612           <wsse:Security>
3613             <wsu:Timestamp wsu:Id="Timestamp">
3614               <wsu:Created>...</wsu:Created>
3615               <wsu:Expires>...</wsu:Expires>
3616             </wsu:Timestamp>
3617             <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3618             ...
3619             </wsse:BinarySecurityToken>
3620             <xenc:EncryptedKey wsu:Id="RecipientEncryptedKey" >
3621               ...
3622               <xenc:ReferenceList>
3623                 <xenc:DataReference URI="#enc_Signature" />
3624                 <xenc:DataReference URI="#enc_SomeUsernameToken" />
3625                 ...
3626               </xenc:ReferenceList>
3627             </xenc:EncryptedKey>
3628             <xenc:EncryptedData ID="enc_SomeUsernameToken" >
3629               <!-- Plaintext UsernameToken
3630               <wsse:UsernameToken wsu:Id="SomeUsernameToken" >
3631               ...
3632               </wsse:UsernameToken>
3633               -->
3634               ...
3635             </xenc:EncryptedData>
3636             <wsse:BinarySecurityToken wsu:Id="SomeSupportingToken" >
3637             ...
3638             </wsse:BinarySecurityToken>
3639             <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3640             ...
3641             </wsse:BinarySecurityToken>
3642             <xenc:EncryptedData ID="enc_Signature">
3643               <!-- Plaintext Signature
3644               <ds:Signature Id="Signature">
3645                 <ds:SignedInfo>
3646                   <ds:References>
3647                     <ds:Reference URI="#Timestamp" >...</ds:Reference>
3648                     <ds:Reference URI="#SomeUsernameToken" >...</ds:Reference>
3649                     <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3650                     <ds:Reference URI="#InitiatorToken" >...</ds:Reference>
3651                     <ds:Reference URI="#Header1" >...</ds:Reference>
3652                     <ds:Reference URI="#Header2" >...</ds:Reference>
3653                     <ds:Reference URI="#Body" >...</ds:Reference>
3654                   </ds:References>
3655                 </ds:SignedInfo>
3656                 <ds:SignatureValue>...</ds:SignatureValue>
3657                 <ds:KeyInfo>
3658                   <wsse:SecurityTokenReference>
3659                     <wsse:Reference URI="#InitiatorToken" />
3660                   </wsse:SecurityTokenReference>
3661                 </ds:KeyInfo>
```

```
3662              </ds:Signature>
3663              -->
3664              ...
3665           </xenc:EncryptedData>
3666           <ds:Signature>
3667             <ds:SignedInfo>
3668               <ds:References>
3669                 <ds:Reference URI="#Signature" >...</ds:Reference>
3670                 <ds:Reference URI="#SomeSupportingToken" >...</ds:Reference>
3671               </ds:References>
3672             </ds:SignedInfo>
3673             <ds:SignatureValue>...</ds:SignatureValue>
3674             <ds:KeyInfo>
3675               <wsse:SecurityTokenReference>
3676                 <wsse:Reference URI="#SomeSupportingToken" />
3677               </wsse:SecurityTokenReference>
3678             </ds:KeyInfo>
3679           </ds:Signature>
3680           <xenc:ReferenceList>
3681             <xenc:DataReference URI="#enc_Body" />
3682             <xenc:DataReference URI="#enc_Header2" />
3683             ...
3684           </xenc:ReferenceList>
3685         </wsse:Security>
3686       </S:Header>
3687       <S:Body wsu:Id="Body">
3688         <xenc:EncryptedData Id="enc_Body">
3689           ...
