



WS-SecurityPolicy Examples

Committee Draft 02

3 February 2010

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-02.doc>
(Authoritative)

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-02.pdf>

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-02.html>

Previous Version:

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cs-01.doc>
(Authoritative)

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cs-01.pdf>

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cs-01.html>

Latest Version:

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.doc>

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.pdf>

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html>

Technical Committee:

OASIS Web Services Secure Exchange TC

Chair(s):

Kelvin Lawrence, IBM

Chris Kaler, Microsoft

Editor(s):

Rich Levinson, Oracle Corporation

Tony Gullotta, SOA Software Inc.

Symon Chang, Oracle Corporation.

Martin Raeppe, SAP AG

Related work:

N/A

Declared XML Namespace(s):

N/A

Abstract:

This document contains examples of how to set up WS-SecurityPolicy [WSSP] policies for a variety of common token types that are described in WS-Security 1.0 [WSS10] and WS-Security 1.1 [WSS11] token profiles [WSSTC]. Particular attention is focused on the different “security bindings” (defined in [WSSP]) within the example policies. Actual messages that have been documented in WS-Security TC [WSSTC] and other WS-Security-based Interops [WSSINTEROPS, WSSXINTEROPS, OTHERINTEROPS] that conform to some of the example policies are referenced when appropriate.

The purpose of this document is to give examples of how policies may be defined for several existing use cases that have been part of the WS-Security Interops that have been conducted (see References section for Interop documents [INTEROPS]). In addition, some example use cases have been included which show some variations from the WS-Security Interop use cases in order to demonstrate how different options and security bindings impact the structure of the policies.

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 “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/ws-sx/>.

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

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

Notices

Copyright © OASIS® 1993–2010. All Rights Reserved.

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks 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	6
1.1	Terminology and Concepts	6
1.1.1	Actors	7
1.1.2	Concepts	10
1.1.2.1	X509 Certificates	10
1.2	Terminology	11
1.3	Normative References	11
1.4	Non-Normative References	12
1.5	Specifications	13
1.6	Interops and Sample Messages	13
2	Scenarios	14
2.1	UsernameToken	14
2.1.1	UsernameToken – no security binding	14
2.1.1.1	UsernameToken with plain text password	14
2.1.1.2	UsernameToken without password	15
2.1.1.3	UsernameToken with timestamp, nonce and password hash	16
2.1.2	Use of SSL Transport Binding	17
2.1.2.1	UsernameToken as supporting token	17
2.1.3	(WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, Encrypt	19
2.1.3.1	(WSS 1.0) Encrypted UsernameToken with X.509v3	22
2.1.4	(WSS 1.1), User Name with Certificates, Sign, Encrypt	25
2.2	X.509 Token Authentication Scenario Assertions	29
2.2.1	(WSS1.0) X.509 Certificates, Sign, Encrypt	29
2.2.2	(WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt	32
2.2.2.1	(WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption	35
2.2.3	(WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt	39
2.2.4	(WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt	42
2.3	SAML Token Authentication Scenario Assertions	48
2.3.1	WSS 1.0 SAML Token Scenarios	49
2.3.1.1	(WSS1.0) SAML1.1 Assertion (Bearer)	49
2.3.1.2	(WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL	51
2.3.1.3	(WSS1.0) SAML1.1 Assertion (HK) over SSL	53
2.3.1.4	(WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional Encrypt	54
2.3.1.5	(WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt	60
2.3.2	WSS 1.1 SAML Token Scenarios	66
2.3.2.1	(WSS1.1) SAML 2.0 Bearer	66
2.3.2.2	(WSS1.1) SAML2.0 Sender Vouches over SSL	70
2.3.2.3	(WSS1.1) SAML2.0 HoK over SSL	72
2.3.2.4	(WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign, Encrypt	76
2.3.2.5	(WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt	81
2.4	Secure Conversation Scenarios	106
2.4.1	(WSS 1.0) Secure Conversation bootstrapped by Mutual Authentication with X.509 Certificates	106

3	Conformance	115
A.	Acknowledgements	116
B.	Revision History.....	119

1 Introduction

This document describes several WS-SecurityPolicy [WS-SECURITYPOLICY] examples. An example typically consists of the security aspects of a high-level Web Service use-case with several variable components. Many of the examples are based on existing use cases that have been conducted during WS-Security Interops [WSS-INTEROPS]. In those examples a reference is included to identify the specific use case in the specific interop document that is being demonstrated.

In the policy examples below, the “wsp” prefix refers to the elements defined in the WS-Policy namespace:

```
xmlns:wsp="http://www.w3.org/ns/ws-policy"
```

the “sp” prefix refers to elements defined in the WS-SecurityPolicy (WS-SP) namespace:

```
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
```

the “t” prefix refers to elements defined in the WS-Trust namespace:

```
xmlns:t="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
```

Where uses cases are based on existing scenarios, those scenarios are referenced at the beginning of the use case section. The explicit documents describing the scenarios are identified by links in [Section 3.2].

1.1 Terminology and Concepts

This section (1.1.*) describes the logical “actors” that participate in the examples. In addition, there is a discussion on general concepts that describes how the logical actors typically relate to the physical message exchanges that take place.

This section also provides a security reference model designed to provide context for the examples in terms of a conceptual framework within which the actors interact, which is intended to help readers understand the trust relationships implicit in the message exchanges shown in the examples.

In these examples there are always three important conceptual entities to recognize that exist on the initiating side of a transaction, where the transaction is being requested by sending an electronic message that contains the details of the what is being requested and by whom (the “entities” become “actors” as the discussion moves from the conceptual to the specific). These three entities are:

- The entity **requesting** the transaction (for example, if the transaction is about an application for a home mortgage loan, then the entity requesting the transaction is the prospective homeowner who will be liable to make the payments on the loan if it is approved).
- The entity **approving** the transaction to be requested. This entity is generally known as an “identity provider”, or an “authority”, and the purpose of this approving entity is to guarantee to a recipient entity that the requesting entity is making a legitimate request (continuing the above example, the authorizing entity in this case would be the organization that helps the prospective homeowner properly fill out the application, possibly a loan officer at the bank, saying that the loan application is approved for further processing).
- The entity **initiating** the actual electronic transaction message (in the above example, this entity is simply the technical software used to securely transmit the mortgage application request to a **recipient** entity that will handle the processing of the mortgage application).

WS-SecurityPolicy is primarily concerned with the technical software used between the initiating entity and the recipient entity, whom are respectively officially referred to as the Initiator and the Recipient in the WS-SecurityPolicy 1.2 specification (WS-SP) (see section 1.4 of that document).

Therefore, the purpose of this section is to give a larger real world context to understanding the examples and how to relate the technical details of WS-SecurityPolicy to the actual logical actors involved in the transactions governed by the technology.

47 The reason for providing this context is to help interested readers understand that while the technology
48 may be securing the integrity and confidentiality of the messages, there are additional questions about
49 transactions such as who is liable for any commitments resulting from the transaction and how those
50 commitments are technically bound within the message exchanges.

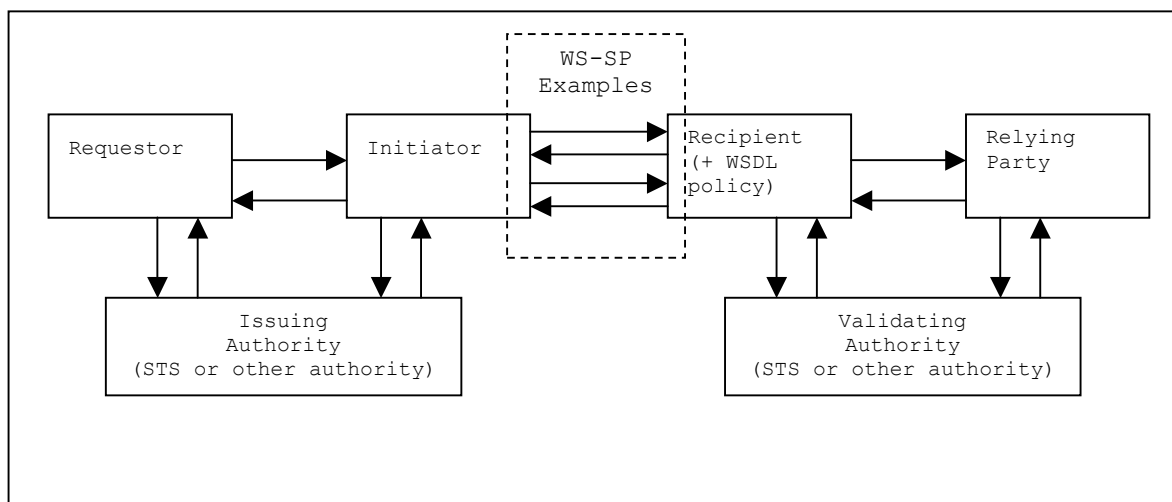
51 The purpose here is not to provide unequivocal answers to all questions regarding liability of
52 transactions, but to give a sense of how the structuring of a request message ties the participating entities
53 to the transaction. Depending on how the WS-SecurityPolicy technology is used, these “ties” can be
54 relatively stronger or weaker. Depending on the nature of the transactions supported by a particular Web
55 Application, the system managers of the Web Services Application being protected using WS-
56 SecurityPolicy techniques, may be interested in a conceptual framework for understanding which WS-SP
57 techniques are more or less appropriate for their needs.

58 These introductory sections are intended to provide this type of conceptual framework for understanding
59 how the examples in this document may be interpreted in the above context of the three entities on the
60 initiating side of the transaction. A complementary model is also provided for the recipient side of the
61 transaction, but since the recipient is generally concerned with validating the request, which is primarily a
62 back office function, less emphasis is focused on the options in that environment except as they might
63 relate back to the initiator side of the transaction.

64 1.1.1 Actors

65 The following diagram shows the actors and the relationships that may be typically involved in a network
66 security scenario:

67



68

69

Figure 1.

70

71 The diagram is intended to show the possible interactions that may occur between actors in any given
72 scenario, although, in general, depending on the policy specified by the recipient, only a subset of the
73 possible interactions will actually occur in a given scenario. Note that the Issuing and Validating
74 Authorities, may, in general be either a WS-Trust Security Token Service (STS) or other authority.

75 First, a brief narrative will describe the diagram, then the actors will be defined in more detail.

76 In a typical example interaction, a Requestor wants to issue a Web Service request to a Web Service that
77 is hosted by the “Recipient” web site on behalf of a RelyingParty, who is actually the business entity
78 responsible for making the service available and with whom any business arrangements with the
79 Requestor are made. One may think of this as an end to end interaction between a Requestor and a

80 RelyingParty, with technical resources being provided by the Initiator on the Requestor side and the
81 Recipient on the RelyingParty side.

82 Technically, what occurs is that the Requestor hands the request to the Initiator, which in turn issues a
83 query to the Recipient and is returned the WSDL [WSDL] describing the Web Service, which generally
84 includes WS-SP policy Assertions [WS-POLICY, WS-POLICYATTACHMENT] that describe the security
85 policies supported by the Recipient for this Web Service (Note: it is not necessary that the information in
86 the Assertions be obtained this way. It may also be stored locally based on out of band agreements
87 between the Requestor and RelyingParty). This interaction is shown by the upper pair of arrows between
88 the Initiator and the Recipient.

89 Upon receipt of the WS-SP policy assertions, the Initiator then interacts with the Requestor and the
90 Issuing Authority, as needed in order to meet the requirements specified by the WS-SP. Generally, what
91 is required here is that credentials and tokens are obtained from the Requestor and Issuing Authority and
92 assembled as required in a new WS-Security request message that is to be issued to the Recipient.

93 (For example, if a UsernameToken is required by the Recipient, one possibility is that the Initiator will
94 query the Requestor for Username and Password and create the appropriate token to include in the
95 Request.

96 Other possibilities exist as well, but WS-SP only governs what the final message must contain. How it
97 gets constructed and how the pieces are assembled are specific to the system environment that is
98 being used.

99 In general, the examples in this document will explain the interactions that might occur to meet the
100 policy requirements, but the actual sequences and specifics will be determined by the setup of the
101 systems involved.)

102 Finally, after assembling the necessary tokens, the Initiator (or Requestor/Initiator) may sign and encrypt
103 the message as required by the WS-SP policy and send it to the Recipient.

104 Similar to the Requestor side, on the Recipient side, the details of how the Recipient processes the
105 message and uses a Validating Authority to validate the tokens and what basis the RelyingParty uses to
106 accept or reject the request is system specific. However, in a general sense, the 3 actors identified on the
107 Recipient side will be involved to accept and process a request.

108 (For example, the Recipient may decrypt an encrypted UsernameToken containing a clear text
109 password and username and pass it to the Validating Authority for validation and authentication,
110 then the Recipient may pass the Request to the RelyingParty, which may in turn issue a request
111 for authorization to the Validating Authority.

112 These details are beyond the scope of WS-SP and the examples in this document, however, knowing that
113 this is the context in which WS-SP operates can be useful to understanding the motivation and
114 usefulness of different examples.)

115 The following list is a reference to identify the logical actors in Figure 1. (In general, these actors may
116 physically be implemented such that more than one actor is included in the physical entity, such as a
117 Security Token Service (STS) [WS-TRUST] that implements both an IssuingAuthority and a
118 ValidatingAuthority. Similarly, in a scenario where a user is at a security-enabled work station, the work
119 station may combine a Requestor and Initiator in a single physical entity.)

120

- 121 • **Requestor:** the person or entity requesting the service and who will be supplying credentials
122 issued by the IssuingAuthority and will be validated by a ValidatingAuthority. The Requestor is the
123 logical entity that supplies the credentials that ultimately get passed to the Recipient that will be
124 trusted by the RelyingParty if they are validated by the ValidatingAuthority. (Note: the logistics of
125 supplying the trusted credential is distinct from the logistics of packaging up the credentials within
126 a WS-Security header in a SOAP message and transmitting that message to the Recipient. The
127 latter logistics are covered by the Initiator, described below. The Requestor and Initiator may be
128 combined into a single physical entity and this is a common occurrence, however they do not

129 have to be and it is also a common occurrence that they are separate physical entities. The latter
130 case is typified by the Requestor being a user at a browser that may be prompted for credentials
131 by the Initiator on a server using HTTP to challenge the Requestor for a username and
132 password.)

- 133 • **IssuingAuthority:** a Security Token Service (STS), which is an organization or entity that
134 typically issues authentication credentials for Requestors. Examples include X509 Certificate
135 Authority, SAML Token Authority, Kerberos Token Authority. (For user passwords, the
136 IssuingAuthority may be thought of as the entity the user contacts for password services, such as
137 changing password, setting reminder phrases, etc.)
- 138 • **Initiator:** the system or entity that sends the message(s) to the Recipient requesting use of the
139 service on behalf of the Requestor. Typically, the Initiator will first contact the Recipient on behalf
140 of the Requestor and obtain the WS-SP policy and determine what tokens from what kind of
141 IssuingAuthority (X509, SAML, Kerberos, etc) that the Requestor will require to access the
142 service and possibly assist the Requestor to obtain those credentials. In addition, based on the
143 WS-SP policy, the Initiator determines how to format the WS-Security headers of the messages
144 being sent and how to use the security binding required by the policy.
- 145 • **Recipient:** the system or entity or organization that provides a web service for use and is the
146 supplier of the WS-SP policy that is contained in each example and is the recipient of messages
147 sent by Initiators. The Recipient manages all the processing of the request message. It may rely
148 on the services of a ValidatingAuthority to validate the credentials contained in the message.
149 When the Recipient has completed the WS-SP directed processing of the message it will
150 generally be delivered to the RelyingParty which continues processing of the message based on
151 the assurances that the Recipient has established by verifying the message is in compliance with
152 the WS-SP policies that are attached to the service description.
- 153 • **ValidatingAuthority:** the organization or entity that typically validates Requestor credentials for
154 Relying Parties, and, in general, maintains those credentials in an ongoing reliable manner.
155 Typically, the Recipient will request the ValidatingAuthority to validate the credentials before
156 delivering the Request to the RelyingParty for further processing.
- 157 • **RelyingParty:** the organization or entity that relies on the security tokens contained in the
158 messages sent by the Initiator as a basis for providing the service. For this document, the
159 RelyingParty may simply be considered to be the entity that continues processing the message
160 after the Recipient has completed all the processing required by the WS-SP policies attached to
161 the service description.

162
163 Of these actors, the Requestor and Initiator can generally be regarded as “client-side” or “requestor-
164 side” actors. The Recipient and RelyingParty (or combined “**RelyingParty/Recipient**”) can be regarded
165 as “server-side” actors.

166 Note 1: In addition to the above labelling of the actors, there is also the notion of
167 “**Sender**”. Generally, the “Sender” may be thought of as a Requestor/Initiator that is
168 independently collecting information from a user (who could be modeled as a separate
169 benign actor to the left of the Requestor in the figure 1.1 from whom the Requestor
170 gathers information that would be included in the message) and is submitting requests on
171 behalf of that user. Generally, the trust of the Recipient is on the Sender, and it is up to
172 the Sender to do whatever is necessary for the Sender to trust the user. Examples of this
173 configuration will be described in the SAML Sender Vouches sections later in this
174 document.

175 Note 2: The person or entity actually making the request is the "Requestor", however,
176 there are 2 common use cases: 1. the Requestor is a user at a browser and the request
177 is intercepted by a web service that can do WS-Security and this web service is the
178 "Initiator" which actually handles the message protection and passes the Requestor's
179 credentials (typically collected via http(s) challenge by the Initiator to the Requestor for
180 username/password) to the Recipient. 2. the Requestor is at a web-service client enabled
181 workstation, where the Requestor person making the request is also in charge of the web

182 service client that is initiating the request, in which case the combined entity may be
183 referred to as a "**Requestor/Initiator**".

184 **1.1.2 Concepts**

185 Physical message exchanges are between the Initiator and Recipient. For the purposes of this document
186 the Initiator and Recipient may be considered to be the physical systems that exchange the messages.
187 The Initiator and Recipient use the keys that are involved in the WS-SP security binding that protects the
188 messages.

189 As described in the previous section, the Requestor is the logical entity that gathers the credentials to be
190 used in the request and the Initiator is the logical entity that inserts the credentials into the request
191 message and does all the rest of the message construction in accordance with the WS-SP policy. The
192 Requestor may generally be thought of as being either a separate physical entity from the Initiator, or as
193 part of a combined physical entity with the Initiator. An example of the latter combined case would be a
194 user at a client workstation equipped with signing and encryption capabilities, where the combined entity
195 may be referred to as a "Requestor/Initiator".

196 Similarly, the IssuingAuthority should generally be thought of as a separate physical entity from the
197 Initiator. However, in some cases, such as SAML sender-vouches, the IssuingAuthority and the Initiator
198 may be the same entity.

199 In some other cases, such as the case where user passwords are involved, the ValidatingAuthority
200 system entity may also comprise the Recipient and the Relying Party, since passwords are typically
201 checked locally for validity.

202 The focus of WS-SP is the notion that policy assertions are attached to the Initiator and/or Recipient,
203 however, the concepts in those policies generally require understanding specifically the relation of the
204 parties involved (i.e. Requestor/Initiator, RelyingParty/Recipient). This is because the Requestor in
205 general does not know in advance what policies each Web Service provider requires and it is necessary
206 for practical purposes to have a front end Initiator resolve the policy and coordinate whatever actions are
207 required to exchange the Requestor tokens for the tokens required by the service. This token exchange
208 may be done using WS-Trust [[WS-TRUST](#)] to prepare the necessary requests and process responses
209 from an STS IssuingAuthority. Examples of these WS-Trust token exchanges may be found in [[WSSX-
210 INTEROP](#)].

211 Typically both the Requestor/Initiator and Recipient/RelyingParty will have relations with the
212 IssuingAuthority/ValidatingAuthority and often establish contact with those Authorities using WS-Trust for
213 the purpose of obtaining and validating tokens used in their Web Service interactions. The policies for
214 using the Authority may also be represented using WS-SP, but they are typically no different from the
215 policies shown in the examples in this document. The policies in this document may be used for any kind
216 of request, whether it be a request to a service provider for general content or operations or a request to
217 an Authority for authentication tokens.

218 In each example the relations between these actors and how the request message is prepared will be
219 described, because, in general, the policy requirements will imply these relationships. Generally, each of
220 the 3 client side actors, the Requestor, the Initiator, and the IssuingAuthority will contribute to the
221 preparation of the request message, which is the primary focus of this document. For validation of the
222 message, the Recipient, the RelyingParty, and the ValidatingAuthority are generally involved, but for this
223 document the focus is simply that the Recipient provides the WS-SP policy that dictates the preparation
224 of the request message.

225

226 **1.1.2.1 X509 Certificates**

227 The specifics of whom is trusted for X509 certificates depends on the specific organization's PKI (Public
228 Key Infrastructure) setup. For this document, we shall assume the Subject of the X509 certificate
229 identifies the actor, which may be the IssuingAuthority, or the Initiator, or the Requestor, depending on
230 the example. We shall also assume that the issuer of the X509 certificates is a general Certificate
231 Authority not directly involved in any authorization of the web service transactions, but is relied on for the
232 validity of the X509 certificate in a manner out of scope of the scenarios covered. In addition, this

233 document does not explicitly address the case of X509 certificates issued by the IssuingAuthority actor.
234 Such use cases are generally implicitly covered if one assumes that such relations are automatically
235 covered by the specifics of the organization PKI setups.
236 However, the IssuingAuthority may issue tokens, such as SAML holder-of-key that contain X509
237 certificates. In these cases, the basis of trust is that the X509 Certificate of the IssuingAuthority was used
238 to protect the X509 certificate of the Requestor which is contained in the signed SAML holder-of-key
239 token. I.e. any “chaining” of tokens is done by referencing those tokens within signed XML structures and
240 not by issuing actual X509 certificates

241 1.2 Terminology

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

245 1.3 Normative References

- 246 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
247 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 248 [WSS10-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.0 (WS-
249 Security 2004)”, March 2004.
250 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
251 security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 252 [WSS11-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.1”,
253 February 2006.
254 [http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-
255 SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
- 256 [WSS10-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.0”,
257 March 2004.
258 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
259 profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
- 260 [WSS11-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.1”,
261 February 2006.
262 [http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-
263 UsernameTokenProfile.pdf](http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf)
- 264 [WSS10-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token
265 Profile 1.0”, March 2004.
266 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
267 1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
- 268 [WSS11-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token
269 Profile 1.1”, February 2006.
270 [http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-
271 x509TokenProfile.pdf](http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf)
- 272 [WSS10-SAML11-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.0”,
273 December 2004.
274 <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>
- 275 [WSS11-SAML1120-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.1”,
276 February 2006.
277 [http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-218-spec-
278 os-SAMLTOKENProfile.pdf](http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-218-spec-os-SAMLTOKENProfile.pdf)
- 279 [WSS11-LIBERTY-SAML20-PROFILE]
280 [http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-
281 security-mechanisms-saml-profile-v2.0.pdf](http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-security-mechanisms-saml-profile-v2.0.pdf)

282 [WSS-TC-ALL-TOKEN-PROFILES] OASIS WS-Security TC links to all other WSS 1.0 and WSS 1.1
283 token profiles
284 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
285 and other documents (interop) in the TC repository:
286 http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss
287 [WS-SECURITYPOLICY] OASIS Standard, "WS-SecurityPolicy 1.2", July 2007.
288 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.doc>
289
290 [WS-SECURECONVERSATION] OASIS Standard, "WS-SecureConversation 1.3", March 2007.
291 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.doc>
292
293 [WS-TRUST] OASIS Standard, "WS-Trust 1.3", March 2007.
294 <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.doc>
295 [WS-POLICY] <http://www.w3.org/TR/ws-policy>
296 [WS-POLICYATTACHMENT] <http://www.w3.org/TR/ws-policy-attach>
297 [WSDL] <http://www.w3.org/TR/wsdl>
298 [SSL] <http://www.ietf.org/rfc/rfc2246.txt>
299 [SAML11-CORE] OASIS Standard, "Assertions and Protocol for the OASIS Security Assertion
300 Markup Language (SAML) V1.1", September 2003.
301 <http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>
302
303 [SAML20-CORE] OASIS Standard, "Assertions and Protocols for the OASIS Security Assertion
304 Markup Language (SAML) V2.0", March 2005.
305 <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>
306 [XML-DSIG] <http://www.w3.org/TR/xmlsig-core/>
307 [XML-ENCR] <http://www.w3.org/TR/xmlenc-core/>
308
309

310 1.4 Non-Normative References

311 WSS10-INTEROP-01: OASIS Working Draft 06, "Web Services Security: Interop 1 Scenarios",
312 Jun 2003.
313 <http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf>
314
315
316 WSS10-INTEROP-02: OASIS Working Draft 06, "Web Services Security: Interop 2 Scenarios",
317 Oct 2003.
318 <http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc>
319
320
321 WSS11-INTEROP-01: OASIS Working Draft 02, "Web Services Security 1.1: Interop Scenario",
322 June 2005. <http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc>
323
324
325 WSS10-KERBEROS-INTEROP: OASIS Working Draft 02, "Web Services Security: Kerberos Token
326 Profile Interop Scenarios", Jan 2005.
327 <http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-interop.doc>
328
329
330 WSS10-SAML11-INTEROP: OASIS Working Draft 12, "Web Services Security: SAML Interop 1
331 Scenarios", July 2004.

332 [http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-](http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc)
333 [draft-12.doc](http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc)
334
335 WSS11-SAML1120-INTEROP: OASIS Working Draft 4, “Web Services Security: SAML 2.0 Interop
336 Scenarios”, January 2005.
337 [http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)
338 [draft-v4.doc](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)
339
340 WSSX-PRE-INTEROP: OASIS Contribution Version 1.0d, “WS-SecureConversation and WS-Trust
341 Interop Scenarios”, September 29, 2004.
342 [http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConve-](http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc)
343 [rsation_Interop.2004-10.doc](http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc)
344
345 WSSX-WSTR-WSSC-INTEROP: OASIS White Paper Version ED-10, “WS-SX Interop Scenarios - Phase
346 2 Scenarios for demonstration of WS-SX TC specifications”, October 31, 2006.
347 [http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc)
348 [10.doc](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc)
349
350 WS-SECURE-INTEROP: OASIS White Paper Version ED-01, “Examples of Secure Web Service
351 Message Exchange”, July 11, 2008.
352 [http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-](http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-message-examples.doc)
353 [message-examples.doc](http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-message-examples.doc)
354
355 WSI-SCM-SAMPLEAPPL: [http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-](http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc)
356 [04/SCMSecurityArchitectureWGD5.00.doc](http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc) (login required)
357
358

359 **1.5 Specifications**

360 **1.6 Interops and Sample Messages**

361 2 Scenarios

362 2.1 UsernameToken

363 UsernameToken authentication scenarios that use simple username password token for authentication.
364 There are several sub-cases.

365 2.1.1 UsernameToken – no security binding

366 In this model a UsernameToken is placed within a WS-Security header in the SOAP Header [WSS10-
367 USERNAME, WSS11-USERNAME]. No other security measure is used.

368 Because no security binding is used, there is no explicit distinction between the Requestor, who is
369 identified in the UsernameToken and the Initiator, who physically sends the message. They may be one
370 and the same or distinct parties. The lack of a security binding indicates that any direct URL access
371 method (ex. HTTP) may be used to access the service.

372 2.1.1.1 UsernameToken with plain text password

373 This scenario is based on the first WS-Security Interop Scenarios Document [WSS10-INTEROP-01
374 Scenario 1 – section 3.4.4]

375 (<http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf>).

376 This policy says that Requestor/Initiator must send a password in a UsernameToken in a WS-Security
377 header to the Recipient (who as the Authority will validate the password). The password is required
378 because that is the default requirement for the Web Services Security Username Token Profile 1.x
379 [WSS10-USERNAME, WSS11-USERNAME].

380 This setup is only recommended where confidentiality of the password is not an issue, such as a pre-
381 production test scenario with dummy passwords, which might be used to establish that the Initiator can
382 read the policy and prepare the message correctly, and that connectivity and login to the service can be
383 performed.

384

```
385 (P001) <wsp:Policy>  
386 (P002) <sp:SupportingTokens>  
387 (P003) <wsp:Policy>  
388 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-  
389 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient"  
390 (P005) </wsp:Policy>  
391 (P006) </sp:SupportingTokens>  
392 (P007) </wsp:Policy>
```

393

394 An example of a message that conforms to the above stated policy is as follows.

395

```
396 (M001) <?xml version="1.0" encoding="utf-8" ?>  
397 (M002) <soap:Envelope xmlns:soap="...">  
398 (M003) <soap:Header>  
399 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">  
400 (M005) <wsse:UsernameToken>  
401 (M006) <wsse:Username>Chris</wsse:Username>  
402 (M007) <wsse:Password Type="http://docs.oasis-  
403 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-  
404 1.0#PasswordText">sirhC</wsse:Password>  
405 (M008) <wsse:Nonce EncodingType="...#Base64Binary"  
406 (M009) >pN...=</wsse:Nonce>
```

```

407 (M010)      <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
408 (M011)      </wsse:UsernameToken>
409 (M012)      </wsse:Security>
410 (M013)      </soap:Header>
411 (M014)      <soap:Body>
412 (M015)      <Ping xmlns="http://xmlsoap.org/Ping">
413 (M016)      <text>EchoString</text>
414 (M017)      </Ping>
415 (M018)      </soap:Body>
416 (M019)      </soap:Envelope>

```

417

418 The UsernameToken element starting on line (M005) satisfies the UsernameToken assertion on line
419 (P004). By default, a Password element is included in the UsernameToken on line (M007) holding a plain
420 text password. Lines (M008-M010) contain an optional Nonce element and Created timestamp, which,
421 while optional, are recommended to improve security of requests against replay and other attacks
422 [[WSS10-USERNAME](#)]. All WS-Security compliant implementations should support the UsernameToken
423 with cleartext password with or without the Nonce and Created elements.

424

425 2.1.1.2 UsernameToken without password

426 This policy is the same as 2.1.1.1 except no password is to be placed in the UsernameToken. There are
427 no credentials to further establish the identity of the Requestor and no security binding that the Initiator is
428 required to use. This is a possible production scenario where all the service provider wants is a
429 UsernameToken to associate with the request. There is no explicit Authority implied in this scenario,
430 except possibly that the username extracted from the UsernameToken would be evaluated by a server-
431 side "Authority" that maintained a list of valid username values.

432

```

433 (P001)      <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
434 (P002)      <sp:SupportingTokens>
435 (P003)      <wsp:Policy>
436 (P004)      <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
437 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
438 (P005)      <wsp:Policy>
439 (P006)      <sp:NoPassword/>
440 (P007)      </wsp:Policy>
441 (P008)      </sp:UsernameToken>
442 (P009)      </wsp:Policy>
443 (P010)      </sp:SupportingTokens>
444 (P011)      </wsp:Policy>

```

445 Lines (P002) – (P010) contain the SupportingToken assertion which includes a UsernameToken
446 indicating that a UsernameToken must be included in the security header.

447 Line (P006) requires that the wsse:Password element must not be present in the UsernameToken.

448 An example of an input message that conforms to the above stated policy is as follows:

449

```

450 (M001)      <?xml version="1.0" encoding="utf-8" ?>
451 (M002)      <soap:Envelope xmlns:soap="...">
452 (M003)      <soap:Header>
453 (M004)      <wsse:Security soap:mustUnderstand="1"
454 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
455 wssecurity-secext-1.0.xsd">
456 (M005)      <wsse:UsernameToken>
457 (M006)      <wsse:Username>Chris</wsse:Username>
458 (M007)      </wsse:UsernameToken>
459 (M008)      </wsse:Security>
460 (M009)      </soap:Header>
461 (M010)      <soap:Body>

```

```

462 (M011) <Ping xmlns="http://xmlsoap.org/Ping">
463 (M012) <text>EchoString</text>
464 (M013) </Ping>
465 (M014) </soap:Body>
466 (M015) </soap:Envelope>

```

467 Lines (M005) – (M007) hold the unsecured UsernameToken which only contains the name of user
468 (M006), but no password.

469

470 2.1.1.3 UsernameToken with timestamp, nonce and password hash

471 This scenario is similar to 2.1.1.1, except it is more secure, because the Requestor password is protected
472 by combining it with a nonce and timestamp, and then hashing the combination. Therefore, this may be
473 considered as a potential production scenario where passwords may be safely used. It may be assumed
474 that the password must be validated by a server-side ValidatingAuthority and so must meet whatever
475 requirements the specific Authority has established.

476

```

477 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
478 (P002) <sp:SupportingTokens>
479 (P003) <wsp:Policy>
480 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
481 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
482 (P005) <wsp:Policy>
483 (P006) <sp:HashPassword/>
484 (P007) </wsp:Policy>
485 (P008) </sp:UsernameToken>
486 (P009) </wsp:Policy>
487 (P010) </sp:SupportingTokens>
488 (P011) </wsp:Policy>

```

489

490 An example of a message that conforms to the above stated policy is as follows.

491

```

492 (M001) <?xml version="1.0" encoding="utf-8" ?>
493 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
494 (M003) <soap:Header>
495 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
496 (M005) <wsse:UsernameToken
497 wsu:Id="uuid-7cee5976-0111-e9c1-e34b-af1e85fa3866">
498 (M006) <wsse:Username>Chris</wsse:Username>
499 (M007) <wsse:Password Type="http://docs.oasis-
500 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
501 1.0#PasswordDigest"
502 >weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==</wsse:Password>
503 (M008) <wsse:Nonce>WSCqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
504 (M009) <wsu:Created>2007-05-01T01:15:30Z</wsu:Created>
505 (M010) </wsse:UsernameToken>
506 (M011) </wsse:Security>
507 (M012) </soap:Header>
508 (M013) <soap:Body wsu:Id="uuid-7cee4264-0111-e0cb-8329-af1e85fa3866">
509 (M014) <Ping xmlns="http://xmlsoap.org/Ping">
510 (M015) <text>EchoString</text>
511 (M016) </Ping>
512 (M017) </soap:Body>
513 (M018) </soap:Envelope>

```

514

515 This message is very similar to the one in section 2.1.1.1. A UsernameToken starts on line (M005) to
516 satisfy the UsernameToken assertion. However, in this example the Password element on line (M007) is

517 of type PasswordDigest to satisfy the HashPassword assertion on line (P006). The Nonce (M008) and
518 Created timestamp (M009) are also included as dictated by the HashPassword assertion. The Nonce and
519 timestamp values are included in the password digest on line (M007).

520 2.1.2 Use of SSL Transport Binding

521 Both server authentication and mutual (client AND server) authentication SSL [SSL] are supported via
522 use of the sp:TransportBinding policy assertion. (For mutual authentication, a RequireClientCertificate
523 assertion may be inserted within the HttpsToken assertion. The ClientCertificate may be regarded as a
524 credential token for authentication of the Initiator, which in the absence of any additional token
525 requirements would generally imply the Initiator is also the Requestor. The Authority would be the issuer
526 of the client certificate.)

527

```
528 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
529   <sp:TransportBinding>  
530     <wsp:Policy>  
531       <sp:TransportToken>  
532         <wsp:Policy>  
533           <sp:HttpsToken />  
534         </wsp:Policy>  
535       </sp:TransportToken>  
536       <sp:AlgorithmSuite>  
537         <wsp:Policy>  
538           <sp:Basic256 />  
539         </wsp:Policy>  
540       </sp:AlgorithmSuite>  
541       <sp:Layout>  
542         <wsp:Policy>  
543           <sp:Strict />  
544         </wsp:Policy>  
545       </sp:Layout>  
546       <sp:IncludeTimestamp />  
547     </wsp:Policy>  
548   </sp:TransportBinding>  
549 </wsp:Policy>
```

550 2.1.2.1 UsernameToken as supporting token

551 Additional tokens can be included as supporting tokens. Each of the UsernameTokens described in
552 section 2.1.1 may be used in this scenario and any clear text information or password will be protected by
553 SSL. So, for example, including a user name token over server authentication SSL we have:

554

```
555 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
556 (P002)   <sp:TransportBinding>  
557 (P003)     <wsp:Policy>  
558 (P004)       <sp:TransportToken>  
559 (P005)         <wsp:Policy>  
560 (P006)           <sp:HttpsToken/>  
561 (P007)         </wsp:Policy>  
562 (P008)       </sp:TransportToken>  
563 (P009)       <sp:AlgorithmSuite>  
564 (P010)         <wsp:Policy>  
565 (P011)           <sp:Basic256/>  
566 (P012)         </wsp:Policy>  
567 (P013)       </sp:AlgorithmSuite>  
568 (P014)       <sp:Layout>  
569 (P015)         <wsp:Policy>  
570 (P016)           <sp:Strict/>  
571 (P017)         </wsp:Policy>  
572 (P018)       </sp:Layout>  
573 (P019)       <sp:IncludeTimestamp/>
```

```

574 (P020) </wsp:Policy>
575 (P021) </sp:TransportBinding>
576 (P022) <sp:SupportingTokens>
577 (P023) <wsp:Policy>
578 (P024) <sp:UsernameToken/>
579 (P025) </wsp:Policy>
580 (P026) </sp:SupportingTokens>
581 (P027) </wsp:Policy>

```

582 Lines (P002) – (P021) contain the TransportBinding assertion which indicates that the message must be
583 protected by a secure transport protocol like SSL or TLS.