3690           <ds:KeyInfo>
3691             <wsse:SecurityTokenReference>
3692               <wsse:Reference URI="#RecipientEncryptedKey" />
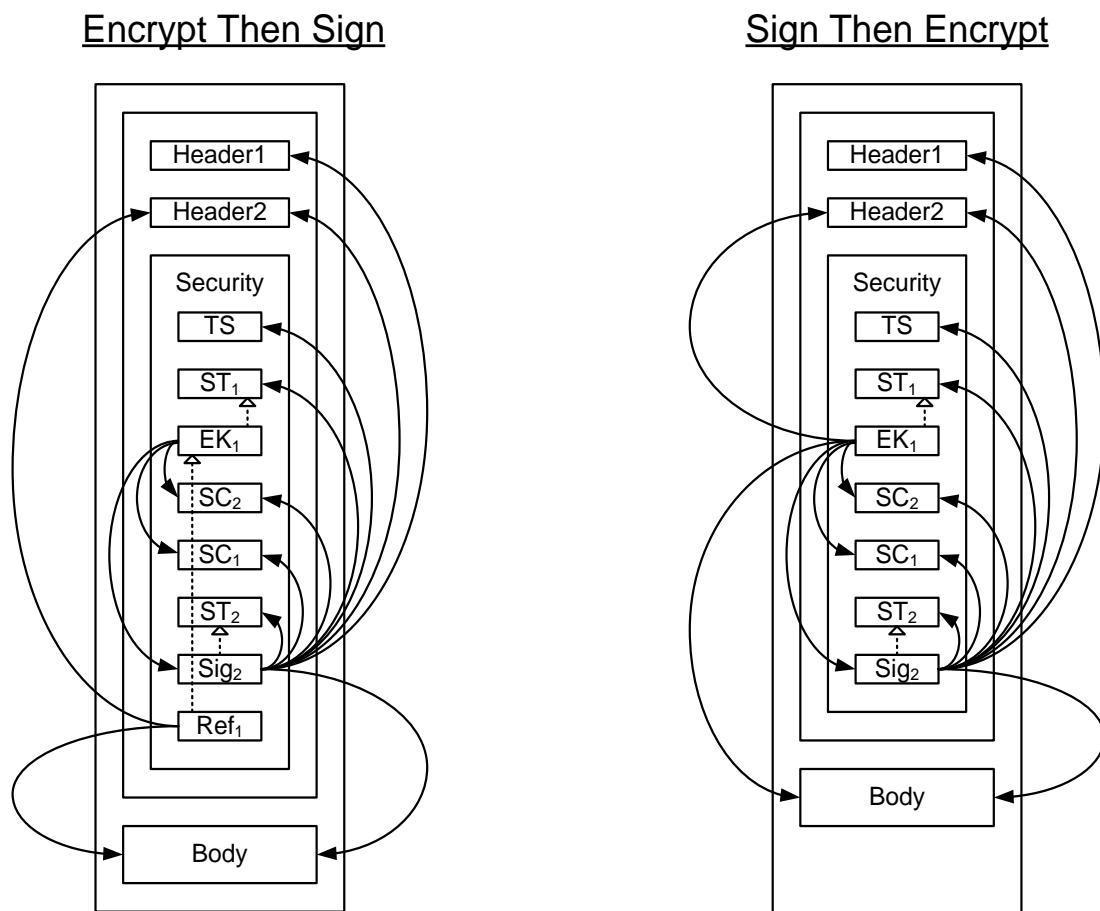3693             </wsse:SecurityTokenReference>
3694           </ds:KeyInfo>
3695         </xenc:EncryptedData>
3696       </S:Body>
3697     </S:Envelope>
```

3698 C.3.3 Recipient to Initiator Messages

3699 Messages sent from recipient to initiator have the following layout:

3700   1. A `wsu:Timestamp` element if [Timestamp] is 'true'.

3701   2. If an [Initiator Token] is specified, and the associated `sp:IncludeToken` attribute is
3702   .../IncludeToken/Always, then the [Initiator Token].

3703   3. If an [Initiator Token] is specified and [Protection Order] is 'SignBeforeEncrypting' or
3704   [SignatureProtection] is 'true' then an xenc:EncryptedKey element, containing a key encrypted for
3705   the initiator. The xenc:EncryptedKey element MUST include an xenc:ReferenceList containing a
3706   reference to all the message parts specified in EncryptedParts assertions in the policy. If
3707   [Signature Protection] is 'true' then the reference list MUST also contain a reference to the
3708   message signature from 6 below, if any and references to the
3709   `wsse11:SignatureConfirmation` elements from 4 below, if any.

3710   4. If [Signature Confirmation] is 'true', then a `wsse11:SignatureConfirmation` element for each
3711   signature in the corresponding message sent from initiator to recipient. If there are no signatures
3712   in the corresponding message from the initiator to the recipient, then a
3713   `wsse11:SignatureConfirmation` element with no Value attribute.

3714   5. If a [Recipient Token] is specified, and the associated `sp:IncludeToken` attribute is
3715   .../IncludeToken/Always, then the [Recipient Token].

3716   6. If a [Recipient Token] is specified, then a signature based on the key in the [Recipient Token],
3717   over the `wsu:Timestamp` from 1 above, the `wsse11:SignatureConfirmation` elements
3718   from 4 above, and any message parts specified in SignedParts assertions in the policy. If [Token
3719   Protection] is 'true' then the signature MUST also cover the [Recipient Token].

3720   7. If an [Initiator Token] is specified and [Protection Order] is 'EncryptBeforeSigning' then if
3721   [Signature Protection] is 'false' then an xenc:EncryptedKey element, containing a key encrypted
3722   for the recipient and a reference list, else if [Signature Protection] is 'true', a reference list. The
3723   reference list includes a reference to all the message parts specified in EncryptedParts assertions
3724   in the policy. The encrypted parts MUST reference the key contained in the xenc:EncryptedKey
3725   element from 3 above.

3726

3727 The following diagram illustrates the security header layout for the recipient to initiator messages:

Encrypt Then Sign   Sign Then Encrypt

3728

3729 The arrows on the right indicate parts that were signed as part of the message signature labeled Sig_2
3730 using the [Recipient Token] labeled ST_2. The arrows on the left from boxes labeled EK_1 indicate
3731 references to parts encrypted using a key encrypted for the [Recipient Token] labeled ST_1. The arrows on
3732 the left from boxes labeled Ref_1 indicate additional references to parts encrypted using the key contained
3733 in the encrypted key labeled EK_1. The dotted arrows inside the box labeled Security indicate the token
3734 used as the basis for each cryptographic operation. Two `wsse11:SignatureConfirmation` elements
3735 labeled SC_1 and SC_2 corresponding to the two signatures in the initial message illustrated previously is
3736 included. In general, the ordering of the items in the security header follows the most optimal layout for a
3737 receiver to process its contents. The rules used to determine this ordering are described in Appendix C.

3738 Recipient to initiator message *Example:*

```
3739    <S:Envelope xmlns:S="..." xmlns:x="..." xmlns:wsu="..."
3740      xmlns:wsse11="..." xmlns:wsse="..."
3741      xmlns:xenc="..." xmlns:ds="...">
3742      <S:Header>
3743        <x:Header1 wsu:Id="Header1" >
3744        ...
3745        </x:Header1>
3746        <wsse11:EncryptedHeader wsu:Id="enc_Header2">
3747          <!-- Plaintext Header2
3748          <x:Header2 wsu:Id="Header2" >
3749          ...
3750          </x:Header2>
3751          -->
3752          ...
3753        </wsse11:EncryptedHeader>
3754        ...
3755        <wsse:Security>
3756          <wsu:Timestamp wsu:Id="Timestamp">
3757            <wsu:Created>...</wsu:Created>
3758            <wsu:Expires>...</wsu:Expires>
3759          </wsu:Timestamp>
3760          <wsse:BinarySecurityToken wsu:Id="InitiatorToken" >
3761          ...
3762          </wsse:BinarySecurityToken>
3763          <xenc:EncryptedKey wsu:Id="InitiatorEncryptedKey" >
3764            ...
3765            <xenc:ReferenceList>
3766              <xenc:DataReference URI="#enc_Signature" />
3767              <xenc:DataReference URI="#enc_SigConf1" />
3768              <xenc:DataReference URI="#enc_SigConf2" />
3769              ...
3770            </xenc:ReferenceList>
3771          </xenc:EncryptedKey>
3772          <xenc:EncryptedData ID="enc_SigConf2" >
3773            <!-- Plaintext SignatureConfirmation
3774            <wsse11:SignatureConfirmation wsu:Id="SigConf2" ...>
3775            ...
3776            </wsse11:SignatureConfirmation>
3777            -->
3778            ...
3779          </xenc:EncryptedData>
3780          <xenc:EncryptedData ID="enc_SigConf1" >
3781            <!-- Plaintext SignatureConfirmation
3782            <wsse11:SignatureConfirmation wsu:Id="SigConf1" ...>
3783            ...
3784            </wsse11:SignatureConfirmation>
3785            -->
3786            ...
3787          </xenc:EncryptedData>
3788          <wsse:BinarySecurityToken wsu:Id="RecipientToken" >
3789          ...