584 Lines (P004) – (P008) hold the TransportToken assertion, indicating that the transport is secured by
585 means of an HTTPS Transport Token, requiring to perform cryptographic operations based on the
586 transport token using the Basic256 algorithm suite (P011).

587 In addition, the Layout assertion in lines (P014) – (P018) require that the order of the elements in the
588 SOAP message security header must conform to rules defined by [WSSECURITYPOLICY](#) that follow the
589 general principle of 'declare before use'.

590 Line (P019) indicates that the wsu:Timestamp element must be present in the SOAP message security
591 header.

592 Lines (P022) – (P026) Lines contain the SupportingToken assertion which includes a UsernameToken
593 indicating that a UsernameToken must be included in the security header.

594 An example of an input message prior to the transport encryption that conforms to the above stated policy
595 is as follows:

```

596 (M001) <?xml version="1.0" encoding="utf-8" ?>
597 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
598 (M003) <soap:Header>
599 (M004) <wsse:Security soap:mustUnderstand="1"
600 < xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
601 wss-wssecurity-secext-1.0.xsd">
602 (M005) <wsu:Timestamp wsu:Id="uuid-8066364f-0111-f371-47bf-ba986d2d7dc4">
603 (M006) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
604 (M007) </wsu:Timestamp>
605 (M008) <wsse:UsernameToken
606 < wsu:Id="uuid-8066368d-0111-e744-f37b-ba986d2d7dc4">
607 (M009) <wsse:Username>Chris</wsse:Username>
608 (M010) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
609 200401-wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
610 (M011) </wsse:UsernameToken>
611 (M012) </wsse:Security>
612 (M013) </soap:Header>
613 (M014) <soap:Body wsu:Id="uuid-8066363f-0111-ffdc-de48-ba986d2d7dc4">
614 (M015) <Ping xmlns="http://xmlsoap.org/Ping">
615 (M016) <text>EchoString</text>
616 (M017) </Ping>
617 (M018) </soap:Body>
618 (M019) </soap:Envelope>

```

619 Lines (M005) – (M007) hold Timestamp element according to the IncludeTimestamp assertion.

620 Lines (M008) – (M011) hold the UsernameToken which contains the name (M009) and password (M010)
621 of the user sending this message.

622

623

623 2.1.3 (WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, 624 Encrypt

625

626 This scenario is based on WS-I SCM Security Architecture Technical requirements for securing the SCM
627 Sample Application, March 2006 [WSI-SCM-SAMPLEAPPL – GetCatalogRequest, SubmitOrderRequest].

628 This use case corresponds to the situation where both parties have X.509v3 certificates (and public-
629 private key pairs). The Initiator includes a user name token that may stand for the Requestor on-behalf-of
630 which the Initiator is acting. The UsernameToken is included as a SupportingToken; this is also
631 encrypted. The Authority for this request is generally the Subject of the Initiator's trusted X.509 Certificate.

632 We model this by using the asymmetric security binding [WSSP] with a UsernameToken
633 SupportingToken.

634 The message level policies in this section and subsequent sections cover a different scope of the web
635 service definition than the security binding level policy and so appear as separate policies and are
636 attached at WSDL Message Policy Subject. These are shown below as input and output policies. Thus,
637 we need a set of coordinated policies one with endpoint subject and two with message subjects to
638 achieve this use case.

639 The policy is as follows:

```
640 (P001) <wsp:Policy wsu:Id="wss10_up_cert_policy" >  
641 (P002)   <wsp:ExactlyOne>  
642 (P003)     <wsp>All>  
643 (P004)       <sp:AsymmetricBinding>  
644 (P005)         <wsp:Policy>  
645 (P006)           <sp:InitiatorToken>  
646 (P007)             <wsp:Policy>  
647 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-  
648 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
649 (P009)                 <wsp:Policy>  
650 (P010)                   <sp:WssX509V3Token10/>  
651 (P011)                 </wsp:Policy>  
652 (P012)                 </sp:X509Token>  
653 (P013)                 </wsp:Policy>  
654 (P014)                 </sp:InitiatorToken>  
655 (P015)                 <sp:RecipientToken>  
656 (P016)                   <wsp:Policy>  
657 (P017)                     <sp:X509Token sp:IncludeToken="http://docs.oasis-  
658 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
659 (P018)                       <wsp:Policy>  
660 (P019)                         <sp:WssX509V3Token10/>  
661 (P020)                       </wsp:Policy>  
662 (P021)                       </sp:X509Token>  
663 (P022)                       </wsp:Policy>  
664 (P023)                     </sp:RecipientToken>  
665 (P024)                     <sp:AlgorithmSuite>  
666 (P025)                       <wsp:Policy>  
667 (P026)                         <sp:Basic256/>  
668 (P027)                       </wsp:Policy>  
669 (P028)                     </sp:AlgorithmSuite>  
670 (P029)                     <sp:Layout>  
671 (P030)                       <wsp:Policy>  
672 (P031)                         <sp:Strict/>  
673 (P032)                       </wsp:Policy>  
674 (P033)                     </sp:Layout>  
675 (P034)                     <sp:IncludeTimestamp/>  
676 (P035)                     <sp:OnlySignEntireHeadersAndBody/>  
677 (P036)                   </wsp:Policy>  
678 (P037)                 </sp:AsymmetricBinding>  
679 (P038)                 <sp:Wss10>  
680 (P039)                   <wsp:Policy>
```

```

681      (P040)      <sp:MustSupportRefKeyIdentifier/>
682      (P041)      </wsp:Policy>
683      (P042)      </sp:Wss10>
684      (P043)      <sp:SignedEncryptedSupportingTokens>
685      (P044)      <wsp:Policy>
686      (P045)      <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
687      open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
688      (P046)      <wsp:Policy>
689      (P047)      <sp:WssUsernameToken10/>
690      (P048)      </wsp:Policy>
691      (P049)      </sp:UsernameToken>
692      (P050)      </wsp:Policy>
693      (P051)      </sp:SignedEncryptedSupportingTokens>
694      (P052)      </wsp:All>
695      (P053)      </wsp:ExactlyOne>
696      </wsp:Policy>
697
698      (P054) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
699      (P055) <wsp:ExactlyOne>
700      (P056) <wsp:All>
701      (P057) <sp:SignedParts>
702      (P058) <sp:Body/>
703      (P059) </sp:SignedParts>
704      (P060) <sp:EncryptedParts>
705      (P061) <sp:Body/>
706      (P062) </sp:EncryptedParts>
707      (P063) </wsp:All>
708      (P064) </wsp:ExactlyOne>
709      (P065) </wsp:Policy>
710
711      (P066) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
712      (P067) <wsp:ExactlyOne>
713      (P068) <wsp:All>
714      (P069) <sp:SignedParts>
715      (P070) <sp:Body/>
716      (P071) </sp:SignedParts>
717      (P072) <sp:EncryptedParts>
718      (P073) <sp:Body/>
719      (P074) </sp:EncryptedParts>
720      (P075) </wsp:All>
721      (P076) </wsp:ExactlyOne>
722      (P077) </wsp:Policy>

```

723 Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token
724 must be used for the message signature and the recipient's token must be used for message encryption.

725 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)
726 indicate that the initiator token must be an X.509 token that must be included with all messages sent to
727 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the
728 WS-Security 1.0 X.509 Token Profile.

729 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)
730 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token
731 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to
732 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the
733 token in any messages where the token is used.

734 Line (P034) requires the inclusion of a timestamp.

735 Lines (P043) – (P051) contain a SignedEncryptedSupportingTokens assertion which identifies the
736 inclusion of an additional token which must be included in the message signature and encrypted. Lines
737 (P045) – (P049) indicate that the supporting token must be a UsernameToken and must be included in all
738 messages to the recipient. Line (P047) dictates the UsernameToken must conform to the WS-Security 1.0
739 UsernameToken Profile.

740 Lines (P055) – (P066) contain a policy that is attached to the input message. Lines (P058) – (P060)
741 require that the body of the input message must be signed. Lines (P061) – (P063) require the body of the
742 input message must be encrypted.

743 Lines (P067) – (P078) contain a policy that is attached to the output message. Lines (P070) – (P072)
744 require that the body of the output message must be signed. Lines (P073) – (P075) require the body of
745 the output message must be encrypted.

746 An example of an input message that conforms to the above stated policy is as follows.

```
747 (M001) <?xml version="1.0" encoding="utf-8" ?>
748 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
749 (M003)   <soap:Header>
750 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
751 (M005)       <xenc:ReferenceList>
752 (M006)         <xenc:DataReference URI="#encUT"/>
753 (M007)         <xenc:DataReference URI="#encBody"/>
754 (M008)       </xenc:ReferenceList>
755 (M009)       <wsu:Timestamp wsu:Id="T0">
756 (M010)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
757 (M011)       </wsu:Timestamp>
758 (M012)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="...#X509v3"
759 EncodingType="...#Base64Binary">
760 (M013)         MIEEZzCCA9CgAwIBAgIQEmtJZc0...
761 (M014)       </wsse:BinarySecurityToken>
762 (M015)       <xenc:EncryptedData wsu:Id="encUT">
763 (M016)         <ds:KeyInfo>
764 (M017)           <wsse:SecurityTokenReference>
765 (M018)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
766 ValueType="...#X509SubjectKeyIdentifier">
767 (M019)               MIGfMa0GCSq...
768 (M020)             </wsse:KeyIdentifier>
769 (M021)           </wsse:SecurityTokenReference>
770 (M022)         </ds:KeyInfo>
771 (M023)         <xenc:CipherData>
772 (M024)           <xenc:CipherValue>...</xenc:CipherValue>
773 (M025)         </xenc:CipherData>
774 (M026)       </xenc:EncryptedData>
775 (M027)       <ds:Signature>
776 (M028)         <ds:SignedInfo>...
777 (M029)           <ds:Reference URI="#T0">...</ds:Reference>
778 (M030)           <ds:Reference URI="#usernameToken">...</ds:Reference>
779 (M031)           <ds:Reference URI="#body">...</ds:Reference>
780 (M032)         </ds:SignedInfo>
781 (M033)       <ds:SignatureValue>HFLP...</ds:SignatureValue>
782 (M034)       <ds:KeyInfo>
783 (M035)         <wsse:SecurityTokenReference>
784 (M036)           <wsse:Reference URI="#binaryToken"/>
785 (M037)         </wsse:SecurityTokenReference>
786 (M038)       </ds:KeyInfo>
787 (M039)       </ds:Signature>
788 (M040)     </wsse:Security>
789 (M041)   </soap:Header>
790 (M042)   <soap:Body wsu:Id="body">
791 (M043)     <xenc:EncryptedData wsu:Id="encBody">
792 (M044)       <ds:KeyInfo>
793 (M045)         <wsse:SecurityTokenReference>
794 (M046)           <wsse:KeyIdentifier EncodingType="...#Base64Binary"
795 ValueType="...#X509SubjectKeyIdentifier">
796 (M047)             MIGfMa0GCSq...
797 (M048)           </wsse:KeyIdentifier>
798 (M049)         </wsse:SecurityTokenReference>
799 (M050)       </ds:KeyInfo>
800 (M051)     <xenc:CipherData>
801 (M052)       <xenc:CipherValue>...</xenc:CipherValue>
```

```
802 (M053) </xenc:CipherData>
803 (M054) </xenc:EncryptedData>
804 (M055) </soap:Body>
805 (M056) </soap:Envelope>
```

806 Line (M006) is an encryption data reference that references the encrypted UsernameToken on lines
807 (M015) – (M024) which was required to be included by the SignedEncryptedSupportingTokens assertion.
808 Lines (M018) – (M020) hold a KeyIdentifier of the recipient's token used to encrypt the UsernameToken
809 as required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the
810 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included
811 token.

812 Line (M007) is an encryption data reference that references the encrypted body of the message on lines
813 (M043) – (M054). The encryption was required by the EncryptedParts assertion of the input message
814 policy. It also uses the recipient token as identified by the KeyIdentifier.

815 Lines (M009) – (M011) contain a timestamp for the message as required by the IncludeTimestamp
816 assertion.

817 Lines (M012) – (M014) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as
818 required by the InitiatorToken assertion.

819 Lines (M027) – (M039) contain the message signature.

820 Line (M029) indicates the message timestamp is included in the signature as required by the
821 IncludeTimestamp assertion definition.

822 Line (M030) indicates the supporting UsernameToken is included in the signature as required by the
823 SignedEncryptedSupportingTokens assertion. Because the token was encrypted its content prior to
824 encryption is included below to better illustrate the reference.

```
825 (M057) <wsse:UsernameToken wsu:Id="usernameToken">
826 (M058) <wsse:Username>Chris</wsse:Username>
827 (M059) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
828 wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
829 (M060) </wsse:UsernameToken>
```

830 Line (M031) indicates the message body is included in the signature as required by the SignedParts
831 assertion of the input message policy.

832 Note that the initiator's BinarySecurityToken is not included in the message signature as it was not
833 required by policy.

834 Line (M036) references the initiator's BinarySecurityToken included in the message for identifying the key
835 used for signing as dictated by the AsymmetricBinding assertion.

836

837 **2.1.3.1 (WSS 1.0) Encrypted UsernameToken with X.509v3**

838 This scenario is based on the first WS-Security Interop Scenarios Document [WSS10-INTEROP-01
839 Scenario 2 – section 4.4.4]

840 (<http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf>).

841 This policy says that Requestor/Initiator must send a password in an encrypted UsernameToken in a WS-
842 Security header to the Recipient (who as the Authority will validate the password). The password is
843 required because that is the default requirement for the Web Services Security Username Token Profile
844 1.x [WSS10-USERNAME, WSS11-USERNAME].

845 This setup is only recommended when the sender cannot provide the "message signature" and it is
846 RECOMMENDED that the receiver employs some security mechanisms external to the message to
847 prevent the spoofing attacks.

848 The policy is as follows:

```
849 (P001) <wsp:Policy wsu:Id="wss10_encrypted_unt_policy" >
850 (P002) <sp:AsymmetricBinding>
851 (P003) <wsp:Policy>
852 (P004) <sp:InitiatorEncryptionToken>
```

```

853 (P005) <wsp:Policy>
854 (P006) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
855 sx/ws-securitypolicy/200702/IncludeToken/Never">
856 (P007) <wsp:Policy>
857 (P008) <sp:WssX509V3Token10/>
858 (P009) </wsp:Policy>
859 (P010) </sp:X509Token>
860 (P011) </wsp:Policy>
861 (P012) </sp:InitiatorEncryptionToken>
862 (P013) <sp:RecipientSignatureToken>
863 (P014) <wsp:Policy>
864 (P015) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
865 sx/ws-securitypolicy/200702/IncludeToken/Never">
866 (P016) <wsp:Policy>
867 (P017) <sp:WssX509V3Token10/>
868 (P018) </wsp:Policy>
869 (P019) </sp:X509Token>
870 (P020) </wsp:Policy>
871 (P021) </sp:RecipientSignatureToken>
872 (P022) <sp:AlgorithmSuite>
873 (P023) <wsp:Policy>
874 (P024) <sp:Basic256/>
875 (P025) </wsp:Policy>
876 (P026) </sp:AlgorithmSuite>
877 (P027) <sp:Layout>
878 (P028) <wsp:Policy>
879 (P029) <sp:Lax/>
880 (P030) </wsp:Policy>
881 (P031) </sp:Layout>
882 (P032) <sp:IncludeTimestamp/>
883 (P033) <sp:OnlySignEntireHeadersAndBody/>
884 (P034) </wsp:Policy>
885 (P035) </sp:AsymmetricBinding>
886 (P036) <sp:Wss10>
887 (P037) <wsp:Policy>
888 (P038) <sp:MustSupportRefKeyIdentifier/>
889 (P039) <sp:MustSupportRefIssuerSerial/>
890 (P040) </wsp:Policy>
891 (P041) </sp:Wss10>
892 (P042) <sp:EncryptedSupportingTokens>
893 (P043) <wsp:Policy>
894 (P044) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-
895 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
896 (P045) <wsp:Policy>
897 (P046) <sp:HashPassword/>
898 (P047) <sp:WssUsernameToken10/>
899 (P048) </wsp:Policy>
900 (P049) </sp:UsernameToken>
901 (P050) </wsp:Policy>
902 (P051) </sp:EncryptedSupportingTokens>
903 (P052) </wsp:Policy>
904
905 (P053) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
906 (P054) <wsp:ExactlyOne>
907 (P055) <wsp:All>
908 (P056) <sp:EncryptedParts>
909 (P057) <sp:Body/>
910 (P058) </sp:EncryptedParts>
911 (P059) </wsp:All>
912 (P060) </wsp:ExactlyOne>
913 (P061) </wsp:Policy>
914
915 (P062) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
916 (P063) <wsp:ExactlyOne>

```

```

917 (P064) <wsp:All>
918 (P065) <sp:SignedParts>
919 (P066) <sp:Body/>
920 (P067) </sp:SignedParts>
921 (P068) </wsp:All>
922 (P069) </wsp:ExactlyOne>
923 (P070) </wsp:Policy>

```

924 Lines (P002) – (P035) contain the AsymmetricBinding assertion which indicates that the recipient's token
925 must be used for both message signature and encryption.

926 Lines (P004) – (P012) contain the InitiatorEncryptionToken assertion. Within that assertion lines (P006) –
927 (P010) indicate that the initiator token must be an X.509 token. Line (P008) dictates the X.509 token must
928 be an X.509v3 security token as described in the WS-Security 1.0 X.509 Token Profile, however as
929 stated on line (P006) it must not be included in any message.

930 Lines (P013) – (P021) contain the RecipientSignatureToken assertion. Within that assertion lines (P015) –
931 (P019) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509
932 Token Profile for message signature, however as stated on line (P017) it must not be included in any
933 message.

934 Line (P032) requires the inclusion of a timestamp.

935 Line (P035) OnlySignEntireHeadersAndBody assertion will only apply to the response message, as there
936 is no signature token defined for the Initiator.

937 Lines (P036) – (P041) contain some WS-Security 1.0 related interoperability requirements, specifically
938 support for key identifier, and issuer serial number.

939 Lines (P042) – (P051) contain an EncryptedSupportingTokens assertion which identifies the inclusion of
940 an additional token which must be included in the message and encrypted. Lines (P044) – (P049)
941 indicate that the supporting token must be a UsernameToken and must be included in all messages to
942 the recipient. Line (P046) dictates the UsernameToken must be hash into digest format, instead of clear
943 text password. Line (P047) dictates the UsernameToken must conform to the WS-Security 1.0
944 UsernameToken Profile.

945 Lines (P053) – (P061) contain a policy that is attached to the input message. Lines (P056) – (P058)
946 require the body of the input message must be encrypted.

947 Lines (P062) – (P070) contain a policy that is attached to the output message. Lines (P065) – (P068)
948 require the body of the output message must be signed.

949

950 An example of a request message that conforms to the above stated policy is as follows.

```

951 (M001) <?xml version="1.0" encoding="utf-8" ?>
952 (M002) <soapenv:Envelope xmlns:soapenv="..." xmlns:xenc="..." . . . >
953 (M003) <soapenv:Header>
954 (M004) <wsse:Security xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..."
955 soapenv:mustUnderstand="1" >
956 (M005) <xenc:EncryptedKey>
957 (M006) <xenc:EncryptionMethod Algorithm=". . .#rsa-oaep-mgf1p">
958 (M007) <ds:DigestMethod Algorithm=". . .#sha1" />
959 (M008) </xenc:EncryptionMethod>
960 (M009) <ds:KeyInfo>
961 (M010) <wsse:SecurityTokenReference >
962 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
963 (M012) Value="...#X509SubjectKeyIdentifier">CuJ. .
964 .=</wsse:KeyIdentifier>
965 (M013) </wsse:SecurityTokenReference>
966 (M014) </ds:KeyInfo>
967 (M015) <xenc:CipherData>
968 (M016) <xenc:CipherValue>dbj...=</xenc:CipherValue>
969 (M017) </xenc:CipherData>
970 (M018) <xenc:ReferenceList>
971 (M019) <xenc:DataReference URI="#encBody"/>
972 (M020) <xenc:DataReference URI="#encUnt"/>

```



```

973 (M021) </xenc:ReferenceList>
974 (M022) </xenc:EncryptedKey>
975 (M023) <xenc:EncryptedData Id="encUnt" Type="...#Element"
976 MimeType="text/xml" Encoding="UTF-8" >
977 (M024) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
978 (M025) <xenc:CipherData>
979 (M026) <xenc:CipherValue>KZf...=</xenc:CipherValue>
980 (M027) </xenc:CipherData>
981 (M028) </xenc:EncryptedData>
982 (M029) <wsu:Timestamp >
983 (M030) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
984 (M031) </wsu:Timestamp>
985 (M032) </wsse:Security>
986 (M033) </soapenv:Header>
987 (M034) <soapenv:Body >
988 (M035) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
989 Encoding="UTF-8" >
990 (M036) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
991 (M037) <xenc:CipherData>
992 (M038) <xenc:CipherValue>a/9...B</xenc:CipherValue>
993 (M039) </xenc:CipherData>
994 (M040) </xenc:EncryptedData>
995 (M041) </soapenv:Body>
996 (M042) </soapenv:Envelope>

```

997 Line (M020) is an encryption data reference that references the encrypted UsernameToken on lines
998 (M023) – (M028) which was required to be included by the EncryptedSupportingTokens assertion. Lines
999 (M009) – (M014) hold a KeyIdentifier of the recipient’s token used to encrypt the UsernameToken as
1000 required by the AsymmetricBinding assertion. Because the InitiatorEncryptionAssertion disallowed the
1001 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included
1002 token.

1003 Line (M019) is an encryption data reference that references the encrypted body of the message on lines
1004 (M035) – (M040). The encryption was required by the EncryptedParts assertion of the input message
1005 policy. It also uses the recipient token as identified by the KeyIdentifier.

1006 Lines (M029) – (M031) contain a timestamp for the message as required by the IncludeTimestamp
1007 assertion.

1008 Because the username token was encrypted its content prior to encryption is included below to better
1009 illustrate the reference.

```

1010 (M043) <wsse:UsernameToken wsu:Id="usernameToken">
1011 (M044) <wsse:Username>Chris</wsse:Username>
1012 (M045) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1013 wss-username-token-profile-1.0#PasswordDigest">oY...=</wsse:Password>
1014 (M046) <wsse:Nonce EncodingType="...#Base64Binary">pN...=</wsse:Nonce>
1015 (M047) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
1016 (M048) </wsse:UsernameToken>

```

1017 Line (M046) contains the Nonce element and Line (M047) contains a timestamp. It is recommended that
1018 these two elements should also be included in the PasswordText case for better security
1019 [\[WSS10-USERNAME\]](#).

1020

1021 **2.1.4 (WSS 1.1), User Name with Certificates, Sign, Encrypt**

1022 This scenario is based on the “Examples of Secure Web Service Message Exchange Document”
1023 [\[WS-SECURE-INTEROP\]](#).

1024 The use case here is the following: the Initiator generates a symmetric key; the symmetric key is
1025 encrypted using the Recipient’s certificate and placed in an encrypted key element. The UsernameToken
1026 identifying the Requestor and message body are signed using the symmetric key. The body and

1027 UsernameToken are also encrypted. The Authority for this request is generally the Subject of the
1028 Initiator's X509 certificate.

1029 We can use the symmetric security binding [WSSP] with X509token as the protection token to illustrate
1030 this case. If derived keys are to be used, then the derived keys property of X509Token should be set.

1031 The policy is as follows:

```
1032 (P001) <wsp:Policy wsu:Id="WSS11UsernameWithCertificates_policy">
1033 (P002)   <wsp:ExactlyOne>
1034 (P003)   <wsp>All>
1035 (P004)     <sp:SymmetricBinding>
1036 (P005)       <wsp:Policy>
1037 (P006)         <sp:ProtectionToken>
1038 (P007)           <wsp:Policy>
1039 (P008)             <sp:X509Token sp:IncludeToken="http://docs.oasis-
1040 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
1041 (P009)               <wsp:Policy>
1042 (P010)                 <sp:RequireThumbprintReference/>
1043 (P011)                 <sp:WssX509V3Token11/>
1044 (P012)                 </wsp:Policy>
1045 (P013)                 </sp:X509Token>
1046 (P014)                 </wsp:Policy>
1047 (P015)             </sp:ProtectionToken>
1048 (P016)           <sp:AlgorithmSuite>
1049 (P017)             <wsp:Policy>
1050 (P018)               <sp:Basic256/>
1051 (P019)               </wsp:Policy>
1052 (P020)             </sp:AlgorithmSuite>
1053 (P021)           <sp:Layout>
1054 (P022)             <wsp:Policy>
1055 (P023)               <sp:Strict/>
1056 (P024)               </wsp:Policy>
1057 (P025)             </sp:Layout>
1058 (P026)           <sp:IncludeTimestamp/>
1059 (P027)           <sp:OnlySignEntireHeadersAndBody/>
1060 (P028)           </wsp:Policy>
1061 (P029)         </sp:SymmetricBinding>
1062 (P030)       <sp:SignedEncryptedSupportingTokens>
1063 (P031)         <wsp:Policy>
1064 (P032)           <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
1065 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1066 (P033)             <wsp:Policy>
1067 (P034)               <sp:WssUsernameToken11/>
1068 (P035)               </wsp:Policy>
1069 (P036)             </sp:UsernameToken>
1070 (P037)             </wsp:Policy>
1071 (P038)           </sp:SignedEncryptedSupportingTokens>
1072 (P039)         <sp:Wss11>
1073 (P040)           <wsp:Policy>
1074 (P041)             <sp:MustSupportRefKeyIdentifier/>
1075 (P042)             <sp:MustSupportRefIssuerSerial/>
1076 (P043)             <sp:MustSupportRefThumbprint/>
1077 (P044)             <sp:MustSupportRefEncryptedKey/>
1078 (P045)             </wsp:Policy>
1079 (P046)           </sp:Wss11>
1080 (P047)         </wsp>All>
1081 (P048)       </wsp:ExactlyOne>
1082 (P049)     </wsp:Policy>
1083
1084 (P050) <wsp:Policy wsu:Id="UsernameForCertificates_input_policy">
1085 (P051)   <wsp:ExactlyOne>
1086 (P052)   <wsp>All>
1087 (P053)     <sp:SignedParts>
1088 (P054)       <sp:Body/>
1089 (P055)     </sp:SignedParts>
```

```

1090 (P056) <sp:EncryptedParts>
1091 (P057) <sp:Body/>
1092 (P058) </sp:EncryptedParts>
1093 (P059) </wsp:All>
1094 (P060) </wsp:ExactlyOne>
1095 (P061) </wsp:Policy>
1096
1097 (P062) <wsp:Policy wsu:Id="UsernameForCertificate_output_policy">
1098 (P063) <wsp:ExactlyOne>
1099 (P064) <wsp:All>
1100 (P065) <sp:SignedParts>
1101 (P066) <sp:Body/>
1102 (P067) </sp:SignedParts>
1103 (P068) <sp:EncryptedParts>
1104 (P069) <sp:Body/>
1105 (P070) </sp:EncryptedParts>
1106 (P071) </wsp:All>
1107 (P072) </wsp:ExactlyOne>
1108 (P073) </wsp:Policy>

```

1109 Lines (P004) – (P297) contain a SymmetricBinding assertion which indicates the use of one token to both
1110 sign and encrypt a message.

1111 Lines (P006) – (P015) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P012)
1112 indicate that the protection token must be an X.509 token that must never be included in any messages in
1113 the message exchange. Line (P010) dictates the X.509 token must be an X.509v3 security token as
1114 described in the WS-Security 1.1 X.509 Token Profile. Line (P011) dicates a thumbprint reference must
1115 be used to identify the token in any message.

1116 Line (P026) requires the inclusion of a timestamp.

1117 Lines (P030) – (P038) contain a SignedEncryptedSupportingTokens assertion which identifies the
1118 inclusion of an additional token which must be included in the message signature and encrypted. Lines
1119 (P032) – (P036) indicate that the supporting token must be a UsernameToken and must be included in all
1120 messages to the recipient. Line (P034) dictates the UsernameToken must conform to the WS-Security 1.1
1121 UsernameToken Profile.

1122 Lines (P040) – (P046) contain some WS-Security 1.1 related interoperability requirements, specifically
1123 support for key identifier, issuer serial number, thumbprint, and encrypted key references.

1124 Lines (P050) – (P061) contain a policy that is attached to the input message. Lines (P053) – (P055)
1125 require that the body of the input message must be signed. Lines (P056) – (P058) require the body of the
1126 input message must be encrypted.

1127 Lines (P062) – (P073) contain a policy that is attached to the output message. Lines (P065) – (P067)
1128 require that the body of the output message must be signed. Lines (P068) – (P070) require the body of
1129 the output message must be encrypted.

1130 An example of an input message that conforms to the above stated policy is as follows.

```

1131 (M001) <?xml version="1.0" encoding="utf-8" ?>
1132 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1133 (M003) <soap:Header>
1134 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..."
1135 xmlns:wsu="...">
1136 (M005) <xenc:EncryptedKey wsu:Id="EK">
1137 (M006) <xenc:EncryptionMethod
1138 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
1139 (M007) <ds:KeyInfo>
1140 (M008) <wsse:SecurityTokenReference>
1141 (M009) <wsse:KeyIdentifier ValueType="http://docs.oasis-
1142 open.org/wss/oasis-wss-soap-message-security-1.1#ThumbPrintSHA1">
1143 (M010) LKiQ/CmFrJDJqCLFcjlhIsmZ/+0=
1144 (M011) </wsse:KeyIdentifier>
1145 (M012) </wsse:SecurityTokenReference>
1146 (M013) </ds:KeyInfo>
1147 (M014) </xenc:EncryptedKey>

```

```

1148 (M015) <xenc:ReferenceList>
1149 (M016) <xenc:DataReference URI="#encUT"/>
1150 (M017) <xenc:DataReference URI="#encBody"/>
1151 (M018) </xenc:ReferenceList>
1152 (M019) <wsu:Timestamp wsu:Id="T0">
1153 (M020) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1154 (M021) </wsu:Timestamp>
1155 (M022) <xenc:EncryptedData wsu:Id="encUT">
1156 (M023) <xenc:EncryptionMethod
1157 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1158 (M024) <ds:KeyInfo>
1159 (M025) <wsse:SecurityTokenReference>
1160 (M026) <wsse:Reference URI="#EK"/>
1161 (M027) </wsse:SecurityTokenReference>
1162 (M028) </ds:KeyInfo>
1163 (M029) <xenc:CipherData>
1164 (M030) <xenc:CipherValue>...</xenc:CipherValue>
1165 (M031) </xenc:CipherData>
1166 (M032) </xenc:EncryptedData>
1167 (M033) <ds:Signature>
1168 (M034) <ds:SignedInfo>...
1169 (M035) <ds:Reference URI="#T0">...</ds:Reference>
1170 (M036) <ds:Reference URI="#usernameToken">...</ds:Reference>
1171 (M037) <ds:Reference URI="#body">...</ds:Reference>
1172 (M038) </ds:SignedInfo>
1173 (M039) <ds:SignatureValue>HFLP...</ds:SignatureValue>
1174 (M040) <ds:KeyInfo>
1175 (M041) <wsse:SecurityTokenReference>
1176 (M042) <wsse:Reference URI="#EK"/>
1177 (M043) </wsse:SecurityTokenReference>
1178 (M044) </ds:KeyInfo>
1179 (M045) </ds:Signature>
1180 (M046) </wsse:Security>
1181 (M047) </soap:Header>
1182 (M048) <soap:Body wsu:Id="body">
1183 (M049) <xenc:EncryptedData wsu:Id="encBody">
1184 (M050) <xenc:EncryptionMethod
1185 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1186 (M051) <ds:KeyInfo>
1187 (M052) <wsse:SecurityTokenReference>
1188 (M053) <wsse:Reference URI="#EK"/>
1189 (M054) </wsse:SecurityTokenReference>
1190 (M055) </ds:KeyInfo>
1191 (M056) <xenc:CipherData>
1192 (M057) <xenc:CipherValue>...</xenc:CipherValue>
1193 (M058) </xenc:CipherData>
1194 (M059) </xenc:EncryptedData>
1195 (M060) </soap:Body>
1196 (M061) </soap:Envelope>

```

1197 Lines (M005) – (M014) contain the encrypted symmetric key as required by the use of the
1198 SymmetricBinding assertion starting on line (P004) with an X509Token ProtectionToken assertion. Line
1199 (M006) references the KwRsaOaep Asymmetric Key Wrap algorithm dictated by the Basic 256 Algorithm
1200 Suite assertion on line (P018). Lines (M009) – (M011) hold a KeyIdentifier of the protection token used to
1201 encrypt the symmetric key as required by the SymmetricBinding assertion. Because the ProtectionToken
1202 assertion disallowed the token from being inserted into the message and instead required a thumbprint
1203 reference, a thumbprint reference is included to identify the token.

1204 Line (M016) is an encryption data reference that references the encrypted supporting UsernameToken on
1205 lines (M022) – (M032). The encryption was required by the SignedEncryptedSupportingTokens assertion
1206 on line (P038). Line (M023) references the Aes256 Encryption algorithm dictated by the Basic 256
1207 Algorithm Suite assertion on line (P018). The encrypted symmetric key is used to encrypt the
1208 UsernameToken as referenced on line (M026).

1209 Line (M017) is an encryption data reference that references the encrypted body of the message on lines
1210 (M049) – (M059). The encryption was required by the EncryptedParts assertion of the input message
1211 policy. The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line
1212 (M053).

1213 Lines (M019) – (M021) contain a timestamp for the message as required by the IncludeTimestamp
1214 assertion.

1215 Lines (M033) – (M045) contain the message signature.

1216 Line (M035) indicates the message timestamp is included in the signature as required by the
1217 IncludeTimestamp assertion definition.

1218 Line (M036) indicates the supporting UsernameToken is included in the signature as required by the
1219 SignedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is
1220 included below to better illustrate the reference.

```
1221 (M062) <wsse:UsernameToken wsu:Id="usernameToken">  
1222 (M063) <wsse:Username>Chris</wsse:Username>  
1223 (M064) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-  
1224 200401-wss-username-token-profile-1.0#PasswordText ">sirhC</wsse:Password>  
1225 (M065) </wsse:UsernameToken>
```

1226 Line (M037) indicates the message body is included in the signature as required by the SignedParts
1227 assertion of the input message policy.

1228 Line (M042) references the encrypted symmetric key for signing as dictated by the SymmetricBinding
1229 assertion.

1230 2.2 X.509 Token Authentication Scenario Assertions

1231 2.2.1 (WSS1.0) X.509 Certificates, Sign, Encrypt

1232 This use-case corresponds to the situation where both parties have X.509v3 certificates (and public-
1233 private key pairs). The requestor identifies itself to the service. The message exchange is integrity
1234 protected and encrypted.

1235 This modeled by use of an asymmetric security binding assertion.

1236 The message level policies in this and subsequent sections cover a different scope of the web service
1237 definition than the security binding level policy and so appear as separate policies and are attached at
1238 WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a
1239 set of coordinated policies one with endpoint subject and two with message subjects to achieve this use
1240 case.