3790          </wsse:BinarySecurityToken>
3791
```

```
3792              <xenc:EncryptedData ID="enc_Signature">
3793                <!-- Plaintext Signature
3794                <ds:Signature Id="Signature">
3795                  <ds:SignedInfo>
3796                    <ds:References>
3797                      <ds:Reference URI="#Timestamp" >...</ds:Reference>
3798                      <ds:Reference URI="#SigConf1" >...</ds:Reference>
3799                      <ds:Reference URI="#SigConf2" >...</ds:Reference>
3800                      <ds:Reference URI="#RecipientToken" >...</ds:Reference>
3801                      <ds:Reference URI="#Header1" >...</ds:Reference>
3802                      <ds:Reference URI="#Header2" >...</ds:Reference>
3803                      <ds:Reference URI="#Body" >...</ds:Reference>
3804                    </ds:References>
3805                  </ds:SignedInfo>
3806                  <ds:SignatureValue>...</ds:SignatureValue>
3807                  <ds:KeyInfo>
3808                    <wsse:SecurityTokenReference>
3809                      <wsse:Reference URI="#RecipientToken" />
3810                    </wsse:SecurityTokenReference>
3811                  </ds:KeyInfo>
3812                </ds:Signature>
3813                -->
3814                ...
3815              </xenc:EncryptedData>
3816              <xenc:ReferenceList>
3817                <xenc:DataReference URI="#enc_Body" />
3818                <xenc:DataReference URI="#enc_Header2" />
3819                ...
3820              </xenc:ReferenceList>
3821            </wsse:Security>
3822          </S:Header>
3823          <S:Body wsu:Id="Body">
3824            <xenc:EncryptedData Id="enc_Body">
3825              ...
3826              <ds:KeyInfo>
3827                <wsse:SecurityTokenReference>
3828                  <wsse:Reference URI="#InitiatorEncryptedKey" />
3829                </wsse:SecurityTokenReference>
3830              </ds:KeyInfo>
3831            </xenc:EncryptedData>
3832          </S:Body>
3833        </S:Envelope>
```

D. Signed and Encrypted Elements in the Security Header

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

D.1 Elements signed by the message signature

1. The `wsu:Timestamp` element (Section 6.2).
2. All `wsse11:SignatureConfirmation` elements (Section 9).
3. Security Tokens corresponding to [Initiator Signature Token],[Recipient Signature Token], [Initiator Encryption Token], [Recipient Encryption Token], [Signature Token] or [Encryption Token] when [Token Protection] has a value of 'true' (Section 6.5).
4. Security Tokens corresponding to [Signed Supporting Tokens] (see Section 8.2) or [Signed Endorsing Supporting Tokens] (Section 8.5).

D.2 Elements signed by all endorsing signatures

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

D.3 Elements signed by a specific endorsing signature

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

D.4 Elements that are encrypted

1. The `ds:Signature` element that forms the message signature when [Signature Protection] has a value of 'true' (Section 6.4).
2. All `wsse11:SignatureConfirmation` elements when [Signature Protection] has a value of 'true' (Section 6.4).
3. A `wsse:UsernameToken` may be encrypted when a transport binding is not being used (Section 5.3.1).

E. Acknowledgements

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

Original Authors of the intial contribution:

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

Original Acknowledgements of the initial contribution:

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

    Elliot Waingold, Microsoft

TC Members during the development of this specification:

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