1241 The policies are as follows:

```
1242 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >  
1243 (P002) <wsp:ExactlyOne>  
1244 (P003) <wsp:All>  
1245 (P004) <sp:AsymmetricBinding>  
1246 (P005) <wsp:Policy>  
1247 (P006) <sp:InitiatorToken>  
1248 (P007) <wsp:Policy>  
1249 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1250 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
1251 (P009) <wsp:Policy>  
1252 (P010) <sp:WssX509V3Token10/>  
1253 (P011) </wsp:Policy>  
1254 (P012) </sp:X509Token>  
1255 (P013) </wsp:Policy>  
1256 (P014) </sp:InitiatorToken>  
1257 (P015) <sp:RecipientToken>  
1258 (P016) <wsp:Policy>  
1259 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1260 sx/ws-securitypolicy/200702/IncludeToken/Never">
```

```

1261 (P018) <wsp:Policy>
1262 (P019) <sp:WssX509V3Token10/>
1263 (P020) </wsp:Policy>
1264 (P021) </sp:X509Token>
1265 (P022) </wsp:Policy>
1266 (P023) </sp:RecipientToken>
1267 (P024) <sp:AlgorithmSuite>
1268 (P025) <wsp:Policy>
1269 (P026) <sp:Basic256/>
1270 (P027) </wsp:Policy>
1271 (P028) </sp:AlgorithmSuite>
1272 (P029) <sp:Layout>
1273 (P030) <wsp:Policy>
1274 (P031) <sp:Strict/>
1275 (P032) </wsp:Policy>
1276 (P033) </sp:Layout>
1277 (P034) <sp:IncludeTimestamp/>
1278 (P035) <sp:OnlySignEntireHeadersAndBody/>
1279 (P036) </wsp:Policy>
1280 (P037) </sp:AsymmetricBinding>
1281 (P038) <sp:Wss10>
1282 (P039) <wsp:Policy>
1283 (P040) <sp:MustSupportRefKeyIdentifier/>
1284 (P041) </wsp:Policy>
1285 (P042) </sp:Wss10>
1286 (P043) </wsp:All>
1287 (P044) </wsp:ExactlyOne>
1288 (P045) </wsp:Policy>
1289
1290 (P046) <wsp:Policy wsu:Id="WSS10Anonymous_with_Certificates_input_policy">
1291 (P047) <wsp:ExactlyOne>
1292 (P048) <wsp:All>
1293 (P049) <sp:SignedParts>
1294 (P050) <sp:Body/>
1295 (P051) </sp:SignedParts>
1296 (P052) <sp:EncryptedParts>
1297 (P053) <sp:Body/>
1298 (P054) </sp:EncryptedParts>
1299 (P055) </wsp:All>
1300 (P056) </wsp:ExactlyOne>
1301 (P057) </wsp:Policy>
1302
1303 (P058) <wsp:Policy wsu:Id="WSS10anonymous_with_certs_output_policy">
1304 (P059) <wsp:ExactlyOne>
1305 (P060) <wsp:All>
1306 (P061) <sp:SignedParts>
1307 (P062) <sp:Body/>
1308 (P063) </sp:SignedParts>
1309 (P064) <sp:EncryptedParts>
1310 (P065) <sp:Body/>
1311 (P066) </sp:EncryptedParts>
1312 (P067) </wsp:All>
1313 (P068) </wsp:ExactlyOne>
1314 (P069) </wsp:Policy>

```

1315 Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token
1316 must be used for the message signature and the recipient's token must be used for message encryption.

1317 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)
1318 indicate that the initiator token must be an X.509 token that must be included with all messages sent to
1319 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the
1320 WS-Security 1.0 X.509 Token Profile.

1321 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)
1322 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token

1323 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to
 1324 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the
 1325 token in any messages where the token is used.

1326 Line (P034) requires the inclusion of a timestamp.

1327 Lines (P046) – (P057) contain a policy that is attached to the input message. Lines (P049) – (P051)
 1328 require that the body of the input message must be signed. Lines (P052) – (P054) require the body of the
 1329 input message must be encrypted.

1330 Lines (P058) – (P069) contain a policy that is attached to the output message. Lines (P061) – (P063)
 1331 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of
 1332 the output message must be encrypted.

1333 An example of an input message that conforms to the above stated policy is as follows.

```

1334 (M001) <?xml version="1.0" encoding="utf-8" ?>
1335 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1336 (M003)   <soap:Header>
1337 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
1338 (M005)       <xenc:EncryptedKey >
1339 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1340 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1341 (M008)         </xenc:EncryptionMethod>
1342 (M009)         <ds:KeyInfo>
1343 (M010)           <wsse:SecurityTokenReference >
1344 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1345 (M011) ValueType="...#X509SubjectKeyIdentifier">
1346 (M012)               MIGfMa0GCSq...
1347 (M013)             </wsse:KeyIdentifier>
1348 (M014)           </ds:KeyInfo>
1349 (M015)           <xenc:CipherData>
1350 (M016)             <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1351 (M017)           </xenc:CipherData>
1352 (M018)           <xenc:ReferenceList>
1353 (M019)             <xenc:DataReference URI="#encBody"/>
1354 (M020)           </xenc:ReferenceList>
1355 (M021)         </xenc:EncryptedKey>
1356 (M022)         <wsu:Timestamp wsu:Id="T0">
1357 (M023)           <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1358 (M024)         </wsu:Timestamp>
1359 (M025)         <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="...#X509v3"
1360 (M025) EncodingType="...#Base64Binary">
1361 (M026)           MIIIEZzCCA9CgAwIBAgIQEmtJZc0...
1362 (M027)         </wsse:BinarySecurityToken>
1363 (M028)         <ds:Signature>
1364 (M029)           <ds:SignedInfo>...
1365 (M030)             <ds:Reference URI="#T0">...</ds:Reference>
1366 (M031)             <ds:Reference URI="#body">...</ds:Reference>
1367 (M032)           </ds:SignedInfo>
1368 (M033)           <ds:SignatureValue>HFLP...</ds:SignatureValue>
1369 (M034)         <ds:KeyInfo>
1370 (M035)           <wsse:SecurityTokenReference>
1371 (M036)             <wsse:Reference URI="#binaryToken"/>
1372 (M037)           </wsse:SecurityTokenReference>
1373 (M038)         </ds:KeyInfo>
1374 (M039)       </ds:Signature>
1375 (M040)     </wsse:Security>
1376 (M041)   </soap:Header>
1377 (M042)   <soap:Body wsu:Id="body">
1378 (M043)     <xenc:EncryptedData wsu:Id="encBody">
1379 (M044)       <xenc:CipherData>
1380 (M045)         <xenc:CipherValue>...</xenc:CipherValue>
1381 (M046)       </xenc:CipherData>
1382 (M047)     </xenc:EncryptedData>
1383 (M048)   </soap:Body>

```

1384 (M049) </soap:Envelope>

1385 Line (M019) is an encryption data reference that references the encrypted body of the message on lines
 1386 (M043) – (M047). The encryption was required by the EncryptedParts assertion of the input message
 1387 policy. Lines (M011) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as
 1388 required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the
 1389 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included
 1390 token.

1391 Lines (M022) – (M024) contain a timestamp for the message as required by the IncludeTimestamp
 1392 assertion.

1393 Lines (M025) – (M027) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as
 1394 required by the InitiatorToken assertion.

1395 Lines (M028) – (M039) contain the message signature.

1396 Line (M030) indicates the message timestamp is included in the signature as required by the
 1397 IncludeTimestamp assertion definition.

1398 Line (M031) indicates the message body is included in the signature as required by the SignedParts
 1399 assertion of the input message policy.

1400 Note that the initiator’s BinarySecurityToken is not included in the message signature as it was not
 1401 required by policy.

1402 Lines (M035) – (M037) references the initiator’s BinarySecurityToken included in the message for
 1403 identifying the key used for signing as dictated by the AsymmetricBinding assertion.

1404 2.2.2 (WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt

1405 This scenario is based on WSS Interop, Scenario 3, [Web Services Security: Interop 1](#), Draft 06, Editor,
 1406 Hal Lockhart, BEA Systems

1407 This use case corresponds to the situation where both parties have X.509v3 certificates (and public-
 1408 private key pairs). The requestor wishes to identify itself to the service using its X.509 credential (strong
 1409 authentication). The message exchange needs to be integrity protected and encrypted as well. The
 1410 difference from previous use case is that the X509 token inserted by the client is included in the message
 1411 signature (see <ProtectTokens />).

1412 The policy is as follows:

```

1413 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1414   (P002)   <wsp:ExactlyOne>
1415     (P003)   <wsp>All>
1416       (P004)   <sp:AsymmetricBinding>
1417         (P005)   <wsp:Policy>
1418           (P006)   <sp:InitiatorToken>
1419             (P007)   <wsp:Policy>
1420               (P008)   <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1421                 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1422                 (P009)   <wsp:Policy>
1423                   (P010)   <sp:WssX509V3Token10/>
1424                   </wsp:Policy>
1425                 </sp:X509Token>
1426               </wsp:Policy>
1427             </sp:InitiatorToken>
1428           <sp:RecipientToken>
1429             (P016)   <wsp:Policy>
1430               (P017)   <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1431                 sx/ws-securitypolicy/200702/IncludeToken/Never">
1432                 (P018)   <wsp:Policy>
1433                   (P019)   <sp:WssX509V3Token10/>
1434                   </wsp:Policy>
1435                 </sp:X509Token>
1436               </wsp:Policy>
1437             </sp:RecipientToken>

```



```

1438 (P024) <sp:AlgorithmSuite>
1439 (P025) <wsp:Policy>
1440 (P026) <sp:Basic256/>
1441 (P027) </wsp:Policy>
1442 (P028) </sp:AlgorithmSuite>
1443 (P029) <sp:Layout>
1444 (P030) <wsp:Policy>
1445 (P031) <sp:Strict/>
1446 (P032) </wsp:Policy>
1447 (P033) </sp:Layout>
1448 (P034) <sp:IncludeTimestamp/>
1449 (P035) <sp:ProtectTokens />
1450 (P036) <sp:OnlySignEntireHeadersAndBody/>
1451 (P037) </wsp:Policy>
1452 (P038) </sp:AsymmetricBinding>
1453 (P039) <sp:Wss10>
1454 (P040) <wsp:Policy>
1455 (P041) <sp:MustSupportRefKeyIdentifier/>
1456 (P042) </wsp:Policy>
1457 (P043) </sp:Wss10>
1458 (P044) </wsp:All>
1459 (P045) </wsp:ExactlyOne>
1460 (P046) </wsp:Policy>
1461
1462 (P047) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1463 (P048) <wsp:ExactlyOne>
1464 (P049) <wsp:All>
1465 (P050) <sp:SignedParts>
1466 (P051) <sp:Body/>
1467 (P052) </sp:SignedParts>
1468 (P053) <sp:EncryptedParts>
1469 (P054) <sp:Body/>
1470 (P055) </sp:EncryptedParts>
1471 (P056) </wsp:All>
1472 (P057) </wsp:ExactlyOne>
1473 (P058) </wsp:Policy>
1474
1475 (P059) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1476 (P060) <wsp:ExactlyOne>
1477 (P061) <wsp:All>
1478 (P062) <sp:SignedParts>
1479 (P063) <sp:Body/>
1480 (P064) </sp:SignedParts>
1481 (P065) <sp:EncryptedParts>
1482 (P066) <sp:Body/>
1483 (P067) </sp:EncryptedParts>
1484 (P068) </wsp:All>
1485 (P069) </wsp:ExactlyOne>
1486 (P070) </wsp:Policy>

```

1487 Lines (P004) – (P038) contain the AsymmetricBinding assertion which indicates that the initiator’s token
1488 must be used for the message signature and the recipient’s token must be used for message encryption.

1489 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)
1490 indicate that the initiator token must be an X.509 token that must be included with all messages sent to
1491 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the
1492 WS-Security 1.0 X.509 Token Profile.

1493 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)
1494 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token
1495 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to
1496 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the
1497 token in any messages where the token is used.

1498 Line (P034) requires the inclusion of a timestamp.

1499 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the
1500 token used to generate that signature.

1501 Lines (P047) – (P058) contain a policy that is attached to the input message. Lines (P050) – (P052)
1502 require that the body of the input message must be signed. Lines (P053) – (P055) require the body of the
1503 input message must be encrypted.

1504 Lines (P059) – (P070) contain a policy that is attached to the output message. Lines (P062) – (P064)
1505 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of
1506 the output message must be encrypted.

1507 An example of an input message that conforms to the above stated policy is as follows.

```

1508 (M001) <?xml version="1.0" encoding="utf-8" ?>
1509 (M002) <soapenv:Envelope xmlns:soapenv="..."
1510 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="..." xmlns:xenc="..." ...>
1511 (M003)   <soapenv:Header>
1512 (M004)     <wsse:Security xmlns:wsse="..." xmlns:ds="..."
1513 soapenv:mustUnderstand="1">
1514 (M005)       <xenc:EncryptedKey >
1515 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1516 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1517 (M008)         </xenc:EncryptionMethod>
1518 (M009)         <ds:KeyInfo>
1519 (M010)           <wsse:SecurityTokenReference >
1520 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1521 ValueType="...#X509SubjectKeyIdentifier">CuJd...=</wsse:KeyIdentifier>
1522 (M012)           </wsse:SecurityTokenReference>
1523 (M013)         </ds:KeyInfo>
1524 (M014)         <xenc:CipherData>
1525 (M015)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1526 (M016)         </xenc:CipherData>
1527 (M017)         <xenc:ReferenceList>
1528 (M018)           <xenc:DataReference URI="#encBody"/>
1529 (M019)         </xenc:ReferenceList>
1530 (M020)       </xenc:EncryptedKey>
1531 (M021)       <wsu:Timestamp wsu:Id="Timestamp" >
1532 (M022)         <wsu:Created>2007-03-26T16:53:39Z</wsu:Created>
1533 (M023)       </wsu:Timestamp>
1534 (M024)       <wsse:BinarySecurityToken wsu:Id="bst" ValueType="...#X509v3"
1535 EncodingType="...#Base64Binary">MIID...=</wsse:BinarySecurityToken>
1536 (M025)       <ds:Signature>
1537 (M026)         <ds:SignedInfo>
1538 (M027)           <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#" />
1539 (M028)           <ds:SignatureMethod Algorithm="...#rsa-sha1"/>
1540 (M029)           <ds:Reference URI="#Timestamp">
1541 (M030)             <ds:Transforms>... </ds:Transforms>
1542 (M031)             <ds:DigestMethod Algorithm="...#sha1"/>
1543 (M032)             <ds:DigestValue>+g0I...=</ds:DigestValue>
1544 (M033)           </ds:Reference>
1545 (M034)           <ds:Reference URI="#Body">...</ds:Reference>
1546 (M035)           <ds:Reference URI="#bst">...</ds:Reference>
1547 (M036)         </ds:SignedInfo>
1548 (M037)         <ds:SignatureValue>RRT...=</ds:SignatureValue>
1549 (M038)       </ds:Signature>
1550 (M039)       <wsse:SecurityTokenReference >
1551 (M040)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1552 ValueType="...#X509SubjectKeyIdentifier">Xeg5...=</wsse:KeyIdentifier>
1553 (M041)       </wsse:SecurityTokenReference>
1554 (M042)     </ds:KeyInfo>
1555 (M043)   </ds:Signature>
1556 (M044) </wsse:Security>
1557 (M045) </soapenv:Header>
1558 (M046) <soapenv:Body wsu:Id="Body" >
1559 (M047)   <xenc:EncryptedData Id="encBody" Type="...#Content"
1560 (M048)     MimeTypes="text/xml" Encoding="UTF-8" >

```

```

1561 (M049) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1562 (M050) <xenc:CipherData>
1563 (M051) <xenc:CipherValue>W84fn...1</xenc:CipherValue>
1564 (M052) </xenc:CipherData>
1565 (M053) </xenc:EncryptedData>
1566 (M054) </soapenv:Body>
1567 (M055) </soapenv:Envelope>

```

1568 Line (M018) is an encryption data reference that references the encrypted body of the message on lines
1569 (M047) – (M048). The encryption was required by the EncryptedParts assertion of the input message
1570 policy. Lines (M009) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as
1571 required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the
1572 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included
1573 token.

1574 Lines (M021) – (M023) contain a timestamp for the message as required by the IncludeTimestamp
1575 assertion.

1576 Line (M024) contains the BinarySecurityToken holding the X.509v3 certificate of the initiator as required
1577 by the InitiatorToken assertion.

1578 Lines (M025) – (M043) contain the message signature.

1579 Line (M029) indicates the message timestamp is included in the signature as required by the
1580 IncludeTimestamp assertion definition.

1581 Line (M034) indicates the message body is included in the signature as required by the SignedParts
1582 assertion of the input message policy.

1583 Line (M035) indicates the BinarySecurityToken on Line (M024) is included in the signature as required by
1584 the ProtectTokens assertion of the AsymmetricBinding assertion policy.

1585 Note that the recipient’s token is not explicitly included in the security header, as it is required not to be by
1586 policy (P017). Instead a KeyIdentifier, line (M011) is used to identify the recipient’s that should be used to
1587 decrypt the EncryptedData.

1588 Lines (M039) – (M041) reference the initiator’s BinarySecurityToken on line (M034), which is included in
1589 the message to contain the key used for verifying as dictated by the AsymmetricBinding assertion.

1590 **2.2.2.1 (WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption**

1591 This scenario is based on WSS Interop, Scenario 4, [Web Services Security: Interop 2](#).

1592 A common variation on the previous example is where X.509 Certificates are still used for authentication,
1593 but a mutually agreed upon symmetric key is used for encryption.

1594 In this use case the policy is the same except that the InitiatorToken and RecipientToken are now each
1595 just SignatureTokens.

1596 A second variation in this use case is that the mutually agreed upon symmetric encryption key is
1597 characterized as an “IssuedToken” with a mutually agreed upon URI used to identify the out of band (not
1598 included in the message) token, which is included in a SupportingTokens element.

1599 The policy is as follows:

```

1600 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1601 (P002) <wsp:ExactlyOne>
1602 (P003) <wsp:All>
1603 (P004) <sp:AsymmetricBinding>
1604 (P005) <wsp:Policy>
1605 (P006) <sp:InitiatorSignatureToken>
1606 (P007) <wsp:Policy>
1607 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1608 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1609 (P009) <wsp:Policy>
1610 (P010) <sp:WssX509V3Token10/>
1611 (P011) </wsp:Policy>
1612 (P012) </sp:X509Token>
1613 (P013) </wsp:Policy>

```

```

1614 (P014) </sp:InitiatorSignatureToken>
1615 (P015) <sp:RecipientSignatureToken>
1616 (P016) <wsp:Policy>
1617 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1618 sx/ws-securitypolicy/200702/IncludeToken/Never">
1619 (P018) <wsp:Policy>
1620 (P019) <sp:WssX509V3Token10/>
1621 (P020) </wsp:Policy>
1622 (P021) </sp:X509Token>
1623 (P022) </wsp:Policy>
1624 (P023) </sp:RecipientSignatureToken>
1625 (P024) <sp:AlgorithmSuite>
1626 (P025) <wsp:Policy>
1627 (P026) <sp:Basic256/>
1628 (P027) </wsp:Policy>
1629 (P028) </sp:AlgorithmSuite>
1630 (P029) <sp:Layout>
1631 (P030) <wsp:Policy>
1632 (P031) <sp:Strict/>
1633 (P032) </wsp:Policy>
1634 (P033) </sp:Layout>
1635 (P034) <sp:IncludeTimestamp/>
1636 (P035) <sp:ProtectTokens />
1637 (P036) <sp:OnlySignEntireHeadersAndBody/>
1638 (P037) </wsp:Policy>
1639 (P038) </sp:AsymmetricBinding>
1640 (P039) <sp:SupportingTokens>
1641 (P040) <wsp:Policy>
1642 (P041) <sp:IssuedToken>
1643 (P042) <sp:Issuer>SomeMutuallyAgreedURI</sp:Issuer>
1644 (P043) <sp:RequireExternalReference/>
1645 (P044) </sp:IssuedToken>
1646 (P045) <sp:EncryptedParts>
1647 (P046) <sp:Body/>
1648 (P047) </sp:EncryptedParts>
1649 (P048) </wsp:Policy>
1650 (P049) </sp:SupportingTokens>
1651 (P050) <sp:Wss10>
1652 (P051) <wsp:Policy>
1653 (P052) <sp:MustSupportRefKeyIdentifier/>
1654 (P053) </wsp:Policy>
1655 (P054) </sp:Wss10>
1656 (P055) </wsp:All>
1657 (P056) </wsp:ExactlyOne>
1658 (P057) </wsp:Policy>
1659
1660 (P058) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1661 (P059) <wsp:ExactlyOne>
1662 (P060) <wsp:All>
1663 (P061) <sp:SignedParts>
1664 (P062) <sp:Body/>
1665 (P063) </sp:SignedParts>
1666 (P064) <sp:EncryptedParts>
1667 (P065) <sp:Body/>
1668 (P066) </sp:EncryptedParts>
1669 (P067) </wsp:All>
1670 (P068) </wsp:ExactlyOne>
1671 (P069) </wsp:Policy>
1672
1673 (P070) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1674 (P071) <wsp:ExactlyOne>
1675 (P072) <wsp:All>
1676 (P073) <sp:SignedParts>
1677 (P074) <sp:Body/>

```

```

1678 (P075)      </sp:SignedParts>
1679 (P076)      <sp:EncryptedParts>
1680 (P077)      <sp:Body/>
1681 (P078)      </sp:EncryptedParts>
1682 (P079)      </wsp:All>
1683 (P080)      </wsp:ExactlyOne>
1684 (P081) </wsp:Policy>

```

1685 Lines (P004) – (P038) contain the AsymmetricBindingAssertion which indicates that the initiator’s token
1686 must be used for the message signature and the recipient’s token must be used for message encryption.

1687 Lines (P006) – (P014) contain the InitiatorSignatureToken assertion. Within that assertion lines (P008) –
1688 (P012) indicate that the initiator token must be an X.509 token that must be included with all messages
1689 sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as
1690 described in the WS-Security 1.0 X.509 Token Profile. By specifying that this is an
1691 InitiatorSignatureToken, it will only be used to sign the message and not used for encryption.

1692 Lines (P015) – (P023) contain the RecipientSignatureToken assertion. Within that assertion lines (P017)
1693 – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0
1694 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead,
1695 according to the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to
1696 identify the token in any messages where the token is used. By specifying that this is an
1697 RecipientSignatureToken, it will only be used to sign the response and not used for encryption.

1698 Line (P034) requires the inclusion of a timestamp.

1699 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the
1700 token or token reference associated with the generation of that signature.

1701 Lines (P039) – (P049) contain a SupportingTokens assertion that contains an IssuedToken (P041) –
1702 (P044), which contains an Issuer element (P042) that identifies an explicit URI
1703 (“SomeMutuallyAgreedURI”) that must be used to indicate what token will be used. (Since the content of
1704 the IssuedToken element is specific to the Issuer, any mechanism can be used to identify the key, and in
1705 this case the simplest method of identifying the issuer with the explicit key has been chosen only to
1706 illustrate one possible method, but not to recommend as opposed to any other method.) Line (P043),
1707 RequireExternalReference indicates that the IssuedToken requires a token that is referenced external to
1708 the message. Lines (P045) – (P047) indicate that the token is to be used for encrypting parts of the
1709 message, explicitly, line (P046), the Body of the message.

1710 Lines (P058) – (P069) contain a policy that is attached to the input message. Lines (P061) – (P063)
1711 require that the body of the input message must be signed. Lines (P064) – (P066) require the body of the
1712 input message must be encrypted.

1713 Lines (P070) – (P081) contain a policy that is attached to the output message. Lines (P073) – (P075)
1714 require that the body of the output message must be signed. Lines (P076) – (P078) require the body of
1715 the output message must be encrypted.

1716

1717 The following example message is derived from the WSS Interop2 document.

1718

```

1719 (M001)      <?xml version="1.0" encoding="utf-8" ?>
1720 (M002)      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1721 (M003)          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1722 (M004)          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1723 (M005)      <soap:Header>
1724 (M006)          <wsse:Security soap:mustUnderstand="1"
1725 (M007)              xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1726 (M008)          <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1727 (M009)              <xenc:DataReference URI="#enc" />
1728 (M010)          </xenc:ReferenceList>
1729 (M011)          <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1730 (M012)              wsu:Id="timestamp">
1731 (M013)              <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1732 (M014)          </wsu:Timestamp>
1733 (M015)          <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1734 (M016)              EncodingType="wsse:Base64Binary">

```

```

1735 (M017)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1736 (M018)      wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1737 (M019)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1738 (M020)      <SignedInfo>
1739 (M021)      <CanonicalizationMethod
1740 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1741 (M023)      <SignatureMethod
1742 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1743 (M025)      <Reference URI="#body">
1744 (M026)      <Transforms>
1745 (M027)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1746 (M028)      </Transforms>
1747 (M029)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1748 (M030)      <DigestValue>QTV...dw</DigestValue>
1749 (M031)      </Reference>
1750 (M032)      <Reference URI="#myCert">
1751 (M033)      <Transforms>
1752 (M034)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1753 (M035)      </Transforms>
1754 (M036)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1755 (M037)      <DigestValue>XYZ...ab</DigestValue>
1756 (M038)      </Reference>
1757 (M039)      <Reference URI="#timestamp">
1758 (M040)      <Transforms>
1759 (M041)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1760 (M042)      </Transforms>
1761 (M043)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1762 (M044)      <DigestValue>XYZ...ab</DigestValue>
1763 (M045)      </Reference>
1764 (M046)      </SignedInfo>
1765 (M047)      <SignatureValue>H+x0...gUw</SignatureValue>
1766 (M048)      <KeyInfo>
1767 (M049)      <wsse:SecurityTokenReference>
1768 (M050)      <wsse:Reference URI="#myCert" />
1769 (M051)      </wsse:SecurityTokenReference>
1770 (M052)      </KeyInfo>
1771 (M053)      </Signature>
1772 (M054)      </wsse:Security>
1773 (M055)      </soap:Header>
1774 (M056)      <soap:Body wsu:Id="body"
1775 (M057)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1776 (M058)      <xenc:EncryptedData Id="enc"
1777 (M059)      Type="http://www.w3.org/2001/04/xmlenc#Content"
1778 (M060)      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1779 (M061)      <xenc:EncryptionMethod
1780 (M062)      Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
1781 (M063)      <xenc:KeyInfo>
1782 (M064)      <xenc:KeyName>SomeMutuallyAgreedURI</xenc:KeyName>
1783 (M065)      </xenc:KeyInfo>
1784 (M066)      <xenc:CipherData>
1785 (M067)      <xenc:CipherValue>AYb...Y8</xenc:CipherValue>
1786 (M068)      </xenc:CipherData>
1787 (M069)      </xenc:EncryptedData>
1788 (M070)      </soap:Body>
1789 (M071)      </soap:Envelope>

```

1790 In the above example, the main point of attention is line (M064), where the KeyName of the
1791 EncryptedData element is "SomeMutuallyAgreedURI". As indicated above the specific techniques used to
1792 identify and apply this token are totally within agreed upon methods define by the mutual parties.

1793 Lines (M005) – (M055) contain the SOAP Header element, which contains the WS-Security header
1794 (M006) – (M054).

1795 The ReferenceList (M008) – (M010) contains an internal reference (M009), "enc", identifying the
1796 EncryptedData element Id, line (M058). This EncryptedData (M058) – (M069) inside the SOAP Body,
1797 (M056) – (M070), was required to be encrypted by the policy (P058) – (P069) as described above.

1798 The Timestamp (M011) – (M014) is required by the policy line (P034).

1799 The BinarySecurityToken (M015) – (M018) contains the actual certificate used to sign the request as
1800 required by the policy, (P008), IncludeToken/AlwaysToRecipient.

1801 The Signature (M019) – (M053) contains a SignedInfo (M020) – (M046) that identifies the “#body” (M025)
1802 the “#myCert” (M032), and the “#timestamp” (M039) as the elements covered by the signature. The
1803 Reference (M032) – (M038) with URI=”#myCert” that covers the BinarySecurityToken (M015) – (M018)
1804 was required by ProtectTokens in the policy (P035).

1805 The KeyInfo (M048) – (M052) uses a SecurityTokenReference to point to the “myCert”
1806 BinarySecurityToken.

1807 The EncryptedData (M058) – (M069) was described above, where it was pointed out that line (M064)
1808 contains the external reference (“SomeMutuallyAgreedURI”) to the mutually shared symmetric key used
1809 for encryption.

1810 **2.2.3 (WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt**

1811 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”
1812 [\[WS-SECURE-INTEROP\]](#) (see also sec 2.1.4)

1813 In this use case the Request is signed using DerivedKeyToken1(K), then encrypted using a
1814 DerivedKeyToken2(K) where K is ephemeral key protected for the server's certificate. Response is signed
1815 using DKT3(K), (if needed) encrypted using DKT4(K). The requestor does not wish to identify himself; the
1816 message exchange is protected using derived symmetric keys. As a simpler, but less secure, alternative,
1817 ephemeral key K (instead of derived keys) could be used for message protection by simply omitting the
1818 sp:RequireDerivedKeys assertion.

1819 The policy is as follows:

```
1820 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">  
1821 (P002)   <wsp:ExactlyOne>  
1822 (P003)     <wsp:All>  
1823 (P004)       <sp:SymmetricBinding>  
1824 (P005)         <wsp:Policy>  
1825 (P006)           <sp:ProtectionToken>  
1826 (P007)             <wsp:Policy>  
1827 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1828 sx/ws-securitypolicy/200702/IncludeToken/Never">  
1829 (P009)                 <wsp:Policy>  
1830 (P010)                   <sp:RequireDerivedKeys/>  
1831 (P011)                   <sp:RequireThumbprintReference/>  
1832 (P012)                   <sp:WssX509V3Token11/>  
1833 (P013)                 </wsp:Policy>  
1834 (P014)               </sp:X509Token>  
1835 (P015)             </wsp:Policy>  
1836 (P016)           </sp:ProtectionToken>  
1837 (P017)         <sp:AlgorithmSuite>  
1838 (P018)           <wsp:Policy>  
1839 (P019)             <sp:Basic256/>  
1840 (P020)           </wsp:Policy>  
1841 (P021)         </sp:AlgorithmSuite>  
1842 (P022)       <sp:Layout>  
1843 (P023)         <wsp:Policy>  
1844 (P024)           <sp:Strict/>  
1845 (P025)         </wsp:Policy>  
1846 (P026)       </sp:Layout>  
1847 (P027)     <sp:IncludeTimestamp/>  
1848 (P028)     <sp:OnlySignEntireHeadersAndBody/>  
1849 (P029)   </wsp:Policy>  
1850 (P030) </sp:SymmetricBinding>  
1851 (P031) <sp:Wss11>  
1852 (P032)   <wsp:Policy>  
1853 (P033)     <sp:MustSupportRefKeyIdentifier/>  
1854 (P034)     <sp:MustSupportRefIssuerSerial/>  
1855 (P035)     <sp:MustSupportRefThumbprint/>  
1856 (P036)     <sp:MustSupportRefEncryptedKey/>  
1857 (P037)     <sp:RequireSignatureConfirmation/>  
1858 (P038)   </wsp:Policy>
```

```

1859 (P039)      </sp:Wss11>
1860 (P040)      </wsp:All>
1861 (P041)      </wsp:ExactlyOne>
1862 (P042) </wsp:Policy>
1863
1864 (P043) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_input_policy">
1865 (P044)   <wsp:ExactlyOne>
1866 (P045)   <wsp:All>
1867 (P046)   <sp:SignedParts>
1868 (P047)   <sp:Body/>
1869 (P048)   </sp:SignedParts>
1870 (P049)   <sp:EncryptedParts>
1871 (P050)   <sp:Body/>
1872 (P051)   </sp:EncryptedParts>
1873 (P052)   </wsp:All>
1874 (P053)   </wsp:ExactlyOne>
1875 (P054) </wsp:Policy>
1876
1877 (P055) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_output_policy">
1878 (P056)   <wsp:ExactlyOne>
1879 (P057)   <wsp:All>
1880 (P058)   <sp:SignedParts>
1881 (P059)   <sp:Body/>
1882 (P060)   </sp:SignedParts>
1883 (P061)   <sp:EncryptedParts>
1884 (P062)   <sp:Body/>
1885 (P063)   </sp:EncryptedParts>
1886 (P064)   </wsp:All>
1887 (P065)   </wsp:ExactlyOne>
1888 (P066) </wsp:Policy>

```

1889 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token
1890 must be used for both message signature and message encryption.

1891 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)
1892 indicate that the ProtectionToken must be an X.509 token that MUST NOT be included with any message
1893 sent between the Initiator and Recipient.

1894 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an
1895 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the
1896 MustSupportRefThumbprint assertion on line (P035) and RequireThumbprintReference on line (P011), a
1897 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token
1898 is used.

1899 Line (P027) requires the inclusion of a timestamp.

1900 Lines (P031) – (P039) contain some WS-Security 1.1 related interoperability requirements, specifically
1901 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires
1902 signature confirmation on the response.

1903 Lines (P043) – (P054) contain a policy that is attached to the input message. Lines (P045) – (P048)
1904 require that the body of the input message must be signed. Lines (P049) – (P051) require the body of the
1905 input message must be encrypted.

1906 Lines (P055) – (P066) contain a policy that is attached to the output message. Lines (P057) – (P060)
1907 require that the body of the output message must be signed. Lines (P061) – (P063) require the body of
1908 the output message must be encrypted.

1909 An example of an input message that conforms to the above stated policy is as follows.

1910 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,
1911 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the
1912 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-
1913 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```

1914 (M001) <?xml version="1.0" encoding="utf-8" ?>
1915 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">

```



```

1916 (M003) <env:Header>
1917 (M004) <wsse:Security xmlns:wsse="..." xmlns:wssell="..." xmlns:wsc="..."
1918 env:mustUnderstand="1">
1919 (M005) <xenc:EncryptedKey Id="encKey" >
1920 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1921 (M007) <ds:DigestMethod Algorithm...#sha1" />
1922 (M008) </xenc:EncryptionMethod>
1923 (M009) <ds:KeyInfo >
1924 (M010) <wsse:SecurityTokenReference >
1925 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1926 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
1927 (M012) </wsse:SecurityTokenReference>
1928 (M013) </ds:KeyInfo>
1929 (M014) <xenc:CipherData>
1930 (M015) <xenc:CipherValue>TE...=</xenc:CipherValue>
1931 (M016) </xenc:CipherData>
1932 (M017) </xenc:EncryptedKey>
1933 (M018) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey1">
1934 (M019) wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1935 (M020) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1936 (M021) </wsse:SecurityTokenReference>
1937 (M022) <wsc:Generation>0</wsc:Generation>
1938 (M023) <wsc:Length>32</wsc:Length>
1939 (M024) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1940 (M025) <wsc:Nonce>39...=</wsc:Nonce>
1941 (M026) </wsc:DerivedKeyToken>
1942 (M027) <xenc:ReferenceList>
1943 (M028) <xenc:DataReference URI="#encBody"/>
1944 (M029) </xenc:ReferenceList>
1945 (M030) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey2">
1946 (M031) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1947 (M032) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1948 (M033) </wsse:SecurityTokenReference>
1949 (M034) <wsc:Generation>0</wsc:Generation>
1950 (M035) <wsc:Length>32</wsc:Length>
1951 (M036) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1952 (M037) <wsc:Nonce>...=</wsc:Nonce>
1953 (M038) </wsc:DerivedKeyToken>
1954 (M039) <wsu:Timestamp wsu:Id="Timestamp" >
1955 (M040) <wsu:Created>2007-03-26T23:43:13Z</wsu:Created>
1956 (M041) </wsu:Timestamp>
1957 (M042) <ds:Signature >
1958 (M043) <ds:SignedInfo>
1959 (M044) ...
1960 (M045) <ds:Reference URI="#Timestamp">...</ds:Reference>
1961 (M046) <ds:Reference URI="#Body">...</ds:Reference>
1962 (M047) </ds:SignedInfo>
1963 (M048) <ds:SignatureValue>Yu...=</ds:SignatureValue>
1964 (M049) <ds:KeyInfo>
1965 (M050) <wsse:SecurityTokenReference >
1966 (M051) <wsse:Reference URI="#DKey2" ValueType=".../dk"/>
1967 (M052) </wsse:SecurityTokenReference>
1968 (M053) </ds:KeyInfo>
1969 (M054) </ds:Signature>
1970 (M055) </wsse:Security>
1971 (M056) </env:Header>
1972 (M057) <env:Body wsu:Id="Body" >
1973 (M058) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
1974 Encoding="UTF-8" >
1975 (M059) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1976 (M060) <ds:KeyInfo >
1977 (M061) <wsse:SecurityTokenReference >
1978 (M062) <wsse:Reference URI="#DKey1" ValueType=".../dk"/>
1979 (M063) </wsse:SecurityTokenReference>
1980 (M064) </ds:KeyInfo>
1981 (M065) <xenc:CipherData>
1982 (M066) <xenc:CipherValue>p53...f</xenc:CipherValue>
1983 (M067) </xenc:CipherData>
1984 (M068) </xenc:EncryptedData>
1985 (M069) </env:Body>

```

1986 (M070) </env:Envelope>

1987

1988 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1
 1989 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.
 1990 Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is
 1991 used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value
 1992 type of #ThumbprintSHA1 for Thumbprint Reference, it is required by the policy line (P011) of the
 1993 Thumbprint Reference.

1994 Lines (M018) – (M022) is the first wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines
 1995 (M005) – (M017), which is referenced using the WSS1.1 #EncryptedKey SecurityTokenReference
 1996 mechanism contained in lines (M019) – (M021), and used for body encryption and it is referenced on line
 1997 (M062).

1998 Lines (M027) – (M029) is an encryption data reference that references the encrypted body of the
 1999 message on lines (M058) – (M068). The encryption was required by the EncryptedParts assertion of the
 2000 input message policy.

2001 Lines (M030) – (M038) is the second wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines
 2002 (M005) – (M017) and used for signature referenced on line (M051).

2003 Lines (M039) – (M041) contain a timestamp for the message as required by the IncludeTimestamp
 2004 assertion.

2005 Lines (M042) – (M054) contain the message signature.

2006 Line (M045) indicates the message timestamp is included in the signature as required by the
 2007 IncludeTimestamp assertion definition.

2008 Line (M046) indicates the message body is included in the signature as required by the SignedParts
 2009 assertion of the input message policy.

2010 **2.2.4 (WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt**

2011 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”
 2012 [[WS-SECURE-INTEROP](#)] (see also sec 2.1.4)

2013 Client and server X509 certificates are used for client and server authorization respectively. Request is
 2014 signed using K, then encrypted using K, K is ephemeral key protected for server's certificate. Signature
 2015 corresponding to K is signed using client certificate. Response is signed using K, encrypted using K,
 2016 encrypted key K is not included in response. Alternatively, derived keys can be used for each of the
 2017 encryption and signature operations by simply adding an sp:RequireDerivedKeys assertion.

2018 The policy is as follows:

```

2019
2020 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">
2021 (P002)   <wsp:ExactlyOne>
2022 (P003)   <wsp:All>
2023 (P004)     <sp:SymmetricBinding>
2024 (P005)       <wsp:Policy>
2025 (P006)         <sp:ProtectionToken>
2026 (P007)           <wsp:Policy>
2027 (P008)             <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2028 sx/ws-securitypolicy/200702/IncludeToken/Never">
2029 (P009)               <wsp:Policy>
2030 (P010)                 <sp:RequireDerivedKeys/>
2031 (P011)                 <sp:RequireThumbprintReference/>
2032 (P012)                 <sp:WssX509V3Token11/>
2033 (P013)               </wsp:Policy>
2034 (P014)             </sp:X509Token>
2035 (P015)           </wsp:Policy>
2036 (P016)         </sp:ProtectionToken>
2037 (P017)       <sp:AlgorithmSuite>
2038 (P018)     </wsp:Policy>

```

```

2039 (P019)         <sp:Basic256/>
2040 (P020)         </wsp:Policy>
2041 (P021)         </sp:AlgorithmSuite>
2042 (P022)         <sp:Layout>
2043 (P023)         <wsp:Policy>
2044 (P024)         <sp:Strict/>
2045 (P025)         </wsp:Policy>
2046 (P026)         </sp:Layout>
2047 (P027)         <sp:IncludeTimestamp/>
2048 (P028)         <sp:OnlySignEntireHeadersAndBody/>
2049 (P029)         </wsp:Policy>
2050 (P030)         </sp:SymmetricBinding>
2051 (P031)         <sp:EndorsingSupportingTokens>
2052 (P032)         <wsp:Policy>
2053 (P033)         <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2054 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2055 (P034)         <wsp:Policy>
2056 (P035)         <sp:RequireThumbprintReference/>
2057 (P036)         <sp:WssX509V3Token11/>
2058 (P037)         </wsp:Policy>
2059 (P038)         </sp:X509Token>
2060 (P039)         </wsp:Policy>
2061 (P040)         </sp:EndorsingSupportingTokens>
2062 (P041)         <sp:Wss11>
2063 (P042)         <wsp:Policy>
2064 (P043)         <sp:MustSupportRefKeyIdentifier/>
2065 (P044)         <sp:MustSupportRefIssuerSerial/>
2066 (P045)         <sp:MustSupportRefThumbprint/>
2067 (P046)         <sp:MustSupportRefEncryptedKey/>
2068 (P047)         <sp:RequireSignatureConfirmation/>
2069 (P048)         </wsp:Policy>
2070 (P049)         </sp:Wss11>
2071 (P050)         </wsp:All>
2072 (P051)         </wsp:ExactlyOne>
2073 (P052)         </wsp:Policy>
2074
2075 (P053) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_input_policy">
2076 (P054)   <wsp:ExactlyOne>
2077 (P055)   <wsp:All>
2078 (P056)     <sp:SignedParts>
2079 (P057)       <sp:Body/>
2080 (P058)     </sp:SignedParts>
2081 (P059)     <sp:EncryptedParts>
2082 (P060)       <sp:Body/>
2083 (P061)     </sp:EncryptedParts>
2084 (P062)   </wsp:All>
2085 (P063) </wsp:ExactlyOne>
2086 (P064) </wsp:Policy>
2087
2088 (P065) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_output_policy">
2089 (P066)   <wsp:ExactlyOne>
2090 (P067)   <wsp:All>
2091 (P068)     <sp:SignedParts>
2092 (P069)       <sp:Body/>
2093 (P070)     </sp:SignedParts>
2094 (P071)     <sp:EncryptedParts>
2095 (P072)       <sp:Body/>
2096 (P073)     </sp:EncryptedParts>
2097 (P074)   </wsp:All>
2098 (P075) </wsp:ExactlyOne>
2099 (P076) </wsp:Policy>

```

2100 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token
2101 must be used for both message signature and message encryption.

2102 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)
2103 indicate that the initiator token must be an X.509 token that must not be included with all messages sent
2104 between the recipient and Recipient.

2105 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an
2106 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the
2107 MustSupportRefThumbprint assertion on line (P043) and RequireThumbprintReference on line (P011), a
2108 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token
2109 is used.

2110 Line (P027) requires the inclusion of a timestamp.

2111 Lines (P031) – (P040) contain the EndorsingSupportingTokens assertion which indicates that the message
2112 signature should be endorsed by client's X509 certificate on line (P033) – (P038). Line (P033)
2113 IncludeToken=".../AlwaysToRecipient" indicates the endorsing is only required when the message is sent to recipient.
2114 Line (P036) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security
2115 1.1 X.509 Token Profile. and RequireThumbprintReference on line (P035), a Thumbprint Reference of
2116 KeyIdentifier must be used to identify the token in any messages where the token is used.

2117 Lines (P041) – (P049) contain some WS-Security 1.1 related interoperability requirements, specifically
2118 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires
2119 signature confirmation on the response.

2120 Lines (P053) – (P064) contain a policy that is attached to the input message. Lines (P055) – (P058)
2121 require that the body of the input message must be signed. Lines (P059) – (P061) require the body of the
2122 input message must be encrypted.

2123 Lines (P065) – (P076) contain a policy that is attached to the output message. Lines (P067) – (P070)
2124 require that the body of the output message must be signed. Lines (P071) – (P073) require the body of
2125 the output message must be encrypted.

2126 An example of an input message that conforms to the above stated policy is as follows.

2127 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,
2128 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the
2129 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-
2130 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
2131 (M001) <?xml version="1.0" encoding="utf-8" ?>
2132 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wssu="...">
2133 (M003)   <env:Header>
2134 (M004)     <wsse:Security xmlns:wsse="..." xmlns:wssell="..." env:mustUnderstand="1">
2135 (M005)       <xenc:EncryptedKey Id="encKey" >
2136 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
2137 (M007)           <ds:DigestMethod Algorithm="...#sha1" />
2138 (M008)         </xenc:EncryptionMethod>
2139 (M009)         <ds:KeyInfo >
2140 (M010)           <wsse:SecurityTokenReference >
2141 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2142 (M012)               Value="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
2143 (M013)             </wsse:SecurityTokenReference>
2144 (M014)           </ds:KeyInfo>
2145 (M015)           <xenc:CipherData>
2146 (M016)             <xenc:CipherValue>eI...=</xenc:CipherValue>
2147 (M017)           </xenc:CipherData>
2148 (M018)         </xenc:EncryptedKey>
2149 (M019)         <wsse:BinarySecurityToken Value="...#X509v3"
2150 (M020)           EncodingType="...#Base64Binary">MI...=</wsse:BinarySecurityToken>
2151 (M021)         <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
2152 (M022)           wsu:Id="derivedKeyToken1">
2153 (M023)           <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey" >
2154 (M024)             <wsse:Reference Value="...#EncryptedKey" URI="#encKey"/>
2155 (M025)           </wsse:SecurityTokenReference>
2156 (M026)           <wsc:Generation>0</wsc:Generation>
2157 (M027)           <wsc:Length>32</wsc:Length>
2158 (M028)           <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2159 (M029)           <wsc:Nonce>+R...=</wsc:Nonce>
2160 (M030)         </wsc:DerivedKeyToken>
```

```

2161 (M029) <wsu:Timestamp wsu:Id="Timestamp" >
2162 (M030) <wsu:Created>2007-03-31T04:27:21Z</wsu:Created>
2163 (M031) </wsu:Timestamp>
2164 (M032) <n1:ReferenceList xmlns:n1=".../xmlenc#">
2165 (M033) <n1:DataReference URI="#encBody"/>
2166 (M034) </n1:ReferenceList>
2167 (M035) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_shal"
2168 wsu:Id="derivedKeyToken2">
2169 (M036) <wsse:SecurityTokenReference wsse1:TokenType="...#EncryptedKey" >
2170 (M037) <wsse:Reference ValueType="...EncryptedKey" URI="#encKey"/>
2171 (M038) </wsse:SecurityTokenReference>
2172 (M039) <wsc:Generation>0</wsc:Generation>
2173 (M040) <wsc:Length>32</wsc:Length>
2174 (M041) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2175 (M042) <wsc:Nonce>wL...=</wsc:Nonce>
2176 (M043) </wsc:DerivedKeyToken>
2177 (M044) <ds:Signature Id="messageSignature">
2178 (M045) <ds:SignedInfo>
2179 (M046) ...
2180 (M047) <ds:Reference URI="#Timestamp">
2181 (M048) ...
2182 (M049) </ds:Reference>
2183 (M050) <ds:Reference URI="#Body">
2184 (M051) ...
2185 (M052) </ds:Reference>
2186 (M053) </ds:SignedInfo>
2187 (M054) <ds:SignatureValue>abcdefg</ds:SignatureValue>
2188 (M055) <ds:KeyInfo>
2189 (M056) <wsse:SecurityTokenReference >
2190 (M057) <wsse:Reference URI="#derivedKeyToken2" ValueType=".../dk"/>
2191 (M058) </wsse:SecurityTokenReference>
2192 (M059) </ds:KeyInfo>
2193 (M060) </ds:Signature>
2194 (M061) <ds:Signature >
2195 (M062) <ds:SignedInfo>
2196 (M063) ...
2197 (M064) <ds:Reference URI="#messageSignature">
2198 (M065) ...
2199 (M066) </ds:Reference>
2200 (M067) </ds:SignedInfo>
2201 (M068) <ds:SignatureValue>hijklmnop</ds:SignatureValue>
2202 (M069) <ds:KeyInfo>
2203 (M070) <wsse:SecurityTokenReference >
2204 (M071) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2205 ValueType="...#ThumbprintSHA1">Pj...=</wsse:KeyIdentifier>
2206 (M072) </wsse:SecurityTokenReference>
2207 (M073) </ds:KeyInfo>
2208 (M074) </ds:Signature>
2209 (M075) </wsse:Security>
2210 (M076) </env:Header>
2211 (M077) <env:Body wsu:Id="Body" >
2212 (M078) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
2213 Encoding="UTF-8" >
2214 (M079) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
2215 (M080) <ds:KeyInfo >
2216 (M081) <wsse:SecurityTokenReference >
2217 (M082) <wsse:Reference URI="#derivedKeyToken1" ValueType=".../dk"/>
2218 (M083) </wsse:SecurityTokenReference>
2219 (M084) </ds:KeyInfo>
2220 (M085) <xenc:CipherData>
2221 (M086) <xenc:CipherValue>70...=</xenc:CipherValue>
2222 (M087) </xenc:CipherData>
2223 (M088) </xenc:EncryptedData>
2224 (M089) </env:Body>
2225 (M090) </env:Envelope>
2226 (M091)

```

2227
2228 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1
2229 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.

2230 Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token
 2231 from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.
 2232 In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint
 2233 Reference, and it is required by the policy line (P011) of the Thumbprint Reference.

2234 Line (M019) is an X509 BinarySecurityToken that contains the actual X509 certificate that is used by the
 2235 endorsing signature described below. The token is required to be present in the message based on the
 2236 line (P033) that requires this token for the message sent to the recipient.

2237 Lines (M020) – (M027) is the first derived key token. It is derived from the EncryptedKey of lines (M005) –
 2238 (M017) and used for body encryption referenced on line (M081).

2239 Lines (M028) – (M030) contain a Timestamp element for the message as required by the
 2240 IncludeTimestamp assertion.

2241 Lines (M031) – (M033) contain an encryption data reference that references the encrypted body of the
 2242 message on lines (M077) – (M087). The encryption was required by the EncryptedParts assertion of the
 2243 input message policy.

2244 Lines (M034) – (M042) is the second derived key token. It is derived from the EncryptedKey of lines
 2245 (M005) – (M017) and used by the signature that references it on line (M056).

2246 Lines (M043) – (M059) contain the message signature. Lines (M054) – (M058) is its KeyInfo block that
 2247 indicates the second derived key token should be used to verify the message signature.

2248 Line (M057) indicates the message timestamp is included in the signature as required by the
 2249 IncludeTimestamp assertion definition.

2250 Line (M060) indicates the message body is included in the signature as required by the SignedParts
 2251 assertion of the input message policy.

2252 Lines (M060) – (M073) contain the endorsing signature. It signs the message signature, referenced on
 2253 line (M063), per EndorsingSupportingTokens assertion policy requirement on lines (P031) – (P040). Line
 2254 (M070), the KeyIdentifier, has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is
 2255 required by the policy line (P035) of the Thumbprint Reference. This Thumbprint Reference must match
 2256 the Thumbprint of the X509 Certificate contained in the BinarySecurityToken on line (M019), based on the
 2257 IncludeToken requirement line (P033).

2258

2259 An example of an output message that conforms to the above stated policy follows:

2260

2261

2262

2263

2264

2265

2266

2267

2268

2269

2270

2271

2272

2273

2274

2275

2276

2277

2278

2279

2280

2281

2282

2283

2284

2285

2286

2287

```
(R001) <env:Envelope xmlns:env="..." xmlns:wsu="...">
(R002)   <env:Header>
(R003)     <wsse:Security env:mustUnderstand="1" xmlns:wsse="...">
(R004)       <wsu:Timestamp wsu:Id="Timestamp" >
(R005)         <wsu:Created>2007-03-31T04:27:25Z</wsu:Created>
(R006)       </wsu:Timestamp>
(R007)     <wsc:DerivedKeyToken wsu:Id="derivedKeyToken1" xmlns:wsc=".../sc">
(R008)       <wsse:SecurityTokenReference>
(R009)         <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
(R010)           >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
(R011)       </wsse:SecurityTokenReference>
(R012)     <wsc:Offset>0</wsc:Offset>
(R013)     <wsc:Length>24</wsc:Length>
(R014)     <wsc:Nonce>NfSNXZLYAA8mocQz19KWjg==</wsc:Nonce>
(R015)   </wsc:DerivedKeyToken>
(R016) <wsc:DerivedKeyToken wsu:Id="derivedKeyToken2" xmlns:wsc=".../sc">
(R017)   <wsse:SecurityTokenReference>
(R018)     <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
(R019)       >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
(R020)   </wsse:SecurityTokenReference>
(R021)   <wsc:Nonce>1F/CtyQ9d1Ro8E3+uZYmgQ==</wsc:Nonce>
(R022) </wsc:DerivedKeyToken>
(R023) <enc:ReferenceList xmlns:enc=".../xmlenc#">
(R024)   <enc:DataReference URI="#encBody"/>
(R025) </enc:ReferenceList>
(R026) <wsse11:SignatureConfirmation wsu:Id="sigconf1"
(R027)   Value="abcdefg=" xmlns:wsse11="..."/>
```

```

2288 (R028) <wsse1:SignatureConfirmation wsu:Id="sigconf2"
2289 (R029) Value="hijklmnop=" xmlns:wsse1="..."/>
2290 (R030) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
2291 (R031) <SignedInfo>
2292 (R032) <CanonicalizationMethod
2293 (R033) Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2294 (R034) <SignatureMethod
2295 (R035) Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
2296 (R036) <Reference URI="#msgBody">
2297 (R037) ...
2298 (R038) </Reference>
2299 (R039) <Reference URI="#Timestamp">
2300 (R040) ...
2301 (R041) </Reference>
2302 (R042) <Reference URI="#sigconf1">
2303 (R043) ...
2304 (R044) </Reference>
2305 (R045) <Reference URI="#sigconf2">
2306 (R046) ...
2307 (R047) </Reference>
2308 (R048) </SignedInfo>
2309 (R049) <SignatureValue>3rAxsfJ2LjF7liRQX2EH/0DBmzE=</SignatureValue>
2310 (R050) <KeyInfo>
2311 (R051) <wsse:SecurityTokenReference>
2312 (R052) <wsse:Reference URI="#derivedKeyToken1" />
2313 (R053) </wsse:SecurityTokenReference>
2314 (R054) </KeyInfo>
2315 (R055) </Signature>
2316 (R056) </wsse:Security>
2317 (R057) </env:Header>
2318 (R058) <env:Body wsu:Id="msgBody">
2319 (R059) <enc:EncryptedData Id="encBody" Type="...xmlenc#Content"
2320 (R060) xmlns:enc=".../xmlenc#">
2321 (R061) <enc:EncryptionMethod Algorithm=".../xmlenc#aes256-cbc" />
2322 (R062) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
2323 (R063) <wsse:SecurityTokenReference xmlns:wsse="...">
2324 (R064) <wsse:Reference URI="#derivedKeyToken2" />
2325 (R065) </wsse:SecurityTokenReference>
2326 (R066) </KeyInfo>
2327 (R067) <enc:CipherData>
2328 (R068) <enc:CipherValue>y+eV...pu</enc:CipherValue>
2329 (R069) </enc:CipherData>
2330 (R070) </enc:EncryptedData>
2331 (R071) </env:Body>
2332 (R072) </env:Envelope>
2333

```

2334 Lines (R001)-(R072) contain the Response message returned to the Initiator.

2335 Lines (R004)-(R006) contain the Timestamp required by the Policy (P027).

2336 Lines (R007)-(R015) and (R016)-(R022) contain the information necessary for the Initiator to derive the
2337 keys using the WS-SecureConversation shared secret, the identified key (R010) for DK1 and (R019) for
2338 DK2, which reference the same key, and the Nonce (R014) for DK1 and (R021) for DK2, which are
2339 different for the two derived keys.

2340 Lines (R023)-(R025) contain a ReferenceList that indicates the message Body (R058) contains the data
2341 to be encrypted. The encryption was required by the output policy (P079).

2342 Lines (R026)-(R027) contain the SignatureConfirmation for the SignatureValue in the request message
2343 (M054) which was required to be included by the policy (P047) RequireSignatureConfirmation.

2344 Lines (R028)-(R029) contain the SignatureConfirmation for the 2nd SignatureValue in the request
2345 message (M068), which is also required by the policy (P047).

2346 Lines (R030)-(R055) contain the response message signature that covers the Timestamp element
2347 (R039)->(R004) required to be signed by the policy (P027), the two SignatureConfirmation elements
2348 (R042)->(R026) and (R045)->(R028), which are required to be signed because they are required
2349 elements of the policy (P047), and the message Body (R036)->(R058), which is required to be signed by
2350 the output message policy (P076). The signature may be verified using derivedKeyToken1 as indicated
2351 on line (R052).

2352 Lines (R059)-(R070) contain the encrypted data as required by the policy (P079) and may be decrypted
2353 using derivedKeyToken2 as indicated on line (R064).

2354 2.3 SAML Token Authentication Scenario Assertions

2355 For SAML, the combination of SAML and WSS version numbers is supported (WssSamIV11Token10,
2356 WssSamIV11Token11, WssSamIV20Token11).

2357 Instead of explicitly including the SAML Assertion, a wsse:KeyIdentifier reference can also be used. To
2358 enable this last behavior, the IncludeToken attribute is set to [http://docs.oasis-open.org/ws-sx/ws-](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never)
2359 [securitypolicy/200702/IncludeToken/Never](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never).

2360 In all of the SAML scenarios, the SAML Assertion ConfirmationMethod expected to be used by the
2361 Initiator is communicated implicitly by the context of the sp: security binding in use and type of sp:
2362 element containing the SamlToken. There are 3 general SAML ConfirmationMethod use cases covered:
2363 holder-of-key (hk), sender-vouches (sv), and bearer (br).

2364

2365 For the **holder-of-key** case, there is always a contained reference (or value) in the SAML
2366 Assertion to key material that may be used for message protection in the scenario. The hk case
2367 can be assumed if the WssSamIV**Token** element appears either in the sp:InitiatorToken
2368 element of the sp:AsymmetricBinding element, or in the sp:ProtectionToken element of the
2369 sp:SymmetricBinding element. In the sp:TransportBinding case, if the WssSamIV**Token**
2370 element appears in an sp:EndorsingSupportingTokens or sp:SignedEndorsingSupportingTokens
2371 element, the SAML Assertion type can also be assumed to be hk.

2372 The **sender-vouches** case can be assumed if the WssSamIV**Token** version will always
2373 appear in an sp:SignedSupportingTokens element to indicate that the SAML Authority associated
2374 with this token is the Initiator that signs the message.

2375 The **bearer** case can be assumed if the WssSamIV**Token** version appears in an
2376 sp:SupportingTokens element (it may be signed as a SignedPart, but it is not involved in the
2377 message protection).

2378

2379 It is recognized that other uses cases might exist, where these guidelines might need further elaboration
2380 in order to address all contingencies, however that is outside the scope of the current version of this
2381 document.

2382 Note: in the discussions below the term “SamlToken” or “SamlToken assertion” refers to the
2383 policy requirement that a “SAML Assertion” be used as that token.

2384 Note: SAML Assertions have issuers. In addition, these Assertions are generally signed with an
2385 X509 certificate. In general, the SAML IssuingAuthority may be regarded as the Subject of the
2386 X509 certificate, which is trusted by a RelyingParty. In general, as well, there is usually a known
2387 mapping between the Issuer identified within the SAML Assertion and the Subject of the X509
2388 certificate used to sign that Assertion. It will be assumed in this document that the SAML
2389 IssuingAuthority may be identified by either of these identities and that, in general, a RelyingParty
2390 will check the details as necessary.

2391 The concept of the sp:”message signature” within the context of the **sp:TransportBinding** is not clearly
2392 identified within the current version of [[WS-SecurityPolicy](#)], however, there are certain inferences that are
2393 used in the use cases that follow that are identified here for reference.

- 2394 • Based on the diagrams in section 8 of [[WS-SecurityPolicy](#)], which show messages with a
2395 “message signature” followed by a diagram showing use of transport security the only difference
2396 is that the message signature diagrams contain a block labelled “Sig1”, which is the message
2397 signature, and the transport security diagrams do not have this block.
- 2398 • Considering the fact that when [[SSL](#)] Client Certificates are used for client authentication, that the
2399 client uses the client certificate private key to sign hash of SSL handshake messages in an SSL
2400 CertificateVerify message that is part of the SSL protocol, the assumption is made that
2401 subsequent data sent by the client on this link is effectively protected for the purposes of data

- 2402 integrity and confidentiality, and that the data sent on the link is authorized to be sent by the
2403 holder of the private key associated with the client certificate.
- 2404 • Based on the considerations in the bullets above, and with intention of maintaining functional
2405 consistency with the WS-SecurityPolicy notions of sp: message signature,
2406 sp:SignedSupportingTokens, and sp:SignedEndorsingSupportingTokens, use of client certificates
2407 with SSL will be considered equivalent to having a “virtual message signature” provided by the
2408 Initiator’s client certificate which covers all the data in the SOAP request that the Initiator sends
2409 on the SSL link.
 - 2410 • As such, in the above context, the client certificate may be regarded as providing both a virtual
2411 Signing function for tokens and Timestamp that appear in the wsse:Security header, as well as a
2412 virtual Endorsing function for tokens that contain the client certificate or a reference to the client
2413 certificate that appear in the wsse:Security header.

2414 The net effect of the above assumptions for the SAML use cases in this section that use the
2415 **sp:TransportBinding** is that if a **client certificate is required** to be used with the **sp:HttpsToken**
2416 assertion then:

- 2417 1. A SAML **sender-vouches** Assertion contained in a wsse:Security header block may be
2418 considered to be an **sp:SignedSupportingToken** when used in an sp:TransportBinding
2419 using an sp:HttpsToken assertion that contains an sp:RequireClientCertificate assertion,
2420 because the client certificate is being used as the IssuingAuthority behind the SAML
2421 sender-vouches Assertion, which requires the IssuingAuthority to sign the combined
2422 message and Assertion.
- 2423 2. A SAML **holder-of-key** Assertion contained in a wsse:Security header block may be
2424 considered to be an **sp:SignedEndorsingSupportingToken** when used in an
2425 sp:TransportBinding using an sp:HttpsToken assertion that contains an
2426 sp:RequireClientCertificate assertion AND that the either the client certificate or a
2427 reference to it is contained in the saml:SubjectConfirmationData/ds:KeyInfo element of
2428 the SAML holder-of-key Assertion.
- 2429 3. A SAML **bearer** Assertion contained in a wsse:Security header block may only be
2430 considered to be an **sp:SupportingToken**, because they are only “incidentally” covered
2431 by the virtual message signature as a “pass through” token provided by the Requestor to
2432 be evaluated by the Recipient that is being “passed through” by the Initiator, but which
2433 the Initiator takes no responsibility.

2434 Ultimately, the responsibilities associated with the granting access to resources is determined by the
2435 agreements between RelyingParties and IssuingAuthorities. Those agreements need to take into
2436 consideration the security characteristics of the bindings which support access requests as to what kind
2437 of bindings are required to “adequately protect” the requests for the associated business purposes. These
2438 examples are intended to show how SAML Assertions can be used in a variety of WS-SecurityPolicy
2439 contexts and how the different SAML token types (hk, sv, bearer) may be used in different configurations
2440 relating the IssuingAuthority, the Requestor, the Initiator, the Recipient, and the RelyingParty.

2441 **2.3.1 WSS 1.0 SAML Token Scenarios**

2442 **2.3.1.1 (WSS1.0) SAML1.1 Assertion (Bearer)**

2443 Initiator adds a SAML assertion (bearer) representing the Requestor to the SOAP security header.

2444 Since the SamlToken is listed in the SupportingTokens element, it will not explicitly be covered by a
2445 message signature. The Initiator may infer that a Saml Bearer Assertion is acceptable to meet this
2446 requirement, because the Initiator is not required to explicitly cover a SupportingToken with a signature.

2447 The SAML assertion itself could be signed. The IssuingAuthority in this scenario is the Issuer (and signer,
2448 if the Assertion is signed) of the SAML Assertion. The Initiator simply passes the token through and is not
2449 actively involved in the trust relationship between the IssuingAuthority that issued the SAML Assertion
2450 and the Requestor who is the Subject of the SAML Assertion.

2451 This scenario might be used in a SAML Federation application where a browser-based user with a SAML
2452 Assertion indicating that user's SSO (Single Sign On) credential has submitted a request to a portal using
2453 the Assertion as a credential (either directly or indirectly via a SAML Artifact [SAML11]), and the portal as
2454 Initiator is generating a web service request on behalf of this user, but the trust that the Recipient has for
2455 the Requestor is based on the Assertion and its IssuingAuthority, not the Initiator who delivered the
2456 request.

2457

```
2458 (P001) <wsp:Policy>  
2459 (P002)   <sp:SupportingTokens>  
2460 (P003)   <wsp:Policy>  
2461 (P004)     <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
2462           sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
2463 (P005)       <wsp:Policy>  
2464 (P006)         <sp:WssSamlV11Token10/>  
2465 (P007)       </wsp:Policy>  
2466 (P008)     </sp:SamlToken>  
2467 (P009)   </wsp:Policy>  
2468 (P010) </sp:SupportingTokens>  
2469 (P011) </wsp:Policy>
```

2470

2471 Lines (P001)-(P011) contain the endpoint wsp:Policy, which contains no bindings because none is
2472 required to access the service protected by this policy, however the service does require that the
2473 Requestor provide a Supporting Token in the sp:SupportingTokens element (P002)-(P010), which must
2474 be an sp:SamlToken (P004)-(P008), which must be an sp:WssSamlV11Token10 (P006), which means
2475 that the SamlToken must be a SAML 1.1 saml:Assertion that is sent in compliance with the
2476 [WSS10-SAML11-PROFILE].

2477 As explained in section 2.3 above, the fact that the sp:SamlToken is simply in an sp:SupportingTokens
2478 element indicates to the Initiator that a SAML bearer Assertion is what is expected to accompany the
2479 request.

2480 The following is a sample request taken from [WSS11-SAML1120-PROFILE] that complies with the
2481 WSS 1.0 compatible policy above

2482

```
2483 (M001) <S12:Envelope xmlns:S12="...">  
2484 (M002)   <S12:Header>  
2485 (M003)     <wsse:Security xmlns:wsse="...">  
2486 (M004)       <saml:Assertion xmlns:saml="...">  
2487 (M005)         AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"  
2488 (M006)         IssueInstant="2003-04-17T00:46:02Z"  
2489 (M007)         Issuer="www.opensaml.org"  
2490 (M008)         MajorVersion="1"  
2491 (M009)         MinorVersion="1">  
2492 (M010)       <saml:AuthenticationStatement>  
2493 (M011)         <saml:Subject>  
2494 (M012)           <saml:NameIdentifier  
2495 (M013)             NameQualifier="www.example.com"  
2496 (M014)             Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">  
2497 (M015)               uid=joe,ou=people,ou=saml-demo,o=baltimore.com  
2498 (M016)             </saml:NameIdentifier>  
2499 (M017)           <saml:SubjectConfirmation>  
2500 (M018)             <saml:ConfirmationMethod>  
2501 (M019)               urn:oasis:names:tc:SAML:1.0:cm:bearer  
2502 (M020)             </saml:ConfirmationMethod>  
2503 (M021)           </saml:SubjectConfirmation>  
2504 (M022)         </saml:Subject>  
2505 (M023)       </saml:AuthenticationStatement>  
2506 (M024)     </saml:Assertion>  
2507 (M025)   </wsse:Security>  
2508 (M026) </S12:Header>  
2509 (M027) <S12:Body>
```

```
2510 (M028) . . .
2511 (M029) </S12:Body>
2512 (M030) </S12:Envelope>
```

2513

2514 Lines (M001)-(M030) contains the SOAP:Envelope, which contains the SOAP:Header (M002)-(M026)
2515 and the SOAP:Body (M027)-(M029).

2516 The SOAP Header contains a wsse:Security header block (M003)-(M025) which simply contains the
2517 saml:Assertion.

2518 The saml:Assertion (M004)-(M024) is Version 1.1 (M008)-(M009), which is required by the policy
2519 sp:WssSamlV11Token10 assertion (P006). The Assertion contains a saml:AuthenticationStatement
2520 (M010)-(M023) that contains a saml:NameIdentifier (M012)-(M016) that identifies the saml:Subject, who is
2521 the Requestor (who submitted the request from a browser) and it contains a saml:SubjectConfirmation
2522 element (M017)-(M021), which specifies the saml:ConfirmationMethod to be "bearer" (M019), which
2523 meets the policy requirement (P006).

2524 For general context to relate this scenario to Figure 1, the Requestor at a browser will have obtained
2525 either the saml:Assertion above or an Artifact identifying that Assertion from an IssuingAuthority and sent
2526 it to the portal in the context of some request the Requestor is trying to make. The portal recognizes that
2527 to meet the needs of this request that it must invoke a web service that has publicized the policy above
2528 (P001)-(P011). Therefore, the portal will now take on the role of Initiator (Fig 1) and assemble a SOAP
2529 request (M001)-(M030) and submit it to the service as described in detail above.

2530 2.3.1.2 (WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL

2531 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #2](#)
2532 section 4.4.4]

2533 Initiator adds a SAML Assertion (sv) to the SOAP Security Header. Because the TransportBinding
2534 requires a Client Certificate AND the SAML token is in a SignedSupportingTokens element, when the
2535 Initiator uses the Client Certificate to protect the message under SSL, the Initiator may be considered as
2536 effectively "signing" the SAML sv Assertion and acting as a SAML Authority (i.e. the issuer of the sv
2537 Assertion). By including the sv assertion in the header and using the Client Certificate to protect the
2538 message, the Initiator takes responsibility for binding the Requestor, who is the Subject of the Assertion
2539 to the contents of the message.

2540 Note: because SSL does not retain cryptographic protection of the message after the message is
2541 delivered, messages protected using only this mechanism cannot be used as the basis for
2542 non-repudiation.

2543

```
2544 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2545 wsu:Id="Wss10SamlSvV11Tran_policy">
2546 (P002) <sp:TransportBinding>
2547 (P003) <wsp:Policy>
2548 (P004) <sp:TransportToken>
2549 (P005) <wsp:Policy>
2550 (P006) <sp:HttpsToken>
2551 (P007) <wsp:Policy>
2552 (P008) <sp:RequireClientCertificate>
2553 (P009) </wsp:Policy>
2554 (P010) </sp:HttpsToken>
2555 (P011) </wsp:Policy>
2556 (P012) </sp:TransportToken>
2557 (P013) <sp:AlgorithmSuite>
2558 (P014) <wsp:Policy>
2559 (P015) <sp:Basic256 />
2560 (P016) </wsp:Policy>
2561 (P017) </sp:AlgorithmSuite>
2562 (P018) <sp:Layout>
2563 (P019) <wsp:Policy>
2564 (P020) <sp:Strict />
```

```

2565 (P021) </wsp:Policy>
2566 (P022) </sp:Layout>
2567 (P023) <sp:IncludeTimestamp />
2568 (P024) </wsp:Policy>
2569 (P025) </sp:TransportBinding>
2570 (P026) <sp:SignedSupportingTokens>
2571 (P027) <wsp:Policy>
2572 (P028) <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
2573 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2574 (P029) <wsp:Policy>
2575 (P030) <sp:WssSamlV11Token10/>
2576 (P031) </wsp:Policy>
2577 (P032) </sp:SamlToken>
2578 (P033) </wsp:Policy>
2579 (P034) </sp:SignedSupportingTokens>
2580 (P035) </wsp:Policy>

```

2581
2582 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
2583 by a secure transport protocol such as SSL or TLS.

2584 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
2585 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
2586 transport token using the Basic256 algorithm suite (P015).

2587 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of
2588 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section
2589 4.3.1].

2590 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)
2591 since there are no cryptographic tokens required to be present in the WS-Security header.

2592 A timestamp (P023) is required to be included in an acceptable message.

2593 Lines (P026)-(P034) contain a SignedSupportingTokens assertion, which indicates that the referenced
2594 token is effectively "signed" by the combination of usage of the client certificate for SSL authentication
2595 and the cryptographic protection of SSL. However, the "signed" characteristic will not be present after the
2596 message is delivered from the transport layer to the recipient.