3911	Lloyd Burch, Novell
3912	Scott Cantor, Internet2
3913	Greg Carpenter, Microsoft Corporation
3914	Steve Carter, Novell
3915	Ching-Yun (C.Y.) Chao, IBM
3916	Martin Chapman, Oracle Corporation
3917	Kate Cherry, Lockheed Martin
3918	Henry (Hyenvui) Chung, IBM
3919	Luc Clement, Systinet Corp.
3920	Paul Cotton, Microsoft Corporation
3921	Glen Daniels, Sonic Software Corp.
3922	Peter Davis, Neustar, Inc.
3923	Martijn de Boer, SAP AG
3924	Werner Dittmann, Siemens AG
3925	Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
3926	Fred Dushin, IONA Technologies
3927	Petr Dvorak, Systinet Corp.
3928	Colleen Evans, Microsoft Corporation
3929	Ruchith Fernando, WSO2
3930	Mark Fussell, Microsoft Corporation
3931	Vijay Gajjala, Microsoft Corporation
3932	Marc Goodner, Microsoft Corporation
3933	Hans Granqvist, VeriSign
3934	Martin Gudgin, Microsoft Corporation
3935	Tony Gullotta, SOA Software Inc.
3936	Jiandong Guo, Sun Microsystems
3937	Phillip Hallam-Baker, VeriSign
3938	Patrick Harding, Ping Identity Corporation
3939	Heather Hinton, IBM
3940	Frederick Hirsch, Nokia Corporation
3941	Jeff Hodges, Neustar, Inc.
3942	Will Hopkins, BEA Systems, Inc.
3943	Alex Hristov, Otecia Incorporated
3944	John Hughes, PA Consulting
3945	Diane Jordan, IBM
3946	Venugopal K, Sun Microsystems
3947	Chris Kaler, Microsoft Corporation
3948	Dana Kaufman, Forum Systems, Inc.
3949	Paul Knight, Nortel Networks Limited
3950	Ramanathan Krishnamurthy, IONA Technologies
3951	Christopher Kurt, Microsoft Corporation
3952	Kelvin Lawrence, IBM
3953	Hubert Le Van Gong, Sun Microsystems
3954	Jong Lee, BEA Systems, Inc.
3955	Rich Levinson, Oracle Corporation
3956	Tommy Lindberg, Dajeil Ltd.
3957	Mark Little, JBoss Inc.
3958	Hal Lockhart, BEA Systems, Inc.
3959	Mike Lyons, Layer 7 Technologies Inc.
3960	Eve Maler, Sun Microsystems
3961	Ashok Malhotra, Oracle Corporation
3962	Anand Mani, CrimsonLogic Pte Ltd
3963	Jonathan Marsh, Microsoft Corporation
3964	Robin Martherus, Oracle Corporation
3965	Miko Matsumura, Infravio, Inc.
3966	Gary McAfee, IBM
3967	Michael McIntosh, IBM

3968   John Merrells, Sxip Networks SRL
3969   Jeff Mischkinsky, Oracle Corporation
3970   Prateek Mishra, Oracle Corporation
3971   Bob Morgan, Internet2
3972   Vamsi Motukuru, Oracle Corporation
3973   Raajmohan Na, EDS
3974   Anthony Nadalin, IBM
3975   Andrew Nash, Reactivity, Inc.
3976   Eric Newcomer, IONA Technologies
3977   Duane Nickull, Adobe Systems
3978   Toshihiro Nishimura, Fujitsu Limited
3979   Rob Philpott, RSA Security
3980   Denis Pilipchuk, BEA Systems, Inc.
3981   Darren Platt, Ping Identity Corporation
3982   Martin Raepple, SAP AG
3983   Nick Ragouzis, Enosis Group LLC
3984   Prakash Reddy, CA
3985   Alain Regnier, Ricoh Company, Ltd.
3986   Irving Reid, Hewlett-Packard
3987   Bruce Rich, IBM
3988   Tom Rutt, Fujitsu Limited
3989   Maneesh Sahu, Actional Corporation
3990   Frank Siebenlist, Argonne National Laboratory
3991   Joe Smith, Apani Networks
3992   Davanum Srinivas, WSO2
3993   Yakov Sverdlov, CA
3994   Gene Thurston, AmberPoint
3995   Victor Valle, IBM
3996   Asir Vedamuthu, Microsoft Corporation
3997   Greg Whitehead, Hewlett-Packard
3998   Ron Williams, IBM
3999   Corinna Witt, BEA Systems, Inc.
4000   Kyle Young, Microsoft Corporation

4001

4002

4003

4004

4005

4006

4007

4008

4009

4010

4011

4012

4013

F. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
01	2-20-2007	Marc Goodner	Prepared new draft from CD01 Updated cover pages and Notices per latest OASIS template PR009 – Updated section 10, appendix A PR010 – Updated @TrustVersion PR011 – Updated section 5.3.5, 5.3.7 PR014 – Updated section 6.4 PR015 – Added new section 5.2, updates throughout section 5 token assertions for Issuer, IssuerName and Claims PR016 – Added section 5.4.11 PR017 – Updates throughout, implicit changed to implied PR018 – Updates throughout, changed wsp:Policy from optional to required PR019 – Updated sections 5.3.6, 5.3.7 PR021 – Updated sections 1, 1.2 and 1.5 (section 1.5 has placeholders for future references)
02	2-21-2007	Marc Goodner	Kept diff lines from CD01 Updated namespace, 200512 -> 200702 PR021 – Fixed placeholder references in 1.5 PR013 – Added new section 8.6, updated sections 8 and 8.1 PR020 – Updated sections 1.5, 4.1.1 and 4.2.1 PR022 – Updated section 10.1 PR023 – Updated sections 4.1.1, 4.1.2, 4.2.1, 4.2.2, and 4.2.3
03	3-06-2007	Marc Goodner	Kept diff lines from CD01 Added clarification on required wsp:Policy in 2.2 as result of finding further schema exemplar errors on this. Corrected wsp:Policy ? instances in protection tokens in section 5 Corrected incorrect close tag in 5.4.11