2597 Lines (P028)-(P032) indicate the signed token is a SAML Assertion (sender-vouches as described above
2598 in section 2.3) and on line (P030) that it is a SAML 1.1 Assertion and that the WS-Security 1.0 SAML
2599 Profile [[WSS10-SAML11-PROFILE](#)] is being used.

2600 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP](#) "Scenario #2 -
2601 Sender-Vouches: Unsigned: SSL" section 4.4.4]

2602 Here is the example request from that scenario:

```

2603 (M001) <?xml version="1.0" encoding="utf-8" ?>
2604 (M002) <soap:Envelope
2605 (M003)   xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
2606 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2607 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2608 (M006)   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401
2609 (M007)   -wss-wssecurity-secext-1.0.xsd"
2610 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
2611 (M009)   -wssecurity-utility-1.0.xsd">
2612 (M010) <soap:Header>
2613 (M011)   <wss:Security soap:mustUnderstand="1">
2614 (M012)     <wsu:Timestamp>
2615 (M013)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2616 (M014)     </wsu:Timestamp>
2617 (M015)   <saml:Assertion
2618 (M016)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2619 (M017)     MajorVersion="1" MinorVersion="1"
2620 (M018)     AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
2621 (M019)     Issuer=www.opensaml.org

```

```

2622 (M020) IssueInstant="2002-06-19T16:58:33.173Z">
2623 (M021) <saml:Conditions
2624 (M022) NotBefore="2002-06-19T16:53:33.173Z"
2625 (M023) NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2626 (M024) <saml:AuthenticationStatement
2627 (M025) AuthenticationMethod=
2628 (M026) "urn:oasis:names:tc:SAML:1.0:am:password"
2629 (M027) AuthenticationInstant="2002-06-19T16:57:30.000Z">
2630 (M028) <saml:Subject>
2631 (M029) <saml:NameIdentifier
2632 (M030) NameQualifier=www.example.com
2633 (M031) Format="">
2634 (M032) uid=joe,ou=people,ou=saml-demo,o=example.com
2635 (M033) </saml:NameIdentifier>
2636 (M034) <saml:SubjectConfirmation>
2637 (M035) <saml:ConfirmationMethod>
2638 (M036) urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2639 (M037) </saml:ConfirmationMethod>
2640 (M038) </saml:SubjectConfirmation>
2641 (M039) </saml:Subject>
2642 (M040) </saml:AuthenticationStatement>
2643 (M041) </saml:Assertion>
2644 (M042) </wsse:Security>
2645 (M043) </soap:Header>
2646 (M044) <soap:Body>
2647 (M045) <Ping xmlns="http://xmlsoap.org/Ping">
2648 (M046) <text>EchoString</text>
2649 (M047) </Ping>
2650 (M048) </soap:Body>
2651 (M049) </soap:Envelope>

```

2652

2653 Lines (M011)-(M042) contain the WS-Security 1.0 header required by the WssSamIV11Token10
2654 assertion (P030).

2655 Lines (M012)-(M014) contain the wsu:Timestamp required by IncludeTimestamp assertion (P023).

2656 Lines (M015)-(M041) contain the SAML 1.1 Assertion required by the WssSamIV11Token10 assertion
2657 (P030).

2658 Note that the additional requirements identified in the policy above are met by the SSL transport
2659 capabilities and do not appear in any form in the SOAP message (M001)-(M049).

2660 **2.3.1.3 (WSS1.0) SAML1.1 Assertion (HK) over SSL**

2661 Initiator adds a SAML assertion (hk) to the SOAP Security Header. Because the TransportBinding
2662 requires a Client Certificate AND the SAML token is in an EndorsingSupportingTokens element, the
2663 Initiator may be considered to be authorized by the issuer of the hk SAML assertion to bind message
2664 content to the Subject of the assertion. If the Client Certificate matches the certificate identified in the hk
2665 assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor,
2666 who would be the Subject of the hk assertion, to the content of the message.

2667 In this scenario, the IssuingAuthority is the issuer(signer) of the hk SAML Assertion. The Requestor is the
2668 Subject of the Assertion and the Initiator is authorized by the Authority to bind the Assertion to the
2669 message using the ClientCertificate identified in the SAML Assertion, which should also be used to sign
2670 the timestamp of the message with a separate Signature included in the WS-Security header.

2671

```

2672 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2673 (P002) wsu:Id="Wss10SamlHkV11Tran_policy">>
2674 (P003) <sp:TransportBinding>
2675 (P004) <wsp:Policy>
2676 (P005) <sp:TransportToken>
2677 (P006) <wsp:Policy>
2678 (P007) <sp:HttpsToken>

```

```

2679 (P008) <wsp:Policy>
2680 (P009) <sp:RequireClientCertificate>
2681 (P010) </wsp:Policy>
2682 (P011) </sp:HttpsToken>
2683 (P012) </wsp:Policy>
2684 (P013) </sp:TransportToken>
2685 (P014) <sp:AlgorithmSuite>
2686 (P015) <wsp:Policy>
2687 (P016) <sp:Basic256 />
2688 (P017) </wsp:Policy>
2689 (P018) </sp:AlgorithmSuite>
2690 (P019) <sp:Layout>
2691 (P020) <wsp:Policy>
2692 (P021) <sp:Strict />
2693 (P022) </wsp:Policy>
2694 (P023) </sp:Layout>
2695 (P024) <sp:IncludeTimestamp />
2696 (P025) </wsp:Policy>
2697 (P026) </sp:TransportBinding>
2698 (P027) <sp:SignedEndorsingSupportingTokens>
2699 (P028) <wsp:Policy>
2700 (P029) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2701 open.org/ws-sx/ws-
2702 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2703 (P030) <wsp:Policy>
2704 (P031) <sp:WssSamlV11Token10/>
2705 (P032) </wsp:Policy>
2706 (P033) </sp:SamlToken>
2707 (P034) </wsp:Policy>
2708 (P035) </sp:SignedEndorsingSupportingTokens>
2709 (P036) <wsp:Policy>

```

2710

2711 Section 2.3.2.3 contains an example message that is similar to one that could be used for the policy
2712 above, except the Signed Endorsing Supporting Token there is a SAML Version 2.0 token, and a SAML
2713 Version 1.1 token would be required here.

2714 **2.3.1.4 (WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional** 2715 **Encrypt**

2716 This scenario is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #3](#)].
2717 In this case, the SAML token is included as part of the message signature and sent only to the Recipient.
2718 The message security is provided using X.509 certificates with both requestor and service having
2719 exchanged these credentials via some out of band mechanism, which provides the basis of trust of each
2720 others' certificates.

2721 In this scenario the SAML Authority is the Initiator who uses the message signature to both provide the
2722 integrity of the message and to establish the Initiator as the SAML Authority based on the X509 certificate
2723 used to sign the message and the SignedSupportingTokens SAML Assertion. Effectively, the SAML
2724 Assertion is being "issued" as part of the message construction process. The Requestor is the Subject of
2725 the SAML Assertion. The Initiator knows that the Recipient is expecting it to be the SAML Authority by the
2726 fact that the policy specifies that the message requires a SignedSupportingTokens SamlToken.

2727

```

2728 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2729 (P002) wsu:Id="Wss10SamlSvV11Asymm_endpoint_policy">
2730 (P003) <wsp:ExactlyOne>
2731 (P004) <wsp:All>
2732 (P005) <sp:AsymmetricBinding>
2733 (P006) <wsp:Policy>
2734 (P007) <sp:InitiatorToken>
2735 (P008) <wsp:Policy>

```

2736 (P009) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2737 open.org/ws-sx/ws-
2738 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2739 <wsp:Policy>
2740 (P011) <sp:WssX509V3Token10/>
2741 (P012) </wsp:Policy>
2742 (P013) </sp:X509Token>
2743 (P014) </wsp:Policy>
2744 (P015) </sp:InitiatorToken>
2745 (P016) <sp:RecipientToken>
2746 (P017) <wsp:Policy>
2747 (P018) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2748 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
2749 <wsp:Policy>
2750 (P020) <sp:WssX509V3Token10/>
2751 (P021) </wsp:Policy>
2752 (P022) </sp:X509Token>
2753 (P023) </wsp:Policy>
2754 (P024) </sp:RecipientToken>
2755 (P025) <sp:AlgorithmSuite>
2756 (P026) <wsp:Policy>
2757 (P027) <sp:Basic256/>
2758 (P028) </wsp:Policy>
2759 (P029) </sp:AlgorithmSuite>
2760 (P030) <sp:Layout>
2761 (P031) <wsp:Policy>
2762 (P032) <sp:Strict/>
2763 (P033) </wsp:Policy>
2764 (P034) </sp:Layout>
2765 (P035) <sp:IncludeTimestamp/>
2766 (P036) <sp:OnlySignEntireHeadersAndBody/>
2767 (P037) </wsp:Policy>
2768 (P038) </sp:AsymmetricBinding>
2769 (P039) <sp:Wss10>
2770 (P040) <wsp:Policy>
2771 (P041) <sp:MustSupportRefKeyIdentifier/>
2772 (P042) </wsp:Policy>
2773 (P043) </sp:Wss10>
2774 (P044) <sp:SignedSupportingTokens>
2775 (P045) <wsp:Policy>
2776 (P046) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2777 open.org/ws-sx/ws-
2778 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2779 <wsp:Policy>
2780 (P048) <sp:WssSamlV11Token10/>
2781 (P049) </wsp:Policy>
2782 (P050) </sp:SamlToken>
2783 (P051) </wsp:Policy>
2784 (P052) </sp:SignedSupportingTokens>
2785 (P053) </wsp:All>
2786 (P054) </wsp:ExactlyOne>
2787 (P055) </wsp:Policy>
2788 (P056) <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_input_policy">
2789 (P057) <wsp:ExactlyOne>
2790 (P058) <wsp:All>
2791 (P059) <sp:SignedParts>
2792 (P060) <sp:Body/>
2793 (P061) </sp:SignedParts>
2794 (P062) <sp:EncryptedParts wsp:Optional="true">
2795 (P063) <sp:Body/>
2796 (P064) </sp:EncryptedParts>
2797 (P065) </wsp:All>
2798 (P066) </wsp:ExactlyOne>
2799 (P067)

```

2800      (P068)    </wsp:Policy>
2801
2802      (P069)    <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_output_policy">
2803      (P070)      <wsp:ExactlyOne>
2804      (P071)        <wsp:All>
2805      (P072)          <sp:SignedParts>
2806      (P073)            <sp:Body/>
2807      (P074)          </sp:SignedParts>
2808      (P075)          <sp:EncryptedParts>
2809      (P076)            <sp:Body/>
2810      (P077)          </sp:EncryptedParts>
2811      (P078)        </wsp:All>
2812      (P079)      </wsp:ExactlyOne>
2813      (P080)    </wsp:Policy>

```

2814 Line (P004) is a `wsp:All` element whose scope carries down to line (P053), which means all 3 of the
2815 contained assertions: (P005)-(P038) `AsymmetricBinding`, (P039)-(P043) `WSS10`, and (P044)-(P052)
2816 `SignedSupportingTokens` must be complied with by message exchanges to a Web Service covered by
2817 this policy.

2818 Lines (P005)-(P038) contain an `AsymmetricBinding` assertion which indicates that the Initiator's token
2819 must be used for the message signature and that the recipient's token must be used for message
2820 encryption.

2821 Lines (P007)-(P015) contain the `InitiatorToken` assertion. Within the `InitiatorToken` assertion, lines
2822 (P009)-(P013) indicate that the Initiator's token must be an `X509Token` that must always be included as
2823 part of the request message (as opposed to an external reference). Line (P011) indicates that the `X509`
2824 token must be an `X.509 V3` signature-verification certificate and used in the manner described in
2825 [\[WSS10-X509-PROFILE\]](#).

2826 Lines (P016)-(P023) contain the `RecipientToken` assertion. Again, this an `X.509 V3` certificate (P020) that
2827 must be used as described in [\[WSS10-X509-PROFILE\]](#), however the token itself, must never be included
2828 in messages in either direction (P018).

2829 Instead (momentarily skipping slightly ahead), policy lines (P039)-(P043), the `WSS10` assertion, indicate
2830 that the Recipient's token must be referenced by a `KeyIdentifier` element, which is described in the `WS-`
2831 `Security 1.0` core specification [\[WSS10-SOAPMSG\]](#).

2832 Lines (P025)-(P029) contain the `AlgorithmSuite` assertion, which specifies the `sp:Basic256` set of
2833 cryptographic components. The relevant values for this example within the `sp:Basic256` components are
2834 the asymmetric signing algorithm, `ds:rsa-sha1`, and the digest algorithm, `ds:sha1`, where "ds:" =
2835 "http://www.w3.org/2000/09/xmlsig#" [\[XML-DSIG\]](#).

2836 Lines (P030)-(P034) contain the `sp:Layout` assertion, which is set to `sp:Strict` (P032), which governs the
2837 required relative layout of various `Timestamp`, `Signature` and `Token` elements in the messages.

2838 Line (P035) `IncludeTimestamp` means a `Timestamp` element must be included in the `WS-Security` header
2839 block and signed by the message signature for messages in both directions.

2840 Line (P036) `OnlySignEntireHeaderAndBody` indicates that the message signature must explicitly cover
2841 the `SOAP:Body` element (not children), and if there are any signed elements in the `SOAP:Header` they
2842 must be direct child elements of the `SOAP:Header`. However, the one exception where the latter condition
2843 is not applied is that if the child of the `SOAP:Header` is a `WS-Security` header (i.e. a `wsse:Security`
2844 element), then individual direct child only elements of that `Security` element may also be signed.

2845 Lines (P044)-(P052) contain a `SignedSupportingTokens` assertion, which contains only one token
2846 (P046)-(P050), a `SamlToken`, which must always be sent to the Recipient (P046) and it must be a
2847 `SAML 1.1 Assertion` token. Because the `SamlToken` is in a "SignedSupportingTokens" element, it is
2848 implicitly a `SAML sender-vouches ConfirmationMethod` token as described in section 2.3 above.

2849 Lines (P057)-(P068) contain a policy that applies only to input messages from the Initiator to the
2850 Recipient.

2851 Lines (P060)-(P062) specify that the `SOAP:Body` element of the input message must be signed by the
2852 Initiator's message signature.

2853 Lines (P063)-(P065) specify that the SOAP:Body element of the input message may optionally be
2854 encrypted (signified by `wsp:Optional="true"`) using the Recipient's encryption token (in this case (P020), it
2855 is an X.509 certificate).

2856 Lines (P069)-(P080) contain a policy that applies only to output messages from the Recipient to the
2857 Initiator.

2858 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be signed by the
2859 Recipient's message signature.

2860 Lines (P075)-(P077) specify that the SOAP:Body element of the output message must be encrypted by
2861 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

2862 Here is an example request:

2863

```
2864 (M001) <?xml version="1.0" encoding="utf-8" ?>
2865 (M002) <S12:Envelope
2866 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
2867 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2868 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2869 (M006)   xmlns:wssse="http://docs.oasis-open.org/wss/2004/01/oasis-
2870 (M007) 200401-wss-wssecurity-secext-1.0.xsd"
2871 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
2872 (M009) 200401-wss-wssecurity-utility-1.0.xsd"
2873 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#
2874 (M011)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
2875 (M012) <S12:Header>
2876 (M013) <wssse:Security S12:mustUnderstand="1">
2877 (M014) <wsu:Timestamp wsu:Id="timestamp">
2878 (M015)   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2879 (M016) </wsu:Timestamp>
2880 (M017) <saml:Assertion
2881 (M018)   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2882 (M019)   IssueInstant="2003-04-17T00:46:02Z"
2883 (M020)   Issuer=www.opensaml.org
2884 (M021)   MajorVersion="1"
2885 (M022)   MinorVersion="1">
2886 (M023) <saml:Conditions
2887 (M024)   NotBefore="2002-06-19T16:53:33.173Z"
2888 (M025)   NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2889 (M026) <saml:AttributeStatement>
2890 (M027) <saml:Subject>
2891 (M028) <saml:NameIdentifier
2892 (M029)   NameQualifier=www.example.com
2893 (M030)   Format="">
2894 (M031)   uid=joe,ou=people,ou=saml-demo,o=example.com
2895 (M032) </saml:NameIdentifier>
2896 (M033) <saml:SubjectConfirmation>
2897 (M034) <saml:ConfirmationMethod>
2898 (M035)   urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2899 (M036) </saml:ConfirmationMethod>
2900 (M037) </saml:SubjectConfirmation>
2901 (M038) </saml:Subject>
2902 (M039) <saml:Attribute>
2903 (M040)   ...
2904 (M041) </saml:Attribute>
2905 (M042)   ...
2906 (M043) </saml:AttributeStatement>
2907 (M044) </saml:Assertion>
2908 (M045) <wssse:SecurityTokenReference wsu:id="STR1">
2909 (M046) <wssse:KeyIdentifier wsu:id="..."
2910 (M047)   ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
2911 2004XXwss-saml-token-profile-1.0#SAMLAssertionID">
2912 (M048)   _a75adf55-01d7-40cc-929f-dbd8372ebdfc
2913 (M049) </wssse:KeyIdentifier>
```

```

2914 (M050) </wsse:SecurityTokenReference>
2915 (M051) <wsse:BinarySecurityToken
2916 (M052)   wsu:Id="attesterCert"
2917 (M053)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2918 (M054) 200401-wss-x509-token-profile-1.0#X509v3"
2919 (M055)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
2920 (M056) 200401-wss-soap-message-security-1.0#Base64Binary">
2921 (M057)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
2922 (M058) </wsse:BinarySecurityToken>
2923 (M059) <ds:Signature>
2924 (M060)   <ds:SignedInfo>
2925 (M061)     <ds:CanonicalizationMethod
2926 (M062)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2927 (M063)     <ds:SignatureMethod
2928 (M064)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2929 (M065)     <ds:Reference URI="#STR1">
2930 (M066)       <ds:Transforms>
2931 (M067)         <ds:Transform
2932 (M068)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
2933 (M069) 200401-wss-soap-message-security-1.0#STR-Transform">
2934 (M070)           <wsse:TransformationParameters>
2935 (M071)             <ds:CanonicalizationMethod
2936 (M072)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2937 (M073)             </wsse:TransformationParameters>
2938 (M074)           </ds:Transform>
2939 (M075)         </ds:Transforms>
2940 (M076)       <ds:DigestMethod
2941 (M077)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2942 (M078)       <ds:DigestValue>...</ds:DigestValue>
2943 (M079)     </ds:Reference>
2944 (M080)     <ds:Reference URI="#MsgBody">
2945 (M081)       <ds:DigestMethod
2946 (M082)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2947 (M083)       <ds:DigestValue>...</ds:DigestValue>
2948 (M084)     </ds:Reference>
2949 (M085)     <ds:Reference URI="#timestamp">
2950 (M086)       <ds:DigestMethod
2951 (M087)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2952 (M088)       <ds:DigestValue>...</ds:DigestValue>
2953 (M089)     </ds:Reference>
2954 (M090)   </ds:SignedInfo>
2955 (M091)   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
2956 (M092)   <ds:KeyInfo>
2957 (M093)     <wsse:SecurityTokenReference wsu:id="STR2">
2958 (M094)       <wsse:Reference wsu:id="..." URI="#attesterCert" />
2959 (M095)     </wsse:SecurityTokenReference>
2960 (M096)   </ds:KeyInfo>
2961 (M097)   </ds:Signature>
2962 (M098) </wsse:Security>
2963 (M099) </S12:Header>
2964 (M100) <S12:Body wsu:Id="MsgBody">
2965 (M101)   <ReportRequest>
2966 (M102)     <TickerSymbol>SUNW</TickerSymbol>
2967 (M103)   </ReportRequest>
2968 (M104) </S12:Body>
2969 (M105) </S12:Envelope>

```

2970

2971 **Note:** For the instructional purposes of this document, in order to be compliant with WS-
2972 SecurityPolicy, this copy of the original message had to be modified from the original. In particular,
2973 the modifications that are applied above are the inclusion of a wsu:Id attribute in the Timestamp
2974 element start tag (M014) and the addition of a ds:Reference element and URI attribute identifying
2975 that Timestamp element in the message signature (M085)-(M089). (The original message in

2976 [WSS10-SAML11-INTEROP scenario #3] is not compliant with WS-SecurityPolicy because the
2977 Timestamp that it contains is not covered by the message signature in that document.)

2978 Lines (M002)-(M0105) contain the SOAP:Envelope, i.e. the whole SOAP message.

2979 Lines (M012)-(M099) contain the SOAP:Header, which is the header portion of the SOAP message.

2980 Lines (M0100)-(M0104) contain the SOAP:Body, which is just some dummy content for test purposes.

2981 Lines (M013)-(M098) contain the wsse:Security header, which is the primary focus of this example.

2982 Lines (M014)-(M016) contain the wsu:Timestamp, which is required by the policy (P035).

2983 Lines (M017)-(M044) contain the SAML Assertion, which is the sp:SamlToken required by the policy
2984 sp:SignedSupportingTokens (P046)-(P050).

2985 Lines (M021)-(M022) identify the SAML Assertion as being Version 1.1 as required by the policy (P048).

2986 Line (M035) identifies the SAML Assertion as having ConfirmationMethod sender-vouches, which is the
2987 implicit requirement of the sp:SamlToken being in the SignedSupportingTokens as described above in the
2988 policy section covering (P044)-(P052).

2989 Lines (M045)-(M050) contain a wsse:SecurityTokenReference which contains a wsse:KeyIdentifier, which
2990 is used to reference the SAML Assertion, as described in [WSS10-SAML11-PROFILE] and required by
2991 the policy (P041).

2992 Lines (M051)-(M058) contain the Initiator's wsse:BinarySecurityToken, which is required by the policy
2993 (P007)-(P015), and in particular lines (M053)-(M054) identify the token as an X.509 V3 token as required
2994 by the policy (P011). Line (M057) contains a truncated portion of the Base64 representation of the actual
2995 certificate, which is included in the message as required by the policy sp:IncludeToken (P009).

2996 Lines (M059)-(M0102) contain the ds:Signature, which is the sp: message signature as required by
2997 various parts of the Endpoint Policy (P001)-(P055) and Input Message Policy (P057)-(P068).

2998 Lines (M060)-(M095) contain the ds:SignedInfo element which describes the signing algorithm used
2999 (M064) and specific elements that are signed (M065)-(M094) as required by the policies, which are
3000 described in detail immediately following.

3001 Lines (M061)-(M062) contain the ds:CanonicalizationMethod Algorithm, which is specified as xml-exc-
3002 c14n#, which is the default value required by the policy sp:AlgorithmSuite (P025)-(P029).

3003 Line (M064) identifies the SignatureMethod Algorithm as ds:rsa-sha1, which is the asymmetric key
3004 signing algorithm required by the policy sp:AlgorithmSuite (P025)-(P029).

3005 Lines (M065)-(M079) identify the SAML Assertion (M017)-(M044) as an element that is signed, which is
3006 referenced through the URI "#STR1" on line (M065), which references the SecurityTokenReference
3007 (M045)-(M050), which in turn uses the identifier on line (M048) to reference the actual SAML Assertion by
3008 its AssertionID attribute on line (M018). The SAML Assertion is required to be signed because it is
3009 identified in the policy within the SignedSupportingTokens element on line (P048).

3010 Lines (M077)-(M078) contain the digest algorithm, which is specified to be sha1, as required by the policy
3011 sp:Basic256 assertion (P027) in the sp:AlgorithmSuite.

3012 Lines (M080)-(M084) identify the SOAP:Body (M0100)-(M0104) as an element that is signed, which is
3013 referenced through the URI "#MsgBody" on line (M080). The SOAP Body is required to be signed by the
3014 input message policy lines (P060)-(P062).

3015 Lines (M085)-(M089) identify the wsu:Timestamp (M014)-(M016) as an element that is signed, which is
3016 referenced through the URI "#timestamp" on line (M085). The wsu:Timestamp is required to be signed by
3017 the policy sp:IncludeTimestamp assertion (P035). Note that the wsu:Timestamp occurs in the
3018 wsse:Security header prior to the ds:Signature as required by the sp:Strict layout policy (P032).

3019 Finally, lines (M092)-(M096) contain the ds:KeyInfo element that identifies the signing key associated with
3020 this ds:Signature element. The actual signing key is referenced through the
3021 wsse:SecurityTokenReference (M093)-(M095), with URI "#attesterCert", which identifies the
3022 InitiatorToken identified by the policy (P011) and contained in the wsse:BinarySecurityToken element
3023 (M051)-(M058), which occurs in the wsse:Security header prior to this ds:Signature element, as required
3024 by the sp:Strict layout policy assertion (P032). (Note: this BinarySecurityToken would need to be included

3025 in the signature, if the sp:AsymmetricBinding assertion in the endpoint policy contained a
3026 sp:ProtectTokens assertion, but it does not, so it does not need to be signed.)

3027 **2.3.1.5 (WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt**

3028 This example is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #4](#)].

3029 In this example the SAML token provides the key material for message security, hence acts as the
3030 Initiator token, and therefore the Requestor and Initiator may be considered to be the same entity. The
3031 SAML HK Assertion contains a reference to the public key of the signer of the message (the Initiator). The
3032 Initiator knows Recipient's public key, which it may use to encrypt the request, but the Initiator does not
3033 share a direct trust relation with the Recipient except indirectly through the SAML Assertion Issuer. The
3034 Recipient has a trust relation with the Authority that issues the SAML HK Assertion. The indirect trust
3035 relation between the Recipient and the Initiator is established by the fact that the Initiator's public key has
3036 been signed by the Authority in the SAML HK Assertion. On the request the message body is signed
3037 using Initiator's private key referenced in the SAML HK Assertion and it is optionally encrypted using
3038 Recipient's server certificate. On the response, the server signs the message using its private key and
3039 encrypts the message using the key provided within SAML HK Assertion.

3040 HK Note: there is a trust model aspect to the WS-Security holder-of-key examples that
3041 implementors may want to be aware of. The [[SAML20-CORE](#)] specification defines the Subject of
3042 the SAML Assertion such that the "presenter" of the Assertion is the entity that "attests" to the
3043 information in the Assertion. "The attesting entity and the actual Subject **may or may not** be the
3044 same entity." In general, these two choices map to the actors in [Figure 1](#) as follows: the
3045 "attesting" entity may be regarded as the Initiator. The Subject entity, about whom the information
3046 in the Assertion applies, may be regarded as the Requestor. In the case where the Subject and
3047 attestor are one and the same, the Initiator and Requestor may be regarded as one and the
3048 same. In this case the potential exists to use the Assertion for non-repudiation purposes with
3049 respect to messages that the Requestor/Initiator signs. When the Subject and attestor are
3050 separate entities, then holder-of-key is more similar to sender-vouches where the Initiator/attestor
3051 is sending the message on behalf of the Subject. The mechanisms for determining whether the
3052 Subject and attestor may be verified to be the same entity are dependent on the arrangements
3053 among the business entities and the structure of the associated application scenarios.

3054

```
3055 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
3056 (P002)   wsu:Id="Wss10SamlHkV11Asymm_endpoint_policy">
3057 (P003)   <wsp:ExactlyOne>
3058 (P004)     <wsp>All>
3059 (P005)       <sp:AsymmetricBinding>
3060 (P006)         <wsp:Policy>
3061 (P007)           <sp:InitiatorToken>
3062 (P008)             <wsp:Policy>
3063 (P009)               <sp:SamlToken sp:IncludeToken="http://docs.oasis-
3064 open.org/ws-sx/ws-
3065 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3066 (P010)                 <wsp:Policy>
3067 (P011)                   <wsp:WssSamlV11Token10/>
3068 (P012)                 </wsp:Policy>
3069 (P013)               </sp:SamlToken>
3070 (P014)             </wsp:Policy>
3071 (P015)           </sp:InitiatorToken>
3072 (P016)         <sp:RecipientToken>
3073 (P017)           <wsp:Policy>
3074 (P018)             <sp:X509Token sp:IncludeToken="http://docs.oasis-
3075 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
3076 (P019)               <wsp:Policy>
3077 (P020)                 <sp:WssX509V3Token10/>
3078 (P021)               </wsp:Policy>
3079 (P022)             </sp:X509Token>
3080 (P023)           </wsp:Policy>
3081 (P024)         </sp:RecipientToken>
```

```

3082 (P025) <sp:AlgorithmSuite>
3083 (P026) <wsp:Policy>
3084 (P027) <sp:Basic256/>
3085 (P028) </wsp:Policy>
3086 (P029) </sp:AlgorithmSuite>
3087 (P030) <sp:Layout>
3088 (P031) <wsp:Policy>
3089 (P032) <sp:Strict/>
3090 (P033) </wsp:Policy>
3091 (P034) </sp:Layout>
3092 (P035) <sp:IncludeTimestamp/>
3093 (P036) <sp:OnlySignEntireHeadersAndBody/>
3094 (P037) </wsp:Policy>
3095 (P038) </sp:AsymmetricBinding>
3096 (P039) <sp:Wss10>
3097 (P040) <wsp:Policy>
3098 (P041) <sp:MustSupportRefKeyIdentifier/>
3099 (P042) </wsp:Policy>
3100 (P043) </sp:Wss10>
3101 (P044) </wsp:All>
3102 (P045) </wsp:ExactlyOne>
3103 (P046) </wsp:Policy>
3104 (P047)
3105 (P048) <wsp:Policy wsu:Id="WSS10SamlHok_input_policy">
3106 (P049) <wsp:ExactlyOne>
3107 (P050) <wsp:All>
3108 (P051) <sp:SignedParts>
3109 (P052) <sp:Body/>
3110 (P053) </sp:SignedParts>
3111 (P054) <wsp:ExactlyOne>
3112 (P055) <wsp:All>
3113 (P056) <sp:EncryptedParts>
3114 (P057) <sp:Body/>
3115 (P058) </sp:EncryptedParts>
3116 (P059) </wsp:All>
3117 (P060) <wsp:All/>
3118 (P061) </wsp:ExactlyOne>
3119 (P062) </wsp:All>
3120 (P063) </wsp:ExactlyOne>
3121 (P064) </wsp:Policy>
3122 (P065)
3123 (P066) <wsp:Policy wsu:Id="WSS10SamlHok_output_policy">
3124 (P067) <wsp:ExactlyOne>
3125 (P068) <wsp:All>
3126 (P069) <sp:SignedParts>
3127 (P070) <sp:Body/>
3128 (P071) </sp:SignedParts>
3129 (P072) <sp:EncryptedParts>
3130 (P073) <sp:Body/>
3131 (P074) </sp:EncryptedParts>
3132 (P075) </wsp:All>
3133 (P076) </wsp:ExactlyOne>
3134 (P077) </wsp:Policy>

```

3135

3136 Lines (P001)-(P046) contain the endpoint policy, and lines (P048)-(P064) and (P066)-(P077) contain the
3137 message input and message output policies, respectively.

3138 Lines (P005)-(P038) contain the AsymmetricBinding assertion, which requires separate security tokens
3139 for the Initiator and Recipient.

3140 Lines (P007)-(P015) contain the InitiatorToken, which is defined to be a SamlToken (P009)-(P013), which
3141 must always be sent to the Recipient (P009). The SamlToken is further specified by the
3142 WssSamIV11Token10 assertion (P011) that requires the token to be a SAML 1.1 Assertion and the token
3143 must be used in accordance with the WS-Security [[WSS10-SAML11-PROFILE](#)]. Furthermore, as

3144 described in section 2.3 above, because the SamlToken assertion appears within an InitiatorToken
3145 assertion, the SAML Assertion must use the holder-of-key ConfirmationMethod, whereby for the indicated
3146 WS-Security profile, the SAML Assertion contains a reference to the Initiator's X.509 signing certificate or
3147 equivalent.

3148 Lines (P016)-(P024) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that
3149 must be used as described in [WSS10-SAML11-PROFILE], however the token itself, must never be
3150 included in messages in either direction (P018) (i.e not only will the Recipient not send the token, but the
3151 Initiator must not send the actual token, even when using it for encryption).

3152 Lines (P025)-(P029) AlgorithmSuite and (P030)-(P034) Layout are the same as described above in
3153 section 2.3.1.4.

3154 Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header
3155 block and signed by the message signature for messages in both directions.

3156 Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover
3157 the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they
3158 must be direct child elements of the SOAP:Header. However, the one exception where the latter condition
3159 is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security
3160 element), then individual direct child only elements of that Security element may also be signed.

3161

3162 Lines (P039)-(P043) contain the WSS10 assertion, which indicates that the Recipient's token must be
3163 referenced by a KeyIdentifier element (P041), the usage of which is described in the WS-Security 1.0
3164 core specification [[WSS10-SOAPMSG](#)].

3165 Lines (P048)-(P064) contain the message input policy that applies only to messages from the Initiator to
3166 the Recipient.

3167 Lines (P051)-(P053) specify that the SOAP:Body element of the input message must be signed by the
3168 Initiator's message signature.

3169 Lines (P054)-(P061) specify that the SOAP:Body element of the input message may optionally (signified
3170 by the empty policy alternative on line (P060)) be encrypted using the Recipient's encryption token (in this
3171 case (P020), it is an X.509 certificate).

3172 Note: Because the input policy above (P048-P064) has the EncryptedParts assertion
3173 (P056-P058) contained in an <ExactlyOne> element (P054-P061), which also contains an empty
3174 policy element (P060), either a message with an encrypted Body element or an unencrypted
3175 Body element will be accepted.

3176 Lines (P066)-(P077) contain a message output policy that applies only to messages from the Recipient to
3177 the Initiator.

3178 Lines (P069)-(P071) specify that the SOAP:Body element of the output message must be signed by the
3179 Recipient's message signature.

3180 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be encrypted by
3181 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

3182

3183 The following example request is taken from the [[WSS10-SAML11-INTEROP](#)] document scenario #4:

3184

```
3185 (M001) <?xml version="1.0" encoding="utf-8" ?>  
3186 (M002) <S12:Envelope  
3187 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/  
3188 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance  
3189 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema  
3190 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401  
3191 (M007)   wss-wssecurity-secext-1.0.xsd"  
3192 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss  
3193 (M009)   wssecurity-utility-1.0.xsd"  
3194 (M010)   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
3195 (M011)   <S12:Header>
```

```

3196 (M012) <wsse:Security S12:mustUnderstand="1">
3197 (M013)   <wsu:Timestamp wsu:Id="timestamp">
3198 (M014)     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
3199 (M015)   </wsu:Timestamp>
3200 (M016)   <saml:Assertion
3201 (M017)     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3202 (M018)     IssueInstant="2003-04-17T00:46:02Z"
3203 (M019)     Issuer=www.opensaml.org
3204 (M020)     MajorVersion="1"
3205 (M021)     MinorVersion="1"
3206 (M022)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
3207 (M023)     <saml:Conditions
3208 (M024)       NotBefore="2002-06-19T16:53:33.173Z"
3209 (M025)       NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
3210 (M026)     <saml:AttributeStatement>
3211 (M027)       <saml:Subject>
3212 (M028)         <saml:NameIdentifier
3213 (M029)           NameQualifier=www.example.com
3214 (M030)           Format="">
3215 (M031)           uid=joe,ou=people,ou=saml-demo,o=example.com
3216 (M032)         </saml:NameIdentifier>
3217 (M033)       <saml:SubjectConfirmation>
3218 (M034)         <saml:ConfirmationMethod>
3219 (M035)           urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
3220 (M036)         </saml:ConfirmationMethod>
3221 (M037)         <ds:KeyInfo>
3222 (M038)           <ds:KeyValue>...</ds:KeyValue>
3223 (M039)         </ds:KeyInfo>
3224 (M040)       </saml:SubjectConfirmation>
3225 (M041)     </saml:Subject>
3226 (M042)     <saml:Attribute
3227 (M043)       AttributeName="MemberLevel"
3228 (M044)       AttributeNamespace=
3229 (M045)         "http://www.oasis-open.org/Catalyst2002/attributes">
3230 (M046)       <saml:AttributeValue>gold</saml:AttributeValue>
3231 (M047)     </saml:Attribute>
3232 (M048)     <saml:Attribute
3233 (M049)       AttributeName="E-mail"
3234 (M050)       AttributeNamespace="http://www.oasis-
3235 (M051)         open.org/Catalyst2002/attributes">
3236 (M052)       <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
3237 (M053)     </saml:Attribute>
3238 (M054)   </saml:AttributeStatement>
3239 (M055)   <ds:Signature>...</ds:Signature>
3240 (M056) </saml:Assertion>
3241 (M057) <ds:Signature>
3242 (M058)   <ds:SignedInfo>
3243 (M059)     <ds:CanonicalizationMethod
3244 (M060)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
3245 (M061)     <ds:SignatureMethod
3246 (M062)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3247 (M063)     <ds:Reference URI="#MsgBody">
3248 (M064)       <ds:DigestMethod
3249 (M065)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3250 (M066)       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
3251 (M067)     </ds:Reference>
3252 (M068)     <ds:Reference URI="#timestamp">
3253 (M069)       <ds:DigestMethod
3254 (M070)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3255 (M071)       <ds:DigestValue>...</ds:DigestValue>
3256 (M072)     </ds:Reference>
3257 (M073)   </ds:SignedInfo>
3258 (M074)   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
3259 (M075)   <ds:KeyInfo>

```

```

3260 (M076) <wsse:SecurityTokenReference wsu:id="STR1">
3261 (M077) <wsse:KeyIdentifier wsu:id="..."
3262 (M078) Value="http://docs.oasis-open.org/wss/2004/XX/oasis-
3263 (M079) 2004XX-wss-saml-token-profile-1.0#SAMLAssertionID">
3264 (M080) _a75adf55-01d7-40cc-929f-dbd8372ebdfc
3265 (M081) </wsse:KeyIdentifier>
3266 (M082) </wsse:SecurityTokenReference>
3267 (M083) </ds:KeyInfo>
3268 (M084) </ds:Signature>
3269 (M085) </wsse:Security>
3270 (M086) </S12:Header>
3271 (M087) <S12:Body wsu:Id="MsgBody">
3272 (M088) <ReportRequest>
3273 (M089) <TickerSymbol>SUNW</TickerSymbol>
3274 (M090) </ReportRequest>
3275 (M091) </S12:Body>
3276 (M092) </S12:Envelope>

```

3277

3278 Note: for the instructional purposes of this document, it was necessary to make changes to the
3279 original sample request to meet the requirements of WS-SecurityPolicy. The changes to the
3280 message above include:

- 3281 • Line (M013): added wsu:Id="timestamp" attribute to sp:Timestamp element.
- 3282 • Lines (M068)-(M072): added a ds:Reference element to include the Timestamp in the
3283 message signature required by the policy sp:IncludeTimestamp assertion (P035).

3284 Lines (M002)-(M092) contain the SOAP Envelope, i.e. the whole SOAP message.

3285 Lines (M011)-(M086) contain the SOAP Header.

3286 Lines (M087)-(M091) contain the unencrypted SOAP Body, which is allowed to be unencrypted based on
3287 line (P060) of the input message policy, which is the alternative choice to encrypting based on lines
3288 (P055)-(P059).

3289 Lines (M012)-(M080) contain the wsse:Security header entry, which is the only header entry in this SOAP
3290 Header.

3291 Lines (M013)-(M015) contain the wsu:Timestamp which is required by the endpoint policy (P035).

3292 Lines (M016)-(M056) contain the SAML Assertion, which is required in the policy by the SamlToken
3293 (P009)-(P013) contained in the InitiatorToken. The SAML Assertion is version 1.1 (M020)-(M021) as
3294 required by the sp:WssSamlV11Token10 assertion (P011). The SAML Assertion is of type that uses the
3295 holder-of-key saml:ConfirmationMethod [[SAML11-CORE](#)] (M034)-(M036) as required by the fact that the
3296 SamlToken is contained in the InitiatorToken assertion of the policy (P009)-(P013), as explained in
3297 section 2.3 above.

3298 Line (M055) contains the ds:Signature of the Issuer of the SAML Assertion. While the details of this
3299 signature are not shown, it is this signature that the Recipient must ultimately trust in order to trust the rest
3300 of the message.

3301 Lines (M057)-(M084) contain the message signature in a ds:Signature element.

3302 Lines (M058)-(M073) contain the ds:SignedInfo element, which identifies the elements to be signed.

3303 Lines (M063)-(M067) contains a ds:Reference to the SOAP:Body, which is signed in the same manner as
3304 example section 2.3.1.4 above.

3305 Lines (M068)-(M072) contains a ds:Reference to the URI "timestamp", which again is signed in the same
3306 manner as the wsu:Timestamp in section 2.3.1.4 above.

3307 Lines (M075)-(M083) contain the ds:KeyInfo element, which identifies the key that should be used to
3308 verify this ds:Signature element. Lines (M076)-(M082) contain a wsse:SecurityTokenReference, which
3309 contains a wsse:KeyIdentifier that identifies the signing key as being contained in a token of
3310 wsse:ValueType "...wss-saml-token-profile-1.0#SAMLAssertionID", which means a SAML V1.1 Assertion
3311 [[WSS10-SAML11-PROFILE](#)]. Line (M080) contains the saml:AssertionID of the SAML Assertion that
3312 contains the signing key. This SAML Assertion is on lines (M016)-(M056) with the correct

3313 saml:AssertionID on line (M017). Note that the fact that the referenced token (the SAML Assertion) occurs
3314 in the wsse:Security header before this ds:KeyInfo element that uses it in compliance with the sp:Strict
3315 layout policy (P032) and the wsse:KeyIdentifier is an acceptable token reference mechanism as specified
3316 in the policy sp:Wss10 assertion containing a sp:MustSupportRefKeyIdentifier assertion (P041).

3317

3317 2.3.2 WSS 1.1 SAML Token Scenarios

3318 This section contains SamlToken examples that use WS-Security 1.1 SOAP Message Security [WSS11]
3319 and the WS-Security 1.1 SAML Profile [WSS11-SAML1120-PROFILE].

3320 2.3.2.1 (WSS1.1) SAML 2.0 Bearer

3321 This example is based on the Liberty Alliance Identity Web Services Framework (ID-WSF 2.0) Security
3322 Mechanism for the SAML 2.0 Profile for WSS 1.1 [WSS11-LIBERTY-SAML20-PROFILE], which itself is
3323 based on the [WSS11-SAML1120-PROFILE].

3324 In this example, an AsymmetricBinding is used for message protection provided by the Initiator, however,
3325 Recipient trust for the content is based only on the SAML bearer token provided by the Requestor.

3326

```
3327 (P001) <wsp:Policy>
3328 (P002)   <sp:AsymmetricBinding>
3329 (P003)     <wsp:Policy>
3330 (P004)       <sp:InitiatorToken>
3331 (P005)         <wsp:Policy>
3332 (P006)           <sp:X509Token sp:IncludeToken="http://docs.oasis-
3333 open.org/ws-sx/ws-
3334 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3335 (P007)             <wsp:Policy>
3336 (P008)               <sp:WssX509V3Token10/>
3337 (P009)             </wsp:Policy>
3338 (P010)           </sp:X509Token>
3339 (P011)         </wsp:Policy>
3340 (P012)       </sp:InitiatorToken>
3341 (P013)     <sp:RecipientToken>
3342 (P014)       <wsp:Policy>
3343 (P015)         <sp:X509Token sp:IncludeToken="http://docs.oasis-
3344 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
3345 (P016)           <wsp:Policy>
3346 (P017)             <sp:WssX509V3Token10/>
3347 (P018)           </wsp:Policy>
3348 (P019)         </sp:X509Token>
3349 (P020)       </wsp:Policy>
3350 (P021)     </sp:RecipientToken>
3351 (P022)   <sp:AlgorithmSuite>
3352 (P023)     <wsp:Policy>
3353 (P024)       <sp:Basic256/>
3354 (P025)     </wsp:Policy>
3355 (P026)   </sp:AlgorithmSuite>
3356 (P027) <sp:Layout>
3357 (P028)   <wsp:Policy>
3358 (P029)     <sp:Strict/>
3359 (P030)   </wsp:Policy>
3360 (P031) </sp:Layout>
3361 (P032) <sp:IncludeTimestamp/>
3362 (P033) <sp:OnlySignEntireHeadersAndBody/>
3363 (P034) </wsp:Policy>
3364 (P035) </sp:AsymmetricBinding>
3365 (P036) <sp:Wss11>
3366 (P037)   <wsp:Policy>
3367 (P038)     <sp:MustSupportRefKeyIdentifier/>
3368 (P039)     <sp:MustSupportRefIssuerSerial/>
3369 (P040)     <sp:MustSupportRefThumbprint/>
3370 (P041)     <sp:MustSupportRefEncryptedKey/>
3371 (P042)   </wsp:Policy>
3372 (P043) </sp:Wss11>
3373 (P044) <sp:SupportingTokens>
3374 (P045)   <wsp:Policy>
```

```

3375 (P046) <sp:SamIToken sp:IncludeToken="http://docs.oasis-open.org/ws-
3376 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3377 (P047) <wsp:Policy>
3378 (P048) <sp:WssSamIv20Token11/>
3379 (P049) </wsp:Policy>
3380 (P050) </sp:SamIToken>
3381 (P051) </wsp:Policy>
3382 (P052) </sp:SupportingTokens>
3383 (P053) </wsp:Policy>

```

3384 Lines (P001)-(P045) contain the endpoint policy, which contains 3 assertions: an sp:AsymmetricBinding
3385 assertion, an sp:Wss11 assertion, and an sp:SupportingTokens assertion.

3386 Lines (P002)-(P035) contain the sp:AsymmetricBinding assertion, which is identical to the same assertion
3387 that is described in section 2.1.3. Please refer to section 2.1.3 for details.

3388 Lines (P036)-(P043) contain an sp:Wss11 assertion, which contains a set of assertions that specify the
3389 token referencing techniques that are required to be supported (P038)-(P041).

3390 Lines (P044)-(P052) contain an sp:SupportingTokens assertion, which contains an sp:SamIToken
3391 (P046)-(P050), which specifies that it must always be sent to the Recipient. The sp:SamIToken contains
3392 an sp:WssSamIv20Token11 assertion (P048), which means that the SamIToken provided must be a
3393 SAML Version 2.0 Assertion that is submitted in compliance with the [\[WSS11-SAML1120-PROFILE\]](#).

3394 The fact that the sp:SamIToken is contained in an sp:SupportingTokens element indicates to the Initiator
3395 that the SAML Assertion must use the saml2:SubjectConfirmation@Method "bearer" as described above
3396 in introductory section 2.3.

3397 The following is an example request compliant with the above policy:

```

3398
3399 (M001) <?xml version="1.0" encoding="UTF-8"?>
3400 (M002) <s:Envelope xmlns:s=".../soap/envelope/" xmlns:sec="..."
3401 (M003) xmlns:wssse="..." xmlns:wsu="..." xmlns:wsa="...addressing"
3402 (M004) xmlns:sb="...liberty:sb" xmlns:pp="...liberty.id-sis-pp"
3403 (M005) xmlns:ds="...xmldsig#" xmlns:xenc="...xmlenc#">
3404 (M006) <s:Header>
3405 (M007) <!-- see Liberty SOAP Binding Spec for reqd, optional hdrs -->
3406 (M008) <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>
3407 (M009) <wsa:To wsu:Id="to">...</wsa:To>
3408 (M010) <wsa:Action wsu:Id="action">...</wsa:Action>
3409 (M011) <wsse:Security mustUnderstand="1">
3410 (M012) <wsu:Timestamp wsu:Id="ts">
3411 (M013) <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
3412 (M014) </wsu:Timestamp>
3413 (M015) <!-- this is the bearer token -->
3414 (M016) <saml2:Assertion xmlns:saml2="...SAML:2.0:assertion"
3415 (M017) Version="2.0" ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
3416 (M018) IssueInstant="2005-04-01T16:58:33.173Z">
3417 (M019) <saml2:Issuer>http://authority.example.com/</saml2:Issuer>
3418 (M020) <!-- signature by the issuer over the assertion -->
3419 (M021) <ds:Signature>...</ds:Signature>
3420 (M022) <saml2:Subject>
3421 (M023) <saml2:EncryptedID>
3422 (M024) <xenc:EncryptedData
3423 (M025) >U2XTCNvRX7B11NK182nmY00TEk==</xenc:EncryptedData>
3424 (M026) <xenc:EncryptedKey>...</xenc:EncryptedKey>
3425 (M027) </saml2:EncryptedID>
3426 (M028) <saml2:SubjectConfirmation
3427 (M029) Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3428 (M030) </saml2:Subject>
3429 (M031) <!-- audience restriction on assertion can limit scope of
3430 (M032) which entity should consume the info in the assertion. -->
3431 (M033) <saml2:Conditions NotBefore="2005-04-01T16:57:20Z"
3432 (M034) NotOnOrAfter="2005-04-01T21:42:4 3Z">
3433 (M035) <saml2:AudienceRestrictionCondition>

```

```

3434 (M036)         <saml2:Audience>http://wsp.example.com</saml2:Audience>
3435 (M037)         </saml2:AudienceRestrictionCondition>
3436 (M038)         </saml2:Conditions>
3437 (M039)         <!-- The AuthnStatement carries info that describes authN
3438 (M040)         event of the Subject to an Authentication Authority -->
3439 (M041)         <saml2:AuthnStatement AuthnInstant="2005-04-01T16:57:30.000Z"
3440 (M042)         SessionIndex="6345789">
3441 (M043)         <saml2:AuthnContext>
3442 (M044)         <saml2:AuthnContextClassRef
3443 (M045)         >...:SAML:2.0:ac:classes:PasswordProtectedTransport
3444 (M046)         </saml2:AuthnContextClassRef>
3445 (M047)         </saml2:AuthnContext>
3446 (M048)         </saml2:AuthnStatement>
3447 (M049)         <!-- This AttributeStatement carries an EncryptedAttribute.
3448 (M050)         Once this element decrypted with supplied key an <Attribute>
3449 (M051)         element bearing endpoint reference can be found, specifying
3450 (M052)         resources which invoker may access. Details on element can be
3451 (M053)         found in discovery service specification. -->
3452 (M054)         <saml2:AttributeStatement>
3453 (M055)         <saml2:EncryptedAttribute>
3454 (M056)         <xenc:EncryptedData
3455 (M057)         Type="http://www.w3.org/2001/04/xmlenc#Element" >
3456 (M058)         mQEMAzRniWkAAAEH9RWir0eKDKyFAB7PoFazx3ftp0vWwbbzqXdgcX8
3457 (M059)         ... hg6nZ5c0I6L6Gn9A=HCQY
3458 (M060)         </xenc:EncryptedData>
3459 (M061)         <xenc:EncryptedKey> ... </xenc:EncryptedKey>
3460 (M062)         </saml2:EncryptedAttribute>
3461 (M063)         </saml2:AttributeStatement>
3462 (M064)         </saml2:Assertion>
3463 (M065)         <!-- This SecurityTokenReference is used to reference the SAML
3464 (M066)         Assertion from a ds:Reference -->
3465 (M067)         <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsu="..."
3466 (M068)         xmlns:wss1="..." wsu:Id="str1" wss1:TokenType=
3467 (M069)         ".../wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
3468 (M070)         <wsse:KeyIdentifier ValueType=
3469 (M071)         ".../wss/oasis-wss-saml-token-profile-1.1#SAMLID"
3470 (M072)         >sxJu9g/vvLG9sAN9bKp/8q0NKU=</wsse:KeyIdentifier>
3471 (M073)         </wsse:SecurityTokenReference>
3472 (M074)         <ds:Signature>
3473 (M075)         <ds:SignedInfo>
3474 (M076)         <!-- in general include a ds:Reference for each wsa:header
3475 (M077)         added according to SOAP binding, plus timestamp, plus ref to
3476 (M078)         assertion to avoid token substitution attacks, plus Body -->
3477 (M079)         <ds:Reference URI="#to">...</ds:Reference>
3478 (M080)         <ds:Reference URI="#action">...</ds:Reference>
3479 (M081)         <ds:Reference URI="#mid">...</ds:Reference>
3480 (M082)         <ds:Reference URI="#ts">...</ds:Reference>
3481 (M083)         <ds:Reference URI="#Str1">
3482 (M084)         <ds:Transform Algorithm="...#STR-Transform">
3483 (M085)         <wsse:TransformationParameters>
3484 (M086)         <ds:CanonicalizationMethod Algorithm=
3485 (M087)         "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
3486 (M088)         </wsse:TransformationParameters>
3487 (M089)         </ds:Transform>
3488 (M090)         </ds:Reference>
3489 (M091)         <ds:Reference URI="#MsgBody">...</ds:Reference>
3490 (M092)         </ds:SignedInfo>
3491 (M093)         ...
3492 (M094)         </ds:Signature>
3493 (M095)         </wsse:Security>
3494 (M096)         </s:Header>
3495 (M097)         <s:Body wsu:Id="MsgBody">
3496 (M098)         <pp:Modify>
3497 (M099)         <!-- this is an ID-SIS-PP Modify message -->

```

```
3498 (M0100) </pp:Modify>
3499 (M0101) </s:Body>
3500 (M0102) </s:Envelope>
```

3501

3502 The sample message above was taken and cosmetically edited from the [[WSS11-LIBERTY-SAML20-](#)
3503 [PROFILE](#)].

3504 Lines (M002)-(M0102) contain the SOAP:Envelope, which contains the SOAP:Header (M006)-(M096)
3505 and the SOAP:Body (M097)-(M0101).

3506 The SOAP Header contains some WS-Addressing (wsa:) parameters (M008)-(M010) (not discussed
3507 here) and a wsse:Security header.

3508 The wsse:Security header (M011)-(M095) contains a wsu:Timestamp (M012)-(M014), a saml2:Assertion
3509 (M016)-(M064), a wsse:SecurityTokenReference (M067)-(M073), and a ds:Signature (M074)-(M094).

3510 The wsu:Timestamp (M012)-(M014) is required by the policy sp:IncludeTimestamp assertion (P032).

3511 The saml2:Assertion (M016)-(M064) is required by the policy sp:WssSamIV20Token11 (P048). The
3512 saml2:Assertion is Version 2.0 (M017) and it is signed by the IssuingAuthority (M021). This is the
3513 ds:Signature (M021) of the IssuingAuthority, identified in the saml2:Issuer element (M019), that the
3514 Recipient trusts with respect to recognizing the Requestor and determining whether the request should be
3515 granted. In addition, this saml2:Assertion uses the "bearer" saml2:SubjectConfirmation@Method, as
3516 required by the policy sp:SamIToken in the sp:SupportingTokens element described above
3517 (P044)-(P052).

3518 Note: the saml2:Assertion contains a saml2:EncryptedID (M023)-(M027), which contains the
3519 identity of the Requestor and a saml2:EncryptedAttribute (M062), which contains information
3520 about the Requestor. It is presumed that prior to issuing this request, that the Requestor
3521 contacted the IssuingAuthority (directly or indirectly) and was granted permission to access the
3522 web service that is now the target of this request. As such, the IssuingAuthority has knowledge of
3523 this web service and presumably the public certificate associated with that service and has used
3524 the public key contained in that certificate to encrypt the aforementioned portions of the
3525 saml2:Assertion, which can only be decrypted by the RelyingParty who presumably has entrusted
3526 the private key of the service with the Recipient, in order that the Recipient may decrypt the
3527 necessary data.

3528 However, before the Recipient can evaluate these aspects of the request, the requirements of the policy
3529 sp:AsymmetricBinding (P002)-(P035) must be met in terms of proper "presentation" of the request as
3530 described below.

3531 Lines (M074)-(M094) contain the message signature ds:Signature element, which contains a
3532 ds:SignedInfo that contains ds:Reference elements that cover the wsa: headers (M079)-(M081), the
3533 wsu:Timestamp (M082) (required by the policy sp:IncludeTimestamp assertion (P032), the
3534 saml2:Assertion (M083)-(M090), and the SOAP:Body (M091).

3535 The ds:Reference (M083)-(M090) covering the saml2:Assertion warrants further examination.

3536 The first point to note is that to reference the saml2:Assertion the ds:Reference uses an STR-
3537 Transform (M084) to reference a wsse:SecurityTokenReference (M067)-(M073), which is in
3538 compliance with policy sp:Wss11 sp:MustSupportRefKeyIdentifier assertion (P038).

3539 The second point to note about this ds:Reference is that is covering the saml2:Assertion even
3540 though the policy references the sp:SamIToken in an sp:SupportingTokens element and not an
3541 sp:SignedSupportingTokens element. The reason this is the case is that it is the fact that an
3542 sp:SupportingTokens (P044)-(P052) element is used that tells the Initiator that a SAML bearer
3543 Assertion is required. However, this only means that the SamIToken is not "required" to be
3544 signed. It is the Initiator's choice whether to sign it or not, and it is generally good practice to do
3545 so in order to prevent a token substitution attack.

3546

3547

3547 2.3.2.2 (WSS1.1) SAML2.0 Sender Vouches over SSL

3548 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP Scenario](#)
3549 #2].

3550 Similar to 2.3.1.2 except SAML token is of version 2.0.

3551

3552

```
3553 (P001) <wsp:Policy>  
3554 (P002)   <sp:TransportBinding>  
3555 (P003)   <wsp:Policy>  
3556 (P004)   <sp:TransportToken>  
3557 (P005)   <wsp:Policy>  
3558 (P006)   <sp:HttpsToken>  
3559 (P007)   <wsp:Policy>  
3560 (P008)   <sp:RequireClientCertificate>  
3561 (P009)   </wsp:Policy>  
3562 (P010)   </sp:HttpsToken>  
3563 (P011)   </wsp:Policy>  
3564 (P012)   </sp:TransportToken>  
3565 (P013)   <sp:AlgorithmSuite>  
3566 (P014)   <wsp:Policy>  
3567 (P015)   <sp:Basic256 />  
3568 (P016)   </wsp:Policy>  
3569 (P017)   </sp:AlgorithmSuite>  
3570 (P018)   <sp:Layout>  
3571 (P019)   <wsp:Policy>  
3572 (P020)   <sp:Strict />  
3573 (P021)   </wsp:Policy>  
3574 (P022)   </sp:Layout>  
3575 (P023)   <sp:IncludeTimestamp />  
3576 (P024)   </wsp:Policy>  
3577 (P025)   </sp:TransportBinding>  
3578 (P026)   <sp:SignedSupportingTokens>  
3579 (P027)   <wsp:Policy>  
3580 (P028)   <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
3581 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3582 (P029)   <wsp:Policy>  
3583 (P030)   <sp:WssSamlV20Token11/>  
3584 (P031)   </wsp:Policy>  
3585 (P032)   </sp:SamlToken>  
3586 (P033)   </wsp:Policy>  
3587 (P034)   </sp:SignedSupportingTokens>  
3588 (P035)   </wsp:Policy>
```

3589 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by
3590 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and
3591 sp:SignedSupportingTokens (P026).

3592 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
3593 by a secure transport protocol such as SSL or TLS.

3594 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
3595 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
3596 transport token using the Basic256 algorithm suite (P015).

3597 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of
3598 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section
3599 4.3.1].

3600 The requirements for the sp:Layout assertion (P018)-(P022) should be automatically met (or may be
3601 considered moot) since there are no cryptographic tokens required to be present in the WS-Security
3602 header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout
3603 would need to be considered.)

3604 A timestamp (P023) is required to be included in an acceptable message.

3605

3606 Here is an example request:

3607

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
(M002) <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
(M003)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
(M004)   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
(M005)   xmlns:wsu=".../oasis-200401-wsswssecurity-utility-1.0.xsd"
(M006)   xmlns:wssse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
(M007)   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
(M008)   <S11:Header>
(M009)     <wsse:Security S11:mustUnderstand="1">
(M010)       <wsu:Timestamp>
(M011)         <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
(M012)       </wsu:Timestamp>
(M013)       <saml2:Assertion ID=" a75adf55-01d7-40cc-929f-dbd8372ebdfc"
(M014)         IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
(M015)         <saml2:Issuer>www.opensaml.org</saml2:Issuer>
(M016)         <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
(M017)           NotOnOrAfter="2006-06-19T17:08:33.173Z" />
(M018)         <saml2:Subject>
(M019)           <saml2:NameID Format=
(M020)             "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
(M021)           >uid=joe,ou=people,ou=saml demo,o=example.com</saml2:NameID>
(M022)           <saml2:SubjectConfirmation Method=
(M023)             "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches" />
(M024)         </saml2:Subject>
(M025)         <saml2:AttributeStatement>
(M026)           <saml2:Attribute>...</saml2:Attribute>
(M027)           <saml2:Attribute>...</saml2:Attribute>
(M028)         </saml2:AttributeStatement>
(M029)       </saml2:Assertion>
(M030)     </wsse:Security>
(M031)   </S11:Header>
(M032)   <S11:Body>
(M033)     <Ping xmlns="http://xmlsoap.org/Ping">
(M034)       <text>EchoString</text>
(M035)     </Ping>
(M036)   </S11:Body>
(M037) </S11:Envelope>
```

3645 Lines (M002)-(M037) contain the SOAP:Envelope, which contains the SOAP:Header (M003)-(M031) and
3646 the SOAP:Body (M032)-(M036).

3647 Lines (M009)-(M030) contain the wsse:Security header, which, in conjunction with the client
3648 certificate (P008), is the primary basis for trust in this example.

3649 Lines (M010)-(M012) contain the wsu:Timestamp, which meets the sp:IncludeTimestamp assertion policy
3650 requirement (P023).

3651 Lines (M013)-(M029) contain the saml2:Assertion, which is required to be Version 2.0 (M014) and to be
3652 used within the [WSS11-SAML1120-PROFILE], because of the policy sp:SamIToken assertion
3653 (P028)-(P032), which contains the sp:WssSamIV20Token11 (P030).

3654 Lines (M022)-(M023) indicate that the SAML Assertion uses the saml2:Method sender-vouches for the
3655 purposes of saml2:SubjectConfirmation, which is required by the fact that the sp:SamIToken appears in
3656 an sp:SignedSupportingTokens assertion (P026)-(P034) in conjunction with the client certificate required
3657 by the sp:RequireClientCertificate assertion (P008) within the sp:TransportBinding as described in section
3658 2.3 above. In addition, the Recipient should be able to correlate the saml2:Issuer (M015) as being
3659 properly associated with the client certificate received from the HTTPS (SSL) connection.

3660

3661

3661 2.3.2.3 (WSS1.1) SAML2.0 HoK over SSL

3662 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)
3663 Scenario #5].

3664 Similar to 2.3.1.3 except SAML token is of version 2.0.

3665 Initiator adds a SAML Assertion (hk) to the SOAP Security Header. In this policy the sp:TransportBinding
3666 requires a Client Certificate AND the sp:Sam1Token is in an sp:SignedEndorsingSupportingTokens
3667 element. A SAML holder-of-key Assertion meets these requirements because it is “virtually signed” by the
3668 message signature as a result of the SSL client certificate authentication procedure as described in
3669 section 2.3. Furthermore, the SAML hk Assertion in this case is a “virtually endorsing”, because the key
3670 identified in the holder-of-key saml2:SubjectConfirmationData is also the client certificate, which is
3671 virtually endorsing its own signature, under the authority of the IssuingAuthority who has signed the
3672 SAML hk Assertion.

3673 As a result, the Initiator may be considered to be authorized by the saml2:Issuer of the hk SAML
3674 Assertion to bind message content to the Subject of the Assertion. If the Client Certificate matches the
3675 certificate identified in the hk Assertion, the Initiator may be regarded as executing SAML hk responsibility
3676 of binding the Requestor, who would be the Subject of the hk Assertion, to the content of the message.

3677 (Note: the same considerations described in section 2.3.1.4 with respect to whether the Subject
3678 of the Assertion and the Subject of the Client Certificate are the same entity determining whether
3679 the Client Certificate is attesting for the Assertion Subject or whether the Client Certificate is
3680 authenticating as the Assertion Subject apply here.)

3681 In this scenario, the IssuingAuthority is the saml2:Issuer(signer) of the hk SAML Assertion. The Requestor
3682 is the Subject of the Assertion and the Initiator is authorized by the IssuingAuthority to bind the Assertion
3683 to the message using the ClientCertificate identified in the SAML Assertion, which may also be
3684 considered to be virtually signing the wsu:Timestamp of the message. Optionally, a separate Signature
3685 may be used to sign the wsu:Timestamp, which the Recipient would also be required to verify was signed
3686 by the client certificate in this example.

3687

```
3688 (P001) <wsp:Policy>  
3689 (P002) <sp:TransportBinding>  
3690 (P003) <wsp:Policy>  
3691 (P004) <sp:TransportToken>  
3692 (P005) <wsp:Policy>  
3693 (P006) <sp:HttpsToken>  
3694 (P007) <wsp:Policy>  
3695 (P008) <sp:RequireClientCertificate>  
3696 (P009) </wsp:Policy>  
3697 (P010) </sp:HttpsToken>  
3698 (P011) </wsp:Policy>  
3699 (P012) </sp:TransportToken>  
3700 (P013) <sp:AlgorithmSuite>  
3701 (P014) <wsp:Policy>  
3702 (P015) <sp:Basic256 />  
3703 (P016) </wsp:Policy>  
3704 (P017) </sp:AlgorithmSuite>  
3705 (P018) <sp:Layout>  
3706 (P019) <wsp:Policy>  
3707 (P020) <sp:Strict />  
3708 (P021) </wsp:Policy>  
3709 (P022) </sp:Layout>  
3710 (P023) <sp:IncludeTimestamp />  
3711 (P024) </wsp:Policy>  
3712 (P025) </sp:TransportBinding>  
3713 (P026) <sp:SignedEndorsingSupportingTokens>  
3714 (P027) <wsp:Policy>  
3715 (P028) <sp:Sam1Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
3716 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3717 (P029) <wsp:Policy>
```



```

3718      (P030)          <sp:WssSamlV20Token11/>
3719      (P031)          </wsp:Policy>
3720      (P032)          </sp:SamlToken>
3721      (P033)          </wsp:Policy>
3722      (P034)          </sp:SignedEndorsingSupportingTokens>
3723      (P035)          </wsp:Policy>

```

3724

3725 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by
3726 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and
3727 sp:EndorsingSupportingTokens (P026).

3728 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
3729 by a secure transport protocol such as SSL or TLS.

3730 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
3731 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
3732 transport token using the Basic256 algorithm suite (P015).

3733 In addition, because this is SAML holder-of-key, a client certificate is required (P008) as the basis of trust
3734 for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section 4.3.1]. In
3735 the holder-of-key case, there will be an additional certificate required in the trust chain, which is the
3736 certificate used to sign the SAML hk Assertion, which will be contained or referenced in the Assertion.

3737 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)
3738 since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a
3739 signature element was included to cover the wsse:Timestamp, then the layout would need to be
3740 considered.)

3741 A timestamp (P023) is required to be included in an acceptable message.

3742 Lines (P026)-(P034) contain an sp:SignedEndorsingSupportingTokens element, which means that the
3743 contained supporting token (the SAML hk Assertion) references a key that will be “signing” (endorsing)
3744 the message signature. The token itself may also be considered to be “signed”, because it is contained in
3745 the message sent over the SSL link. In the case of sp:TransportBinding, there may be no actual
3746 “message signature”, however, when a client certificate is used, the service can be assured that the
3747 connection was set up by the client and that the SSL link guarantees the integrity of the data that is sent
3748 on the link by the client, while it is on the link and when it is received from the link by using the key
3749 referenced by the token. However, it does not guarantee the integrity after the data is received (i.e. after it
3750 is received there is no way to tell whether any changes have been made to it since it has been received).
3751 In any event, within this context a Signed Endorsing Supporting Token can be used to tell the Recipient
3752 that the Issuer of the token is making claims related to the holder of the private key that is referenced by
3753 the token, which in this case would be the private key associated with the client certificate used to set up
3754 the SSL link, as described further below.

3755 Lines (P028)-(P032) contain an sp:SamlToken, which must always be sent to the Recipient.

3756 Line (P030) specifies that the token is of type WssSamlV20Token11, which means that the Endorsing
3757 Supporting Token must be a SAML Version 2.0 token and it must be sent using WS-Security 1.1 using
3758 the [[WSS11-SAML1120-PROFILE](#)]. Note that because the SamlToken is contained in an
3759 sp:EndorsingSupportingTokens element, that it implicitly must be a holder-of-key token as described in
3760 section 2.3 above.

3761 Here is an example request taken from the WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)
3762 Scenario #5] containing only minor cosmetic modifications and corrections.

3763

```

3764      (M001)          <?xml version="1.0" encoding="utf-8" ?>
3765      (M002)          <S11:Envelope
3766      (M003)              xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3767      (M004)              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3768      (M005)              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3769      (M006)              xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3770      (M007)              xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3771      (M008)              xmlns:wssell=".../oasis-wss-wssecurity-secext-1.1.xsd"

```

```

3772 (M009)      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3773 (M010)      xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3774 (M011)      <S11:Header>
3775 (M012)      <wsse:Security S11:mustUnderstand="1">
3776 (M013)      <wsu:Timestamp>
3777 (M014)      <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3778 (M015)      </wsu:Timestamp>
3779 (M016)      <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3780 (M017)      IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3781 (M018)      <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3782 (M019)      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3783 (M020)      <ds:SignedInfo>
3784 (M021)      <ds:CanonicalizationMethod
3785 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3786 (M023)      <ds:SignatureMethod
3787 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3788 (M025)      <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3789 (M026)      <ds:Transforms>
3790 (M027)      <ds:Transform Algorithm=
3791 (M028)      "http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
3792 (M029)      <ds:Transform Algorithm=
3793 (M030)      "http://www.w3.org/2001/10/xml-exc-c14n#">
3794 (M031)      <InclusiveNamespaces
3795 (M032)      PrefixList="#default saml ds xs xsi"
3796 (M033)      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3797 (M034)      </ds:Transform>
3798 (M035)      </ds:Transforms>
3799 (M036)      <ds:DigestMethod Algorithm=
3800 (M037)      "http://www.w3.org/2000/09/xmldsig#sha1" />
3801 (M038)      <ds:DigestValue
3802 (M039)      >Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3803 (M040)      </ds:Reference>
3804 (M041)      </ds:SignedInfo>
3805 (M042)      <ds:SignatureValue>
3806 (M043)      hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgzpkJN9CMLG8ENR4Nrw+n
3807 (M044)      7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmtpt3TD
3808 (M045)      MwuL/cBUj20tBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3809 (M046)      </ds:SignatureValue>
3810 (M047)      <ds:KeyInfo>
3811 (M048)      <ds:X509Data>
3812 (M049)      <ds:X509Certificate>
3813 (M050)      MIICyjcCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxkCzAJBgNVBAYTA1VT
3814 (M051)      ...
3815 (M052)      8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
3816 (M053)      </ds:X509Certificate>
3817 (M054)      </ds:X509Data>
3818 (M055)      </ds:KeyInfo>
3819 (M056)      </ds:Signature>
3820 (M057)      <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3821 (M058)      NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3822 (M059)      <saml2:Subject>
3823 (M060)      <saml2:NameID
3824 (M061)      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3825 (M062)      >uid=joe,ou=people,ou=saml-demo,o=example.com</saml2:NameID>
3826 (M063)      <saml2:SubjectConfirmation
3827 (M064)      Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" />
3828 (M065)      <saml2:SubjectConfirmationData
3829 (M066)      xsi:type="saml2:KeyInfoConfirmationDataType">
3830 (M067)      <ds:KeyInfo>
3831 (M068)      <ds:X509Data>
3832 (M069)      <ds:X509IssuerName>
3833 (M070)      C=ZA, ST=Western Cape, L=Cape Town,
3834 (M071)      O=Thawte Consulting cc,
3835 (M072)      OU=Certification Services Division,

```

```

3836 (M073)          CN=Thawte Server CA/Email=server-certs@thawte.com
3837 (M074)          </ds:X509IssuerName>
3838 (M075)          <X509SerialNumber>12345678</X509SerialNumber>
3839 (M076)          </ds:X509Data>
3840 (M077)          </ds:KeyInfo>
3841 (M078)          </saml2:SubjectConfirmationData>
3842 (M079)          </saml2:Subject>
3843 (M080)          <saml2:AttributeStatement>
3844 (M081)          <saml2:Attribute Name="MemberLevel">
3845 (M082)          <saml2:AttributeValue>gold</saml2:AttributeValue>
3846 (M083)          </saml2:Attribute>
3847 (M084)          <saml2:Attribute Name="E-mail">
3848 (M085)          <saml2:AttributeValue
3849 (M086)          >joe@yahoo.com</saml2:AttributeValue>
3850 (M087)          </saml2:Attribute>
3851 (M088)          </saml2:AttributeStatement>
3852 (M089)          </saml2:Assertion>
3853 (M090)          </wsse:Security>
3854 (M091)          </S11:Header>
3855 (M092)          <S11:Body wsu:Id="MsgBody">
3856 (M093)          <ReportRequest>
3857 (M094)          <TickerSymbol>SUNW</TickerSymbol>
3858 (M095)          </ReportRequest>
3859 (M096)          </S11:Body>
3860 (M097)          </S11:Envelope>

```

3861 Lines (M002)-(M097) contain the SOAP:Envelope, which contains the SOAP:Header (M011)-(M091) and
3862 the SOAP:Body (M092)-(M096).

3863 Lines (M012)-(M090) contain the wsse:Security header, which contains a wsu:Timestamp (M013)-(M015)
3864 and a saml2:Assertion (M016)-(M089).

3865 The wsu:Timestamp (M013)-(M015) may be considered to be virtually signed by the client certificate as
3866 explained above and in section 2.3.

3867 The saml2:Assertion was issued by an IssuingAuthority identified in the saml2:Issuer element (M018).

3868 The saml2:Issuer has used the private key of its enterprise certificate to produce the ds:Signature
3869 element (M019)-(M056) that is an enveloped-signature (M028) that covers its parent XML element, the
3870 saml2:Assertion (M016)-(M089). The ds:KeyInfo (M047)-(M055) within this ds:Signature contains an
3871 actual copy of the Issuer's enterprise certificate, that the Recipient can verify for authenticity to establish
3872 that this message is in conformance with any agreement the RelyingParty may have with the
3873 IssuingAuthority. The details of this verification are outside the scope of this document, however they
3874 involve the structures and systems the RelyingParty has in place recognize and verify certificates and
3875 signatures it receives that are associated with business arrangements with the holders of the private keys
3876 of those certificates.

3877 The saml2:Subject of the saml2:Assertion is contained in lines (M059)-(M079). Within the saml2:Subject
3878 is the saml2:NameID (M060)-(M062), which contains the official name of the Subject of this Assertion and
3879 this entity may be considered to the Requestor associated with this request.

3880 The saml2:SubjectConfirmation (M063)-(M064) has Method "holder-of-key" which implies that there is a
3881 saml2:SubjectConfirmationData element (M065)-(M078) which will contain information that identifies a
3882 signing key that the Initiator will use to bind this saml2:Assertion to a message that is associated with the
3883 Requestor. In this context, the Initiator is acting as "attesting entity" with respect to the Subject as defined
3884 in [SAML20], which means that the Initiator is authorized by the IssuingAuthority to present
3885 saml2:Assertions pertaining to saml2:Subjects to Recipients/RelyingParties. In this example there is a
3886 ds:KeyInfo (M067)-(M077) that identifies a specific certificate (ds:X509IssuerName (M069)-(M074) and
3887 ds:SerialNumber (M075)) that the Initiator must prove possession of the associated private key to verify
3888 this message. In this policy that private key is the one associated with the client certificate that the Initiator
3889 is required to use (P008).

3890 The saml2:AttributeStatement (M080)-(M088) contains some information about the Subject that is
3891 officially being provided in this saml2:Assertion by the IssuingAuthority that may have some significance
3892 to the Relying Party in terms of determining whether access is granted for this request.

3893

3893 **2.3.2.4 (WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign,**
3894 **Encrypt**

3895 Here the message and SAML Assertion are signed using a key derived from the ephemeral key K. The
3896 ephemeral key is encrypted using the Recipient's public key for the request input message and the same
3897 shared ephemeral key, K, is encrypted using the Initiators public key for the response output message.

3898 Alternatively, derived keys can be used for each of signing and encryption operations.

3899 In this scenario the Authority is the Initiator who signs the message with the generated key. In order to
3900 establish trust in the generated key, the Initiator must sign the message signature with a second signature
3901 using an X509 certificate, which is indicated as the EndorsingSupportingToken. This X509 certificate
3902 establishes the Initiator as the SAML Authority.

3903 (Note: there are instructive similarities and differences between this example and example 2.3.1.4, where
3904 the difference is that: here the binding is symmetric and the WSS1.1 is used, whereas in 2.3.1.4 the
3905 binding is asymmetric and WSS1.0 is used. One particular item is that in 2.3.1.4 an
3906 EndorsingSupportingToken is not needed because in 2.3.1.4 the asymmetric binding uses X.509
3907 certificates, which may be inherently trusted as opposed to the ephemeral key, K, used here.)

3908

```
3909 (P001) <wsp:Policy wsu:Id="WSS11SamlWithCertificates_policy">
3910 (P002)   <wsp:ExactlyOne>
3911 (P003)     <wsp>All>
3912 (P004)       <sp:SymmetricBinding>
3913 (P005)         <wsp:Policy>
3914 (P006)           <sp:ProtectionToken>
3915 (P007)             <wsp:Policy>
3916 (P008)               <sp:X509Token sp:IncludeToken=
3917 "http://docs.oasis-open.org/ws-sx/ws-
3918 securitypolicy/200702/IncludeToken/Never">
3919 (P009)                 <wsp:Policy>
3920 (P010)                   <sp:RequireThumbprintReference/>
3921 (P011)                   <sp:RequireDerivedKeys wsp:Optional="true"/>
3922 (P012)                   <sp:WssX509V3Token10/>
3923 (P013)                 </wsp:Policy>
3924 (P014)               </sp:X509Token>
3925 (P015)             </wsp:Policy>
3926 (P016)           </sp:ProtectionToken>
3927 (P017)         <sp:AlgorithmSuite>
3928 (P018)           <wsp:Policy>
3929 (P019)             <sp:Basic256/>
3930 (P020)           </wsp:Policy>
3931 (P021)         </sp:AlgorithmSuite>
3932 (P022)       <sp:Layout>
3933 (P023)         <wsp:Policy>
3934 (P024)           <sp:Strict/>
3935 (P025)         </wsp:Policy>
3936 (P026)       </sp:Layout>
3937 (P027)     <sp:IncludeTimestamp/>
3938 (P028)     <sp:OnlySignEntireHeadersAndBody/>
3939 (P029)   </wsp:Policy>
3940 (P030) </sp:SymmetricBinding>
3941 (P031) <sp:SignedSupportingTokens>
3942 (P032)   <wsp:Policy>
3943 (P033)     <sp:SamlToken sp:IncludeToken=
3944 "http://docs.oasis-open.org/ws-sx/ws-
3945 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3946 (P034)       <wsp:Policy>
3947 (P035)         <sp:WssSamlV11Token11/>
3948 (P036)       </wsp:Policy>
3949 (P037)     </sp:SamlToken>
3950 (P038)   </wsp:Policy>
3951 (P039) </sp:SignedSupportingTokens>
```

```

3952 (P040) <sp:EndorsingSupportingTokens>
3953 (P041) <wsp:Policy>
3954 (P042) <sp:X509Token sp:IncludeToken="AlwaysToRecipient">
3955 (P043) <wsp:Policy>
3956 (P044) <sp:WssX509V3Token11/>
3957 (P045) </wsp:Policy>
3958 (P046) </sp:X509Token>
3959 (P047) </wsp:Policy>
3960 (P048) </sp:EndorsingSupportingTokens>
3961 (P049) <sp:Wss11>
3962 (P050) <wsp:Policy>
3963 (P051) <sp:MustSupportRefKeyIdentifier/>
3964 (P052) <sp:MustSupportRefIssuerSerial/>
3965 (P053) <sp:MustSupportRefThumbprint/>
3966 (P054) <sp:MustSupportRefEncryptedKey/>
3967 (P055) </wsp:Policy>
3968 (P056) </sp:Wss11>
3969 (P057) </wsp:All>
3970 (P058) </wsp:ExactlyOne>
3971 (P059) </wsp:Policy>
3972 (P060)
3973 (P061) <wsp:Policy wsu:Id="SamlForCertificates_input_policy">
3974 (P062) <wsp:ExactlyOne>
3975 (P063) <wsp:All>
3976 (P064) <sp:SignedParts>
3977 (P065) <sp:Body/>
3978 (P066) </sp:SignedParts>
3979 (P067) <sp:EncryptedParts>
3980 (P068) <sp:Body/>
3981 (P069) </sp:EncryptedParts>
3982 (P070) </wsp:All>
3983 (P071) </wsp:ExactlyOne>
3984 (P072) </wsp:Policy>
3985 (P073)
3986 (P074) <wsp:Policy wsu:Id="SamlForCertificate_output_policy">
3987 (P075) <wsp:ExactlyOne>
3988 (P076) <wsp:All>
3989 (P077) <sp:SignedParts>
3990 (P078) <sp:Body/>
3991 (P079) </sp:SignedParts>
3992 (P080) <sp:EncryptedParts>
3993 (P081) <sp:Body/>
3994 (P082) </sp:EncryptedParts>
3995 (P083) </wsp:All>
3996 (P084) </wsp:ExactlyOne>
3997 (P085) </wsp:Policy>

```

3998 Lines (P001)-(P059) contain the policy that requires all the contained assertions to be complied with by
3999 the Initiator. In this case there are 4 assertions: sp:SymmetricBinding (P004)-(P030),
4000 sp:SignedSupportingTokens (P031)-(P039), sp:EndorsingSupportingTokens (P040)-(P048), and
4001 sp:Wss11 (P049)-(P056).

4002 The sp:SymmetricBinding assertion (P004) contains an sp:ProtectionToken (P006)-(P016), which
4003 indicates that both the Initiator and Recipient must use each other's public key respectively associated
4004 with the X509Token (P008)-(P014) associated with the respective message receiver to encrypt the
4005 shared ephemeral symmetric key as explained at the beginning of this section. The messages may either
4006 be encrypted and signed using keys derived from the ephemeral key as indicated by the
4007 sp:RequireDerivedKeys assertion (P011) or the encryption and signing can be done using the ephemeral
4008 key itself without deriving keys, because the RequireDerivedKey assertion is an optional requirement as
4009 indicated by the wsp:Optional attribute on line (P011).

4010 The sp:SignedSupportingTokens assertion (P031) contains an sp:SamIToken assertion (P033)-(P037),
4011 which indicates that a signed SAML Assertion must always be included in the Initiator's request to the
4012 Recipient (AlwaysToRecipient (P033)). This SAML Assertion must be included in the WS-Security header

4013 and referenced and signed as described in the WS-Security 1.1 Profile for SAML [[WSS11-SAML1120-](#)
4014 [PROFILE](#)] as indicated by the sp:WssSamIV11Token11 assertion (P036), which indicates the SAML 1.1
4015 option in that profile (as opposed to the SAML 2.0 option, which would have been indicated by
4016 sp:WssSamIV20Token11).

4017 The sp:EndorsingSupportingToken assertion (P040) is needed because there are no guarantees that the
4018 ephemeral key, K, is still being used by the Initiator with whom the Recipient would have collaborated
4019 originally to establish the ephemeral key as a shared secret. The purpose of the endorsing token is to
4020 sign the signature made by the ephemeral key, using the Initiator's private key, which will explicitly
4021 guarantee the content of this particular Initiator request. The endorsing token in this policy is an
4022 sp:X509Token (P042)-(P046), which, in particular, is an sp:WssX509V3Token11, which must be used in
4023 accordance with the WS-Security 1.1 Profile for X509 Tokens [[WSS11-X509-PROFILE](#)]. (Note: it may be
4024 the case that this X509 certificate is the same X509 certificate referred to in the ProtectionToken
4025 assertion (P012). However, it is important to keep in mind that line (P012) only indicates that the Initiator's
4026 token will be used by the Recipient to protect the ephemeral key for the symmetric binding. Therefore, the
4027 fact that the token is identified in line (P012) as an sp:X509V3Token10 is not directly related to the fact
4028 that the same key may be used for the additional purpose of explicitly signing the request message).

4029 The sp:Wss11 assertion (P049-P056) indicates that WS-Security 1.1 constructs are accepted. (Note also
4030 that eitherWssX509V3Token10 or WssX509V3Token11 may be used with the Wss11 since both WS-
4031 Security 1.0 and WS-Security 1.1 Profiles are supported by WS-Security 1.1)

4032 There are also 2 Policys, one each for the input message and the output message, each of which
4033 contains an assertion indicating the message SOAP Body must be signed (P064)-(P066) for the input
4034 message and (P077)-(P079) for the output message, and each contains an assertion that the message
4035 SOAP Body must be encrypted (P067)-(P069) for the input message and (P080)-(P082) for the output
4036 message.

4037 The following is a sample request that is compliant with this policy.

```
4038 (M001) <?xml version="1.0" encoding="utf-8" ?>
4039 (M002) <S12:Envelope
4040 (M003)   xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
4041 (M004)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4042 (M005)   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4043 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4044         wss-wssecurity-secext-1.0.xsd"
4045 (M007)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4046         wssecurity-utility-1.0.xsd"
4047 (M008)   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4048 (M009)   xmlns:xenc="..."
4049 (M010)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
4050 (M011) <S12:Header>
4051 (M012)   <wsse:Security S12:mustUnderstand="1">
4052 (M013)     <wsu:Timestamp wsu:Id="timestamp">
4053 (M014)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
4054 (M015)     </wsu:Timestamp>
4055 (M016)     <xenc:EncryptedKey Id="encKey" >
4056 (M017)       <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
4057 (M018)         <ds:DigestMethod Algorithm="...#sha1"/>
4058 (M019)       </xenc:EncryptionMethod>
4059 (M020)       <ds:KeyInfo >
4060 (M021)         <wsse:SecurityTokenReference >
4061 (M022)           <wsse:KeyIdentifier EncodingType="...#Base64Binary"
4062             ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
4063 (M023)         </wsse:SecurityTokenReference>
4064 (M024)       </ds:KeyInfo>
4065 (M025)       <xenc:CipherData>
4066 (M026)         <xenc:CipherValue>TE...=</xenc:CipherValue>
4067 (M027)       </xenc:CipherData>
4068 (M028)     </xenc:EncryptedKey>
4069 (M029)     <n1:ReferenceList xmlns:n1=".../xmlenc#">
4070 (M030)       <n1:DataReference URI="#encBody"/>
4071 (M031)     </n1:ReferenceList>
```

```

4072 (M032) <saml:Assertion
4073 (M033)   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
4074 (M034)   IssueInstant="2003-04-17T00:46:02Z"
4075 (M035)   Issuer="www.opensaml.org"
4076 (M036)   MajorVersion="1"
4077 (M037)   MinorVersion="1">
4078 (M038)   <saml:Conditions
4079 (M039)     NotBefore="2002-06-19T16:53:33.173Z"
4080 (M040)     NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
4081 (M041)   <saml:AttributeStatement>
4082 (M042)     <saml:Subject>
4083 (M043)       <saml:NameIdentifier
4084 (M044)         NameQualifier="www.example.com"
4085 (M045)         Format="">
4086 (M046)         uid=joe,ou=people,ou=saml-demo,o=example.com
4087 (M047)       </saml:NameIdentifier>
4088 (M048)     <saml:SubjectConfirmation>
4089 (M049)       <saml:ConfirmationMethod>
4090 (M050)         urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
4091 (M051)       </saml:ConfirmationMethod>
4092 (M052)     </saml:SubjectConfirmation>
4093 (M053)   </saml:Subject>
4094 (M054)   <saml:Attribute>
4095 (M055)     ...
4096 (M056)   </saml:Attribute>
4097 (M057)   ...
4098 (M058) </saml:AttributeStatement>
4099 (M059) </saml:Assertion>
4100 (M060) <wsse:SecurityTokenReference wsu:id="STR1">
4101 (M061)   <wsse:KeyIdentifier wsu:id="..."
4102 (M062)     ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
4103 (M063)     profile-1.0#SAMLAssertionID">
4104 (M064)       _a75adf55-01d7-40cc-929f-dbd8372ebdfc
4105 (M064)     </wsse:KeyIdentifier>
4106 (M065)   </wsse:SecurityTokenReference>
4107 (M066) <wsse:BinarySecurityToken
4108 (M067)   wsu:Id="attesterCert"
4109 (M068)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4110 (M069)   wss-x509-token-profile-1.0#X509v3"
4111 (M069)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
4112 (M069)   200401-wss-soap-message-security-1.0#Base64Binary">
4113 (M070)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
4114 (M071) </wsse:BinarySecurityToken>
4115 (M072) <ds:Signature wsu:Id="message-signature">
4116 (M073)   <ds:SignedInfo>
4117 (M074)     <ds:CanonicalizationMethod
4118 (M075)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4119 (M076)     <ds:SignatureMethod
4120 (M077)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4121 (M078)     <ds:Reference URI="#STR1">
4122 (M079)       <ds:Transforms>
4123 (M080)         <ds:Transform
4124 (M081)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
4125 (M081)           200401-wss-soap-message-security-1.0#STR-Transform">
4126 (M082)         <wsse:TransformationParameters>
4127 (M083)           <ds:CanonicalizationMethod
4128 (M084)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4129 (M085)         </wsse:TransformationParameters>
4130 (M086)       </ds:Transform>
4131 (M087)     </ds:Transforms>
4132 (M088)     <ds:DigestMethod
4133 (M089)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
4134 (M090)     <ds:DigestValue>...</ds:DigestValue>
4135 (M091)   </ds:Reference>

```

```

4136 (M092) <ds:Reference URI="#MsgBody">
4137 (M093) <ds:DigestMethod
4138 (M094) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4139 (M095) <ds:DigestValue>...</ds:DigestValue>
4140 (M096) </ds:Reference>
4141 (M097) <ds:Reference URI="#timestamp">
4142 (M098) <ds:DigestMethod
4143 (M099) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4144 (M100) <ds:DigestValue>...</ds:DigestValue>
4145 (M101) </ds:Reference>
4146 (M102) </ds:SignedInfo>
4147 (M103) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
4148 (M104) <ds:KeyInfo>
4149 (M105) <wsse:SecurityTokenReference wsu:id="STR2">
4150 (M106) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
4151 (M107) </wsse:SecurityTokenReference>
4152 (M108) </ds:KeyInfo>
4153 (M109) </ds:Signature>
4154 (M110) <ds:Signature wsu:Id="endorsing-signature">
4155 (M111) <ds:SignedInfo>
4156 (M112) ...
4157 (M113) <ds:Reference URI="#message-signature">
4158 (M114) ...
4159 (M115) </ds:Reference>
4160 (M116) </ds:SignedInfo>
4161 (M117) <ds:SignatureValue>Bu ...=</ds:SignatureValue>
4162 (M118) <ds:KeyInfo>
4163 (M119) <wsse:SecurityTokenReference >
4164 (M120) <wsse:Reference URI="#attesterCert" ValueType="...#X509v3"/>
4165 (M121) </wsse:SecurityTokenReference>
4166 (M122) </ds:KeyInfo>
4167 (M123) </ds:Signature>
4168 (M124) </wsse:Security>
4169 (M125) </S12:Header>
4170 (M126) <S12:Body wsu:Id="MsgBody">
4171 (M127) <xenc:EncryptedData Id="encBody" Type="...#Content"
4172 MimeType="text/xml" Encoding="UTF-8" >
4173 (M128) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
4174 (M129) <ds:KeyInfo >
4175 (M130) <wsse:SecurityTokenReference>
4176 (M131) <wsse:Reference URI="#enckey" ValueType=".../EncryptedKey"/>
4177 (M132) </wsse:SecurityTokenReference>
4178 (M133) </ds:KeyInfo>
4179 (M134) <xenc:CipherData>
4180 (M135) <xenc:CipherValue>70...=</xenc:CipherValue>
4181 (M136) </xenc:CipherData>
4182 (M137) </xenc:EncryptedData>
4183 (M138) </S12:Body>
4184 (M139) </S12:Envelope>

```

4185

4186 The message above contains a request compliant with the policy described in (P001)-(P072), which
4187 includes the endpoint policy and the input policy.

4188 Lines (M016)-(M028) contain an xenc:EncryptedKey element that contains an Initiator-generated
4189 symmetric signing and encryption key, K, that can be used to decrypt the xenc:EncryptedData contained
4190 in the SOAP S12:Body (M0126) as indicated by the wsse:SecurityTokenReference (M0130)-(M0132) that
4191 uses a direct reference to the Id of the xenc:EncryptedKey, "encKey" on lines (M0131) and (M016).

4192 The xenc:EncryptedKey contains a dsig KeyInfo (M020)-(M024) reference to the Thumbprint of the
4193 Recipient's public X509 certificate (M022). This complies with the policy (P068) that requires the sp:Body
4194 to of the input message to be encrypted. It also complies with the policy (P013) to protect using
4195 encryption based on X509 token and to refer to it by Thumbprint (P014).

4196 Lines (M029)-(M031) contain an xenc:ReferenceList referring to the xenc:EncryptedData in the S12:Body
4197 (M0127)-(M0137).

4198 Lines (M032)-(M059) contain the saml:Assertion as a SignedSupportingToken as required by the
4199 endpoint policy (P035). A saml:Assertion used as a SignedSupportingToken uses the “sender-vouches”
4200 ConfirmationMethod (M049)-(M051) as described in the introductory section 2.3.

4201 Lines (M060)-(M065) contain a WS-Security wsse:SecurityTokenReference that has a KeyIdentifier of
4202 ValueType “...1.0#SAMLAssertionID”, which indicates a SAML 1.1 Assertion as described in the WS-
4203 Security 1.1 [[SAML1120_TOKEN_PROFILE](#)].

4204 Lines (M066)-(M071) contain a wsse:BinarySecurityToken that contains the Initiator’s X509 certificate for
4205 use as an EndorsingSupportingToken as required by line (P042) of the sp:EndorsingSupportingTokens
4206 assertion (P040)-(P048).

4207 Lines (M072)-(M0109) contain the message signature. The dsig Signature covers the following:

- 4208 ➤ The SAML Assertion using the ds:Reference (M078)-(M091), which uses the WS-Security STR-
4209 Transform technique to refer to the SAML Assertion indirectly through the
4210 wsse:SecurityTokenReference (M060) described above. This signature is required by the
4211 sp:SignedSupportingTokens assertion (P031)-(P039).
- 4212 ➤ The message sp:Body (M0126)-(M0138) using the ds:Reference (M092)-(M096) as required by
4213 the input message policy (P065).
- 4214 ➤ The message wsu:Timestamp (M013)-(M015) using the ds:Reference (M097)-(M0101) as
4215 required by the endpoint policy (P027).

4216 The key used to sign the message signature is referenced in the dsig KeyInfo (M0104)-(M0108), which
4217 contains a SecurityTokenReference with a direct URI reference to the xenc:EncryptedKey, “encKey”,
4218 which contains the Initiator-generated signing and encryption key, K, as described above.

4219 Lines (M0110)-(M0123) contain the endorsing signature. The dsig endorsing Signature covers the
4220 following:

- 4221 ➤ The message Signature (M072)-(M0109) using the ds:Reference (M0113)-(M0115), which is a
4222 direct URI reference to the Id “message-signature” on line (M072).

4223 This signature is required by the policy EndorsingSupportingTokens assertion (P040)-(P048) to be an
4224 X509 certificate (P044). The dsig KeyInfo (M0118)-(M0122) contains a WS-Security
4225 SecurityTokenReference with a direct URI to the Initiator’s X509 certificate on line (M066)-(M071) with Id
4226 = “attesterCert”.

4227 Lines (M0127)-(M0137) contain the xenc:EncryptedData, which contains the SOAP S12:Body message
4228 payload that is encrypted using the encryption key, K, contained in the xenc:EncryptedKey CipherValue
4229 (M026), which can only be decrypted using the Recipient’s X509 certificate referred to by
4230 ThumbprintSHA1 on line (M022).

4231 **2.3.2.5 (WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt**

4232 This scenario is based on WS-SX Interop Scenarios Phase 2 (October 31, 2006) [[WSSX-WSTR-WSSC-](#)
4233 [INTEROP](#)] Scenario 5 (Client and STS: Mutual Certificate WSS1.1 (section 3.5 of interop ref), Client and
4234 Service: Issued SAML 1.1 Token for Certificate WSS1.1 (section 4.3 of interop ref)).

4235 In this scenario, the service specifies that the client must obtain an IssuedToken from a designated
4236 Security Token Service (STS), which must be a SAML 1.1 Assertion. The Assertion contains an
4237 ephemeral key K2 in an EncryptedKey element encrypted using service’s certificate. The client also
4238 obtains the same ephemeral key K2 from the RequestedProofToken returned by the STS. The body of
4239 the message from the client to the service is signed using DKT1(K2), encrypted using DKT2(K2), and
4240 endorsed using DKT3(K2), which are keys the client derives from K2 using the algorithm specified in
4241 section 7 of [[WS-SecureConversation](#)]. The response from the service is also signed using derived keys.
4242 In a simpler alternative, ephemeral key K itself could be used for message protection.

4243 Note: In this scenario, in terms of Figure 1 [[Figure01](#)], the STS (Issuing Authority) is the
4244 Issuer of the SAML 1.1 holder-of-key Assertion that contains the ephemeral symmetric
4245 key K2. The service (combined Recipient/RelyingParty) can trust the client (combined

4246 Initiator/Requestor) that uses the ephemeral symmetric key K2 obtained from the
4247 RequestedProofToken, because the same key gets delivered to the service in the SAML
4248 1.1 holder-of-key Assertion, which is signed by the trusted Authority.

4249 This scenario is a 2-step sequence from a WS-SP perspective. These 2 steps will be described at a high
4250 level first in order to explain the context for the policies and messages that follow.

4251 **In step 1** the following takes place:

- 4252 • the client accesses the service's WS-SP policy (P001 -> P116 below), and determines that it
4253 needs to use an IssuedToken from a Security Token Service (STS), specified in the policy.
4254 This IssuedToken will serve as the basis of trust by the service. Effectively, the service only trusts
4255 the client because the client is able to obtain the IssuedToken from the STS.

4256 To complete step 1, the client will then do the following:

- 4257 • the client will access the STS and process the STS policy (PSTS-001 -> PSTS-098 below)
- 4258 • based on the STS policy, the client will send a request to the STS for an IssuedToken (MSTS-001
4259 -> MSTS-0231 below)
- 4260 • the STS will send a response to the client containing the IssuedToken, which in this example is a
4261 SAML 1.1 holder-of-key Assertion (RSTS-001 -> RSTS-0263 below)

4262 **In step 2** the following takes place:

- 4263 • because the client now has the IssuedToken it is now able fulfill the requirements of the service's
4264 WS-SP policy (P001-P116 below) that it accessed in step 1
- 4265 • based on the service's policy the clients sends a request to the service (M001-M0229 below)
- 4266 • the service will send a response to the client containing the requested resource data (R001-
4267 R0153 below)

4268 As an aid to understanding the security properties of the policies and messages in this example, the
4269 following is a list of identifiers used in the text descriptions to reference the cryptographic keys that are
4270 called for in the policies and used in the messages in this example:

- 4271 ➤ **X509T1**: Client's (Requestor/Initiator) X509 certificate used for authentication to the STS.
- 4272 ➤ **X509T2**: STS' (Issuing Authority) X509 certificate used to encrypt keys sent to STS, used to
4273 authenticate STS to Service.
- 4274 ➤ **X509T3**: Service's (Recipient/RelyingParty/ValidatingAuthority) X509 certificate used to encrypt
4275 keys sent to Service.
- 4276 ➤ **K1**: Client-generated ephemeral symmetric key for crypto communication to the STS.
- 4277 ➤ **K2**: Client-generated ephemeral symmetric key for crypto communication between the Client and
4278 the Service.
- 4279 ➤ **K3**: STS-generated proof key for use by Client to authenticate with Service via saml:Assertion.
- 4280 ➤ **DKT1(K2)**: Client-generated derived key used for signing requests to the Service.
- 4281 ➤ **DKT2(K2)**: Client-generated derived key used for encrypting requests to the Service.
- 4282 ➤ **DKT3(K3)**: Client-generated derived key used for endorsing signed requests to the Service.
- 4283 ➤ **DKT4(K2)**: Service-generated derived key used for encrypting responses to the Client.
- 4284 ➤ **DKT5(K2)**: Service-generated derived key used for signing responses to the Client.

4285

4286 Here is the WS-SP Policy that the Service presents to a Client:

4287

```
4288 (P001) <wsp:Policy wsu:Id="Service5-Policy"  
4289 (P002)  
4290 (P003) xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4291 (P004) xmlns:sp"..."  
4292 (P005)
```

```

4293 (P006)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4294 wssecurity-utility-1.0.xsd"
4295 (P007)      >
4296 (P008)      <wsp:ExactlyOne>
4297 (P009)      <wsp:All>
4298 (P010)      <sp:SymmetricBinding>
4299 (P011)      <wsp:Policy>
4300 (P012)      <sp:ProtectionToken>
4301 (P013)      <wsp:Policy>
4302 (P014)      <sp:X509Token IncludeToken=
4303 (P015)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4304 /Never">
4305 (P016)      <wsp:Policy>
4306 (P017)      <sp:RequireDerivedKeys/>
4307 (P018)      <sp:RequireThumbprintReference/>
4308 (P019)      <sp:WssX509V3Token10/>
4309 (P020)      </wsp:Policy>
4310 (P021)      </sp:X509Token>
4311 (P022)      </wsp:Policy>
4312 (P023)      </sp:ProtectionToken>
4313 (P024)      <sp:AlgorithmSuite>
4314 (P025)      <wsp:Policy>
4315 (P026)      <sp:Basic256/>
4316 (P027)      </wsp:Policy>
4317 (P028)      </sp:AlgorithmSuite>
4318 (P029)      <sp:Layout>
4319 (P030)      <wsp:Policy>
4320 (P031)      <sp:Strict/>
4321 (P032)      </wsp:Policy>
4322 (P033)      </sp:Layout>
4323 (P034)      <sp:IncludeTimestamp/>
4324 (P035)      <sp:OnlySignEntireHeadersAndBody/>
4325 (P036)      </wsp:Policy>
4326 (P037)      </sp:SymmetricBinding>
4327 (P038)      <sp:EndorsingSupportingTokens>
4328 (P039)      <wsp:Policy>
4329 (P040)      <sp:IssuedToken IncludeToken=
4330 (P041)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4331 /AlwaysToRecipient">
4332 (P042)      <sp:Issuer>
4333 (P043)      <a:Address>http://example.com/STS</a:Address>
4334 (P044)      </sp:Issuer>
4335 (P045)      <sp:RequestSecurityTokenTemplate>
4336 (P046)      <t:TokenType
4337 (P047)      >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4338 1.1#SAMLV1.1</t:TokenType>
4339 (P048)      <t:KeyType
4340 (P049)      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:KeyTy
4341 pe>
4342 (P050)      <t:KeySize>256</t:KeySize>
4343 (P051)      <t:CanonicalizationAlgorithm
4344 (P052)      >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorithm>
4345 (P053)      <t:EncryptionAlgorithm
4346 (P054)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4347 (P055)      <t:EncryptWith
4348 (P056)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4349 (P057)      <t:SignWith
4350 (P058)      >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4351 (P059)      </sp:RequestSecurityTokenTemplate>
4352 (P060)      <wsp:Policy>
4353 (P061)      <sp:RequireDerivedKeys/>
4354 (P062)      <sp:RequireInternalReference/>
4355 (P063)      </wsp:Policy>
4356 (P064)      </sp:IssuedToken>

```

```

4357 (P065)         </wsp:Policy>
4358 (P066)         </sp:EndorsingSupportingTokens>
4359 (P067)         <sp:Wss11>
4360 (P068)         <wsp:Policy>
4361 (P069)         <sp:MustSupportRefKeyIdentifier/>
4362 (P070)         <sp:MustSupportRefIssuerSerial/>
4363 (P071)         <sp:MustSupportRefThumbprint/>
4364 (P072)         <sp:MustSupportRefEncryptedKey/>
4365 (P073)         <sp:RequireSignatureConfirmation/>
4366 (P074)         </wsp:Policy>
4367 (P075)         </sp:Wss11>
4368 (P076)         <sp:Trust13>
4369 (P077)         <wsp:Policy>
4370 (P078)         <sp:MustSupportIssuedTokens/>
4371 (P079)         <sp:RequireClientEntropy/>
4372 (P080)         <sp:RequireServerEntropy/>
4373 (P081)         </wsp:Policy>
4374 (P082)         </sp:Trust13>
4375 (P083)         </wsp:All>
4376 (P084)         </wsp:ExactlyOne>
4377 (P085)         </wsp:Policy>
4378
4379 (P086)         <wsp:Policy wsu:Id="InOut-Policy"
4380 (P087)
4381 (P088)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4382 (P089)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4383 (P090)
4384 (P091)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4385 (P092)         wssecurity-utility-1.0.xsd">
4386 (P092)         <wsp:ExactlyOne>
4387 (P093)         <wsp:All>
4388 (P094)         <sp:SignedParts>
4389 (P095)         <sp:Body/>
4390 (P096)         <sp:Header Name="To"
4391 (P097)           Namespace="http://www.w3.org/2005/08/addressing"/>
4392 (P098)         <sp:Header Name="From"
4393 (P099)           Namespace="http://www.w3.org/2005/08/addressing"/>
4394 (P100)         <sp:Header Name="FaultTo"
4395 (P101)           Namespace="http://www.w3.org/2005/08/addressing"/>
4396 (P102)         <sp:Header Name="ReplyTo"
4397 (P103)           Namespace="http://www.w3.org/2005/08/addressing"/>
4398 (P104)         <sp:Header Name="MessageID"
4399 (P105)           Namespace="http://www.w3.org/2005/08/addressing"/>
4400 (P106)         <sp:Header Name="RelatesTo"
4401 (P107)           Namespace="http://www.w3.org/2005/08/addressing"/>
4402 (P108)         <sp:Header Name="Action"
4403 (P109)           Namespace="http://www.w3.org/2005/08/addressing"/>
4404 (P110)         </sp:SignedParts>
4405 (P111)         <sp:EncryptedParts>
4406 (P112)         <sp:Body/>
4407 (P113)         </sp:EncryptedParts>
4408 (P114)         </wsp:All>
4409 (P115)         </wsp:ExactlyOne>
4410 (P116)         </wsp:Policy>

```

4411

4412 When a Client encounters the above Service Policy, it determines that it must obtain an IssuedToken
4413 from the designated Issuer. After obtaining the IssuedToken, the client may send the request in
4414 accordance with the rest of the policy. Details of the Service Policy follow. The STS Policy its details
4415 follow after the Service Policy.

4416 Lines (P001)-(P085) above contain the Service Endpoint wsp:Policy, which contains a wsp:All assertion
4417 that requires all 4 of its contained assertions to be complied with: sp:SymmetricBinding (P010)-(P037),
4418 sp:EndorsingSupportingTokens (P038)-(P066), sp:Wss11 (P067)-(P075), and sp:Trust13 (P076)-(P082).

4419 Lines (P010)-(P037) contain the **sp:SymmetricBinding assertion** that requires that derived keys (DKT1,
4420 DKT2, DKT3) be used to protect the message (P017) and that the ephemeral key (K2) used to derive
4421 these keys be encrypted using the service's X509 certificate (X509T3) as specified by the sp:X509Token
4422 assertion (P014)-(P021), which also indicates that the X509 token, itself should not be sent (P015), but
4423 that a Thumbprint reference (P018) to it be sent. Finally, the sp:SymmetricBinding specifies that the
4424 sp:Basic256 sp:AlgorithmSuite be used (P024)-(P028), that sp:Strict sp:Layout be used (P029)-(P033),
4425 that an sp:Timestamp be included (P034), and that only the complete message Body and Headers be
4426 signed (P035), where the Headers part means that only direct child elements of the WS-Security SOAP
4427 header element be signed.

4428 Lines (P038)-(P066) contain the **sp:EndorsingSupportingTokens assertion** that an sp:IssuedToken
4429 (P040)-(P064) be used to sign the message signature and that the IssuedToken must be included with
4430 the request (P040)-(P041). Lines (P042)-(P044) specify the address of the SecurityTokenService (STS)
4431 from which the IssuedToken must be obtained. Lines (P045)-(P059) contain an
4432 sp:RequestSecurityTokenTemplate (P046)-(P058) which contains explicit WS-Trust elements that the
4433 client should directly copy to a t:SecondaryParameters element to include with the WS-Trust
4434 t:RequestSecurityToken to the STS to obtain the sp:IssuedToken. Of particular interest here is that the
4435 t:TokenType (P046)-(P047) requested is a SAML 1.1 Assertion, which will be used to contain the
4436 ephemeral symmetric key (K2) (P048)-(P049). K2 will be used for communication between the Client and
4437 the Service. K2 will be encrypted by the STS using the Service's X509 certificate (X509T3). The Client is
4438 also informed by the IssuedToken assertion that the IssuedToken may only be referenced internally
4439 within the message (P062) and that the ephemeral key (K2) associated with the IssuedToken be used by
4440 the Client to derive the keys (DKT1(K2), DKT2(K2), DKT3(K2)) used in the Client's request to the Service.

4441 Lines (P067)-(P075) contain the **sp:Wss11 assertion** that indicates that WS-Security 1.1 will be used,
4442 which includes Wss11-only features such as Thumbprint (P073), EncryptedKey (P074), and
4443 SignatureConfirmation (P075).

4444 Lines (P076)-(P082) contain the **sp:Trust13 assertion** that indicates the Client should expect to use WS-
4445 Trust 1.3 ([WSTRUST](#)) to obtain the IssuedToken from the STS.

4446 Lines (P086)-(P0116) contain the **operation input and output policies** that the client should use to
4447 determine what parts of the messages are to be signed (P094)-(P0110) and encrypted (P0111)-(P0113).

4448

4449 Here is the WS-SP Policy that the STS presents to a Client:

```
4450 (PSTS-001) <wsp:Policy wsu:Id="STS5-Policy"  
4451 (PSTS-002)  
4452 (PSTS-003)     xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4453 (PSTS-004)     xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"  
4454 (PSTS-005)  
4455 (PSTS-006)     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-  
4456 wss-wssecurity-utility-1.0.xsd"  
4457 (PSTS-007) >  
4458 (PSTS-008)     <wsp:ExactlyOne>  
4459 (PSTS-009)         <wsp>All>  
4460 (PSTS-010)             <sp:SymmetricBinding>  
4461 (PSTS-011)                 <wsp:Policy>  
4462 (PSTS-012)                     <sp:ProtectionToken>  
4463 (PSTS-013)                         <wsp:Policy>  
4464 (PSTS-014)                             <sp:X509Token IncludeToken=  
4465 (PSTS-015) "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo  
4466 ken/Never">  
4467 (PSTS-016)                                 <wsp:Policy>  
4468 (PSTS-017)                                     <sp:RequireThumbprintReference/>  
4469 (PSTS-018)                                     <sp:WssX509V3Token10/>  
4470 (PSTS-019)                                 </wsp:Policy>  
4471 (PSTS-020)                             </sp:X509Token>  
4472 (PSTS-021)                         </wsp:Policy>  
4473 (PSTS-022)                     </sp:ProtectionToken>  
4474 (PSTS-023)                 <sp:AlgorithmSuite>  
4475 (PSTS-024)             <wsp:Policy>  
4476 (PSTS-025)                 <sp:Basic256/>
```

```

4477 (PSTS-026)         </wsp:Policy>
4478 (PSTS-027)         </sp:AlgorithmSuite>
4479 (PSTS-028)         <sp:Layout>
4480 (PSTS-029)         <wsp:Policy>
4481 (PSTS-030)         <sp:Strict/>
4482 (PSTS-031)         </wsp:Policy>
4483 (PSTS-032)         </sp:Layout>
4484 (PSTS-033)         <sp:IncludeTimestamp/>
4485 (PSTS-034)         <sp:OnlySignEntireHeadersAndBody/>
4486 (PSTS-035)         </wsp:Policy>
4487 (PSTS-036)         </sp:SymmetricBinding>
4488 (PSTS-037)         <sp:EndorsingSupportingTokens>
4489 (PSTS-038)         <wsp:Policy>
4490 (PSTS-039)         <sp:X509Token IncludeToken=
4491 (PSTS-040)         "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4492 (PSTS-040)         ken/AlwaysToRecipient">
4493 (PSTS-041)         <wsp:Policy>
4494 (PSTS-042)         <sp:RequireThumbprintReference/>
4495 (PSTS-043)         <sp:WssX509V3Token10/>
4496 (PSTS-044)         </wsp:Policy>
4497 (PSTS-045)         </sp:X509Token>
4498 (PSTS-046)         </wsp:Policy>
4499 (PSTS-047)         </sp:EndorsingSupportingTokens>
4500 (PSTS-048)         <sp:Wss11>
4501 (PSTS-049)         <wsp:Policy>
4502 (PSTS-050)         <sp:MustSupportRefKeyIdentifier/>
4503 (PSTS-051)         <sp:MustSupportRefIssuerSerial/>
4504 (PSTS-052)         <sp:MustSupportRefThumbprint/>
4505 (PSTS-053)         <sp:MustSupportRefEncryptedKey/>
4506 (PSTS-054)         <sp:RequireSignatureConfirmation/>
4507 (PSTS-055)         </wsp:Policy>
4508 (PSTS-056)         </sp:Wss11>
4509 (PSTS-057)         <sp:Trust13>
4510 (PSTS-058)         <wsp:Policy>
4511 (PSTS-059)         <sp:MustSupportIssuedTokens/>
4512 (PSTS-060)         <sp:RequireClientEntropy/>
4513 (PSTS-061)         <sp:RequireServerEntropy/>
4514 (PSTS-062)         </wsp:Policy>
4515 (PSTS-063)         </sp:Trust13>
4516 (PSTS-064)         </wsp:All>
4517 (PSTS-065)         </wsp:ExactlyOne>
4518 (PSTS-066)         </wsp:Policy>
4519
4520 (PSTS-067)         <wsp:Policy wsu:Id="InOut-Policy"
4521 (PSTS-068)
4522 (PSTS-069)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4523 (PSTS-070)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4524 (PSTS-071)
4525 (PSTS-072)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4526 (PSTS-072)         wss-wssecurity-utility-1.0.xsd"
4527 (PSTS-073)         >
4528 (PSTS-074)         <wsp:ExactlyOne>
4529 (PSTS-075)         <wsp:All>
4530 (PSTS-076)         <sp:SignedParts>
4531 (PSTS-077)         <sp:Body/>
4532 (PSTS-078)         <sp:Header Name="To"
4533 (PSTS-079)         Namespace="http://www.w3.org/2005/08/addressing"/>
4534 (PSTS-080)         <sp:Header Name="From"
4535 (PSTS-081)         Namespace="http://www.w3.org/2005/08/addressing"/>
4536 (PSTS-082)         <sp:Header Name="FaultTo"
4537 (PSTS-083)         Namespace="http://www.w3.org/2005/08/addressing"/>
4538 (PSTS-084)         <sp:Header Name="ReplyTo"
4539 (PSTS-085)         Namespace="http://www.w3.org/2005/08/addressing"/>
4540 (PSTS-086)         <sp:Header Name="MessageID"

```

```

4541 (PSTS-087)      Namespace="http://www.w3.org/2005/08/addressing"/>
4542 (PSTS-088)      <sp:Header Name="RelatesTo"
4543 (PSTS-089)      Namespace="http://www.w3.org/2005/08/addressing"/>
4544 (PSTS-090)      <sp:Header Name="Action"
4545 (PSTS-091)      Namespace="http://www.w3.org/2005/08/addressing"/>
4546 (PSTS-092)      </sp:SignedParts>
4547 (PSTS-093)      <sp:EncryptedParts>
4548 (PSTS-094)      <sp:Body/>
4549 (PSTS-095)      </sp:EncryptedParts>
4550 (PSTS-096)      </wsp:All>
4551 (PSTS-097)      </wsp:ExactlyOne>
4552 (PSTS-098)      </wsp:Policy>

```

4553 Above is the STS Policy that the Client will encounter when obtaining the IssuedToken required by the
4554 Service Policy. This policy is quite similar in detail to the Service Policy and therefore only the differences
4555 that are noteworthy will be discussed in the details below.

4556 Similar to the Service Policy, the STS endpoint Policy contains 4 assertions to be complied with:
4557 sp:SymmetricBinding (PSTS-010)-(PSTS-036), sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047),
4558 sp:Wss11 (PSTS-048)-(PSTS-056), and sp:Trust13 (PSTS-057)-(PSTS-063).

4559 Lines (PSTS-010)-(PSTS-036) contain the **sp:SymmetricBinding assertion**, where the main difference
4560 from the previous policy is that here the sp:ProtectionToken is an sp:X509Token that will be used to
4561 encrypt the ephemeral client-generated key (K1) that will be used for Client-STS communication. Derived
4562 keys will not be required in this communication.

4563 Lines (PSTS-037)-(PSTS-047) contain the **sp:EndorsingSupportingTokens assertion**, which in this
4564 case contains an sp:X509Token assertion (PSTS-039)-(PSTS-045) that requires the Client to include
4565 (PSTS-040) its X509 certificate, which must be used to sign the message signature. The STS uses this
4566 mechanism to authenticate the Client.

4567 The **sp:Wss11 assertion** (PSTS-048)-(PSTS-056), **sp:Trust13 assertion** (PSTS-057)-(PSTS-063) and
4568 the **operation input and output policies** (PSTS-067)-(PSTS-098) are the same as those described for
4569 the Service policy above (P067)-(P0116).

4570

4571 Below are included sample messages that comply with the above policies. The messages are presented
4572 in the same order that they would be used in a real scenario and therefore the Client-STS request
4573 (MSTS-001)-(MSTS-0231) and response (RSTS-001)-(RSTS-0263) are presented first, which are then
4574 followed by the Client-Service request (M001-M229) and response (R001)-(R153).

4575 Here is an example Client request to the STS:

```

4576 (MSTS-001)      <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
4577 (MSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
4578 (MSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
4579 (MSTS-004)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4580 (MSTS-005)      wssecurity-secext-1.0.xsd"
4581 (MSTS-005)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4582 (MSTS-006)      wssecurity-utility-1.0.xsd" >
4583 (MSTS-006)      <s:Header>
4584 (MSTS-007)      <a:Action s:mustUnderstand="1" u:Id="_3">
4585 (MSTS-008)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
4586 (MSTS-009)      </a:Action>
4587 (MSTS-010)      <a:MessageID u:Id="_4">
4588 (MSTS-011)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4589 (MSTS-012)      </a:MessageID>
4590 (MSTS-013)      <a:ReplyTo u:Id="_5">
4591 (MSTS-014)      <a:Address>
4592 (MSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4593 (MSTS-016)      </a:Address>
4594 (MSTS-017)      </a:ReplyTo>
4595 (MSTS-018)      <a:To s:mustUnderstand="1" u:Id="_6">
4596 (MSTS-019)      http://server.example.com/STS/Scenarios5-6
4597 (MSTS-020)      </a:To>
4598 (MSTS-021)      <o:Security s:mustUnderstand="1">

```

```

4599 (MSTS-022) <u:Timestamp
4600 (MSTS-023)   u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4601 (MSTS-024)   <u:Created>2005-10-25T00:47:36.144Z</u:Created>
4602 (MSTS-025)   <u:Expires>2005-10-25T00:52:36.144Z</u:Expires>
4603 (MSTS-026)   </u:Timestamp>
4604 (MSTS-027)   <e:EncryptedKey
4605 (MSTS-028)     Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-9">
4606 (MSTS-029)     <e:EncryptionMethod Algorithm=
4607 (MSTS-030)       "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
4608 (MSTS-031)     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
4609 (MSTS-032)       <o:SecurityTokenReference>
4610 (MSTS-033)         <o:KeyIdentifier ValueType=
4611 (MSTS-034)       "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4612 (MSTS-035)       1.1.xsd#ThumbprintSHA1">
4613 (MSTS-036)         W+rgYBmLmVEG//scD7Vo8Kq5G7I=
4614 (MSTS-037)       </o:KeyIdentifier>
4615 (MSTS-038)     </o:SecurityTokenReference>
4616 (MSTS-039)   </KeyInfo>
4617 (MSTS-040)   <e:CipherData>
4618 (MSTS-041)     <e:CipherValue>
4619 (MSTS-042)       <!--base64 encoded cipher-->
4620 (MSTS-043)     </e:CipherValue>
4621 (MSTS-044)   </e:CipherData>
4622 (MSTS-045)   <e:ReferenceList>
4623 (MSTS-046)     <e:DataReference URI="#_2"/>
4624 (MSTS-047)   </e:ReferenceList>
4625 (MSTS-048) </e:EncryptedKey>
4626 (MSTS-049) <o:BinarySecurityToken
4627 (MSTS-050)   u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"
4628 (MSTS-051)   ValueType=
4629 (MSTS-052)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4630 (MSTS-053)   profile-1.0#X509v3"
4631 (MSTS-054)   EncodingType=
4632 (MSTS-055)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
4633 (MSTS-056)   message-security-1.0#Base64Binary">
4634 (MSTS-057)   MIIDDDCCafSgAwIBAgIQM6YEf7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4w
4635 (MSTS-058)   DAYDVQKDAVPQVNJUzEeMBwGAlUEAwVT0FTSVMgSW50ZXJvcCBUZXN0IENBMB4XDTA1
4636 (MSTS-059)   MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVowQjEOMAwGAlUECgwFT0FTSVMxIDAeBgNV
4637 (MSTS-060)   BAsMF09BU01TEIeludGVyb3AgVGZzdCBDZXJ0MQ4wDAYDVQQDDAVBbG1jZTCBnzANBggkq
4638 (MSTS-061)   hkiG9w0BAQEFAAOBjQAwYkCgYEAoqi99By1VYo0aHrkKcNT4DkIgL/SgahbeKdGhrb
4639 (MSTS-062)   u3K2XG7arfD9tqIBIKMfrx4Gp90NJa85AV1yiNsEyvq+mUnMpNcKnLXLOjkTmMcqDYbb
4640 (MSTS-063)   kehJlXPnaWLzve+mW0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz+EzsCAwEA
4641 (MSTS-064)   AaOBkzCBkDAJBgNVHRMEAjAAMDGAlUdHwQsMCoKKImhiRodHRWoi8vaW50ZXJvcC5i
4642 (MSTS-065)   YnRlc3QubmV0L2Nybc9jYs5jcmwwDgYDVR0PAQH/BAQDAgSwMB0GAlUdDgQWBQK410T
4643 (MSTS-066)   UHZ1QV3V2Qt1LNDm+PoxiDafBgNVHSMEGDAWgBTAnSj8wes1oR3WqqqgHBpNwkkPdZAN
4644 (MSTS-067)   BgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW+6yrLXyU1P2xJbEkohXHI5OWwKWleOb9h1kh
4645 (MSTS-068)   WntUalfcFOJAgUyH30TPhldzx1+vK2LPzhoUFKYHE1IyQvokBN2JjFO64BquKCKnZhl
4646 (MSTS-069)   dLRPxBghkTdxQgdf5rCK/wh3xVsZCNTfuMnmlAM6LOAg8QduDah3WFZpEA0s2nwQaCNQ
4647 (MSTS-070)   TNmjJC8tav1CBR6+E5FAMwPXP7pJxn9Fw9OXRYqbRA4v2y7YpbGkG2GI9UvOHw6SGvff4
4648 (MSTS-071)   FRStHMMO35YbpikGsLix3vAsXWwi4rwfVOYzQK00FPNi9RMCUdSH06m9uLWckiCxjos0
4649 (MSTS-072)   FQODZE9l4ATGy9s9hNVwryOJTW==
4650 (MSTS-073)   </o:BinarySecurityToken>
4651 (MSTS-074)   <Signature Id="_0" xmlns="http://www.w3.org/2000/09/xmldsig#">
4652 (MSTS-075)     <SignedInfo>
4653 (MSTS-076)       <CanonicalizationMethod Algorithm=
4654 (MSTS-077)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4655 (MSTS-078)       <SignatureMethod Algorithm=
4656 (MSTS-079)         "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
4657 (MSTS-080)       <Reference URI="#_1">
4658 (MSTS-081)         <Transforms>
4659 (MSTS-082)           <Transform Algorithm=
4660 (MSTS-083)             "http://www.w3.org/2001/10/xml-exc-c14n#" />
4661 (MSTS-084)         </Transforms>
4662 (MSTS-085)       </Reference>

```



```

4663 (MSTS-083) <DigestMethod Algorithm=
4664 (MSTS-084) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4665 (MSTS-085) <DigestValue>VX1fCPwCzVsSc1hZf0BSbCgW2hM=</DigestValue>
4666 (MSTS-086) </Reference>
4667 (MSTS-087) <Reference URI="#_3">
4668 (MSTS-088) <Transforms>
4669 (MSTS-089) <Transform Algorithm=
4670 (MSTS-090) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4671 (MSTS-091) </Transforms>
4672 (MSTS-092) <DigestMethod Algorithm=
4673 (MSTS-093) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4674 (MSTS-094) <DigestValue>FwiFAUuqNDo9SDkk5A28Mg7Pa8Q=</DigestValue>
4675 (MSTS-095) </Reference>
4676 (MSTS-096) <Reference URI="#_4">
4677 (MSTS-097) <Transforms>
4678 (MSTS-098) <Transform Algorithm=
4679 (MSTS-099) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4680 (MSTS-100) </Transforms>
4681 (MSTS-101) <DigestMethod Algorithm=
4682 (MSTS-102) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4683 (MSTS-103) <DigestValue>oM59PsOTpMrDdOcwXYQzjVU10xw=</DigestValue>
4684 (MSTS-104) </Reference>
4685 (MSTS-105) <Reference URI="#_5">
4686 (MSTS-106) <Transforms>
4687 (MSTS-107) <Transform Algorithm=
4688 (MSTS-108) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4689 (MSTS-109) </Transforms>
4690 (MSTS-110) <DigestMethod Algorithm=
4691 (MSTS-111) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4692 (MSTS-112) <DigestValue>KIK3vklFN1QmMdQkplq2azfzrzg=</DigestValue>
4693 (MSTS-113) </Reference>
4694 (MSTS-114) <Reference URI="#_6">
4695 (MSTS-115) <Transforms>
4696 (MSTS-116) <Transform Algorithm=
4697 (MSTS-117) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4698 (MSTS-118) </Transforms>
4699 (MSTS-119) <DigestMethod Algorithm=
4700 (MSTS-120) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4701 (MSTS-121) <DigestValue>RJEE3hrcyCD6PzFJo6fyut6biVg=</DigestValue>
4702 (MSTS-122) </Reference>
4703 (MSTS-123) <Reference
4704 (MSTS-124) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4705 (MSTS-125) <Transforms>
4706 (MSTS-126) <Transform Algorithm=
4707 (MSTS-127) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4708 (MSTS-128) </Transforms>
4709 (MSTS-129) <DigestMethod Algorithm=
4710 (MSTS-130) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4711 (MSTS-131) <DigestValue>zQdN5XpejfqXn0Wko0m5lZYiasE=</DigestValue>
4712 (MSTS-132) </Reference>
4713 (MSTS-133) </SignedInfo>
4714 (MSTS-134) <SignatureValue
4715 (MSTS-135) >iHGJ+xv2VZTjM1Rc7AQJrwLY/aM=</SignatureValue>
4716 (MSTS-136) <KeyInfo>
4717 (MSTS-137) <o:SecurityTokenReference>
4718 (MSTS-138) <o:Reference
4719 (MSTS-139) ValueType=
4720 (MSTS-140) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4721 (MSTS-141) 1.1.xsd#EncryptedKey"
4722 (MSTS-142) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-9"/>
4723 (MSTS-143) </o:SecurityTokenReference>
4724 (MSTS-144) </KeyInfo>
4725 (MSTS-145) </Signature>
4726 (MSTS-0145) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

```

```

4727 (MSTS-0146) <SignedInfo>
4728 (MSTS-0147)   <CanonicalizationMethod Algorithm=
4729 (MSTS-0148)     "http://www.w3.org/2001/10/xml-exc-c14n#" />
4730 (MSTS-0149)   <SignatureMethod Algorithm=
4731 (MSTS-0150)     "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4732 (MSTS-0151)   <Reference URI="#_0">
4733 (MSTS-0152)     <Transforms>
4734 (MSTS-0153)       <Transform Algorithm=
4735 (MSTS-0154)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4736 (MSTS-0155)       </Transforms>
4737 (MSTS-0156)     <DigestMethod Algorithm=
4738 (MSTS-0157)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4739 (MSTS-0158)     <DigestValue>UZKtShk8q6iu9WR5uQZp04iAitg=</DigestValue>
4740 (MSTS-0159)   </Reference>
4741 (MSTS-0160)   </SignedInfo>
4742 (MSTS-0161)   <SignatureValue>
4743 (MSTS-0162)   Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbmCtZFuQz9tnW35Lmst6vI
4744 (MSTS-0163)   RefuPA7ewRLYORAOjf92SxMbeVTlrxQbIQNw0bs4SBSLfAo14=
4745 (MSTS-0164)   </SignatureValue>
4746 (MSTS-0165)   <KeyInfo>
4747 (MSTS-0166)     <o:SecurityTokenReference>
4748 (MSTS-0167)       <o:Reference
4749 (MSTS-0168)         ValueType=
4750 (MSTS-0169)       "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4751 (MSTS-0170)       profile-1.0#X509v3"
4752 (MSTS-0171)         URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-6" />
4753 (MSTS-0172)       </o:SecurityTokenReference>
4754 (MSTS-0173)     </KeyInfo>
4755 (MSTS-0174)   </Signature>
4756 (MSTS-0175)   </o:Security>
4757 (MSTS-0176)   </s:Header>
4758 (MSTS-0177)   <s:Body u:Id="_1">
4759 (MSTS-0178)     <e:EncryptedData
4760 (MSTS-0179)       Id="_2"
4761 (MSTS-0180)       Type="http://www.w3.org/2001/04/xmlenc#Content">
4762 (MSTS-0181)       <e:EncryptionMethod Algorithm=
4763 (MSTS-0182)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
4764 (MSTS-0183)       <e:CipherData>
4765 (MSTS-0184)         <e:CipherValue>
4766 (MSTS-0185)           <!-- base64 encoded octets with encrypted RST request-->
4767 (MSTS-0186)           <!-- Unencrypted form: -->
4768 (MSTS-0187)           <!--
4769 (MSTS-0188)       <t:RequestSecurityToken>
4770 (MSTS-0189)         <t:RequestType>
4771 (MSTS-0190)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
4772 (MSTS-0191)         </t:RequestType>
4773 (MSTS-0192)       <wsp:AppliesTo
4774 (MSTS-0193)         xmlns:wsp="http://www.w3.org/ns/ws-policy">
4775 (MSTS-0194)           <a:EndpointReference
4776 (MSTS-0195)             xmlns:a="http://www.w3.org/2005/08/addressing">
4777 (MSTS-0196)             <a:Address
4778 (MSTS-0197)               >http://server.example.com/Scenarios5</a:Address>
4779 (MSTS-0198)           </a:EndpointReference>
4780 (MSTS-0199)         </wsp:AppliesTo>
4781 (MSTS-0200)       <t:Entropy>
4782 (MSTS-0201)         <t:BinarySecret
4783 (MSTS-0202)           u:Id="uuid-4acf589c-0076-4a83-8b66-5f29341514b7-3"
4784 (MSTS-0203)           Type=
4785 (MSTS-0204)             "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
4786 (MSTS-0205)         >Uv38QLxDQM9gLoDZ6OwYDiFk094nmwu3Wmay7EdKmhv=</t:BinarySecret>
4787 (MSTS-0206)       </t:Entropy>
4788 (MSTS-0207)     <t:KeyType>
4789 (MSTS-0208)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey
4790 (MSTS-0209)     </t:KeyType>

```

```

4791 (MSTS-0209) <t:KeySize>256</t:KeySize>
4792 (MSTS-0210) <t:ComputedKeyAlgorithm>
4793 (MSTS-0211) http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
4794 (MSTS-0212) </t:ComputedKeyAlgorithm>
4795 (MSTS-0213) <t:SecondaryParameters>
4796 (MSTS-0214) <t:TokenType
4797 (MSTS-0215) >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4798 1.1#SAMLV1.1</t:TokenType>
4799 (MSTS-0216) <t:KeyType
4800 (MSTS-0217) >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:Ke
4801 yType>
4802 (MSTS-0218) <t:KeySize>256</t:KeySize>
4803 (MSTS-0219) <t:CanonicalizationAlgorithm
4804 (MSTS-0220) >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorith
4805 m>
4806 (MSTS-0221) <t:EncryptionAlgorithm
4807 (MSTS-0222) >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4808 (MSTS-0222) <t:EncryptWith
4809 (MSTS-0223) >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4810 (MSTS-0223) <t:SignWith
4811 (MSTS-0224) >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4812 (MSTS-0224) </t:SecondaryParameters>
4813 (MSTS-0225) </t:RequestSecurityToken>
4814 (MSTS-0226) -->
4815 (MSTS-0227) </e:CipherValue>
4816 (MSTS-0228) </e:CipherData>
4817 (MSTS-0229) </e:EncryptedData>
4818 (MSTS-0230) </s:Body>
4819 (MSTS-0231) </s:Envelope>

```

4820 The message above is a request by the Client to the STS in compliance with the STS policy (PSTS-001)-
4821 (PSTS-098). Salient features of this message appropriate to the compliance are described below.

4822 Lines (MSTS-027)-(MSTS-047) contain the **e:EncryptedKey element**, which contains the encrypted
4823 client-generated ephemeral key (K1) (MSTS-039)-(MSTS-043). K1 is encrypted using the STS certificate
4824 (X509T2) which is specified by its Thumbprint identifier in the WS-Security o:SecurityTokenReference
4825 (MSTS-032)-(MSTS-037). The e:ReferenceList in the e:EncryptedKey (MSTS-044)-(MSTS-046) indicates
4826 that the encryption key K1 is used to directly encrypt the message Body which is referenced by the
4827 e:DataReference URI="#_2" (MSTS-045).

4828 Lines (MSTS-048)-(MSTS-071) contain a **WS-Security o:BinarySecurityToken**, which contains the
4829 Client's public X509 certificate (X509T1) that is required for Client authentication to the STS by the
4830 sp:EndorsingToken in the STS policy (PSTS-037)-(PSTS-047).

4831 Lines (MSTS-072)-(MSTS-0144) contain the **WS-SP message signature** in an XML Digital Signature
4832 (dsig) element (namespace = ...xmldsig (MSTS-072)). The elements covered by the dsig Signature are
4833 identified in the dsig Reference elements:

- 4834 ➤ dsig Reference (MSTS-078) covers the s:Body (Id="_1" (MSTS-0176))
- 4835 ➤ dsig Reference (MSTS-087) covers the a:Action (Id="_3" (MSTS-007))
- 4836 ➤ dsig Reference (MSTS-096) covers the a:MessageID (Id="_4" (MSTS-010))
- 4837 ➤ dsig Reference (MSTS-0105) covers the a:ReplyTo (Id="_5" (MSTS-013))
- 4838 ➤ dsig Reference (MSTS-0114) covers the a:To (Id="_6" (MSTS-018))
- 4839 ➤ dsig Reference (MSTS-0123) covers the u:Timestamp (Id="uuid ... 3849-10" (MSTS-023))

4840 all as required by the STS Input and Output Policy sp:SignedParts (PSTS-076)-(PSTS-092), which covers
4841 the first 5 elements above (note: if an element is not present that is identified in the policy (such as
4842 FaultTo or RelatesTo) it obviously is not required to be signed, but if present must be signed). The
4843 u:Timestamp is required to be signed by its presence in the STS endpoint policy (PSTS-033).

4844 Lines (MSTS-0136)-(MSTS-0143) of the **WS-SP message signature contain the dsig KeyInfo**, which
4845 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing
4846 validation key (i.e. the key which can be used to verify this dsig:Signature), which in this case is contained

4847 in the e:EncryptedKey element (Id="uuid ... 3849-9" (MSTS-027)) that was described above. Note: at this
4848 point to verify the Signature, the STS will decrypt the e:EncryptedKey using the private key from the STS
4849 certificate, X509T2, which will produce the client-generated ephemeral signing and encryption key, K1,
4850 which, in turn, may be used to validate the message signature. Note also: at this point the STS only
4851 knows whether the data covered by the message signature is valid or not, but the STS does not yet know
4852 the identity of the entity that actually sent the data. That is covered next:

4853 Lines (MSTS-0145)-(MSTS-0173) contain the **WS-SP endorsing signature**, also in an XML Digital
4854 Signature element. The endorsing signature only covers one element, the message signature element,
4855 which is identified by the dsig Reference element:

4856 ➤ dsig Reference (MSTS-0151) covers the dsig Signature (Id="_0" (MSTS-072))
4857 as required by the STS endpoint policy sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047).

4858 Lines (MSTS-0165)-(MSTS-0174) of the **WS-SP endorsing signature contain the dsig KeyInfo**, which
4859 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing
4860 validation key, which in this case is in the o:BinarySecurityToken element (Id="uuid ... 3849-6"
4861 (MSTS-048)) that was also described above. Note: now the STS finally has the credentials necessary to
4862 authenticate the Client. The STS will generally have an identity database of trusted clients and their
4863 associated X509 certificates. If a client successfully produces a signature that can be validated by the
4864 associated certificate in the STS database, which would have to match the certificate in the
4865 o:BinarySecurityToken element, then the client is presumed to be authenticated to the level of security
4866 provided by this mechanism (strong single factor authentication).

4867 Lines (MSTS-0176)-(MSTS-0230) contain the **SOAP message s:Body element**, which contains its
4868 payload in an e:EncryptedData element (MSTS-0177)-(MSTS-0229). This e:EncryptedData element was
4869 identified above by its Id="_2" in the e:ReferenceList (MSTS-044) of the e:EncryptedKey that was
4870 processed by the STS above to obtain the client ephemeral key (K1), with which this e:EncryptedData
4871 can be decrypted. Generally, when the e:EncryptedKey is processed, this e:EncryptedData would have
4872 been decrypted at that time and made available for processing in the event of successful Client
4873 authentication as described above. Because the contents of this payload are relevant to the rest of this
4874 example, the contents of the payload will be briefly described.

4875 Lines (MSTS-0187)-(MSTS-0225) contain the **decrypted contents of the SOAP message s:Body**
4876 **element**, which contain a WS-Trust t:RequestSecurityToken element. The t:RequestType (MSTS-0189)
4877 is "...Issue", which means this is a request to issue a security token, which is the main service of the STS.
4878 The WS-Policy wsp:AppliesTo element (MSTS-0191)-(MSTS-0198) contains the a:Address of the service,
4879 to which the Client is requesting access, a service which the STS is presumably responsible for
4880 authenticating and equipping validated clients to enable their access to. In this case the service is
4881 identified by the URL <http://server.example.com/Scenarios5>, which was part of the WS-SX Interop
4882 [[WSSX-WSTR-WSSC-INTEROP](#)].

4883 Lines (MSTS-0199)-(MSTS-0205) contain the Client entropy required by the Service policy (P079), which
4884 is entropy data provided by the Client to aid in production of the ephemeral key, K2, that will be used for
4885 Client-Service communication. Lines (MSTS-0206)-(MSTS-0212) contain additional parameters used in
4886 the WS-Trust interface to the STS that will not be described here.

4887 Lines (MSTS-0213)-(MSTS-0224) contain the WS-Trust t:SecondaryParameters, which contains the
4888 contents of the Service policy's RequestSecurityTokenTemplate (P045)-(P059), which the Service
4889 requires that the Client pass to the STS as described above. The t:SecondaryParameters contains the
4890 information the Service has requested about the SAML 1.1 IssuedToken that the STS is being requested
4891 to create and deliver to the Client in order the enable the Client to access the Service successfully.

4892 Assuming everything above has executed successfully, the STS will then issue the SAML 1.1
4893 IssuedToken and return it to the Client in a WS-Trust t:RequestSecurityTokenResponse that is described
4894 next.

4895

4896 Here is an example STS response to the above Client request:

```
4897 (RSTS-001) <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope  
4898 (RSTS-002)     xmlns:a=http://www.w3.org/2005/08/addressing  
4899 (RSTS-003)     xmlns:e=http://www.w3.org/2001/04/xmlenc#
```

```

4900 (RSTS-004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
4901 secext-1.1.xsd"
4902 (RSTS-005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4903 wssecurity-secext-1.0.xsd"
4904 (RSTS-006)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4905 wssecurity-utility-1.0.xsd" >
4906 (RSTS-007)      <s:Header>
4907 (RSTS-008)      <a:Action s:mustUnderstand="1" u:Id="_4">
4908 (RSTS-009)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
4909 (RSTS-010)      </a:Action>
4910 (RSTS-011)      <a:RelatesTo u:Id="_5">
4911 (RSTS-012)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4912 (RSTS-013)      </a:RelatesTo>
4913 (RSTS-014)      <a:To s:mustUnderstand="1" u:Id="_6">
4914 (RSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4915 (RSTS-016)      </a:To>
4916 (RSTS-017)      <o:Security s:mustUnderstand="1">
4917 (RSTS-018)      <u:Timestamp
4918 (RSTS-019)      u:Id="uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4919 (RSTS-020)      <u:Created>2005-10-25T00:47:38.718Z</u:Created>
4920 (RSTS-021)      <u:Expires>2005-10-25T00:52:38.718Z</u:Expires>
4921 (RSTS-022)      </u:Timestamp>
4922 (RSTS-023)      <e:ReferenceList>
4923 (RSTS-024)      <e:DataReference URI="#_3"/>
4924 (RSTS-025)      </e:ReferenceList>
4925 (RSTS-026)      <k:SignatureConfirmation u:Id="_0"
4926 (RSTS-027)      Value="iHGJ+xV2VZTjM1Rc7AQJrWLY/aM="/>
4927 (RSTS-028)      <k:SignatureConfirmation u:Id="_1"
4928 (RSTS-029)      Value=
4929 (RSTS-030)      "Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWicmG3xGLZYcc8tbmCtZFuQz9tnW35
4930 Lmst6vRefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLFAo14="/>
4931 (RSTS-032)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
4932 (RSTS-033)      <SignedInfo>
4933 (RSTS-034)      <CanonicalizationMethod Algorithm=
4934 (RSTS-035)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4935 (RSTS-036)      <SignatureMethod Algorithm=
4936 (RSTS-037)      "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
4937 (RSTS-038)      <Reference URI="#_2">
4938 (RSTS-039)      <Transforms>
4939 (RSTS-040)      <Transform Algorithm=
4940 (RSTS-041)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4941 (RSTS-042)      </Transforms>
4942 (RSTS-043)      <DigestMethod Algorithm=
4943 (RSTS-044)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4944 (RSTS-045)      <DigestValue>kKx5bpLLlyucgXQ6exv/PbjsFlA=</DigestValue>
4945 (RSTS-046)      </Reference>
4946 (RSTS-047)      <Reference URI="#_4">
4947 (RSTS-048)      <Transforms>
4948 (RSTS-049)      <Transform Algorithm=
4949 (RSTS-050)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4950 (RSTS-051)      </Transforms>
4951 (RSTS-052)      <DigestMethod Algorithm=
4952 (RSTS-053)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4953 (RSTS-054)      <DigestValue>LB+VGn4fP2z45jg0Mdzyo8yTAWQ=</DigestValue>
4954 (RSTS-055)      </Reference>
4955 (RSTS-056)      <Reference URI="#_5">
4956 (RSTS-057)      <Transforms>
4957 (RSTS-058)      <Transform Algorithm=
4958 (RSTS-059)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4959 (RSTS-060)      </Transforms>
4960 (RSTS-061)      <DigestMethod Algorithm=
4961 (RSTS-062)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4962 (RSTS-063)      <DigestValue>izHLxm6V4Lc3Pss9Y6VRv3I5RPw=</DigestValue>
4963 (RSTS-064)      </Reference>

```

```

4964 (RSTS-065) <Reference URI="#_6">
4965 (RSTS-066) <Transforms>
4966 (RSTS-067) <Transform Algorithm=
4967 (RSTS-068) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4968 </Transforms>
4969 (RSTS-069) <DigestMethod Algorithm=
4970 (RSTS-070) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4971 (RSTS-071) <DigestValue>6LS4X08vC/GMGay2vwmD8fL7J2U=</DigestValue>
4972 (RSTS-072) </Reference>
4973 (RSTS-073) <Reference
4974 (RSTS-074) URI="#uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4975 (RSTS-075) <Transforms>
4976 (RSTS-076) <Transform Algorithm=
4977 (RSTS-077) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4978 (RSTS-078) </Transforms>
4979 (RSTS-079) <DigestMethod Algorithm=
4980 (RSTS-080) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4981 (RSTS-081) <DigestValue>uXGSpCBfbT1fLNBLdGMgy6DGDio=</DigestValue>
4982 (RSTS-082) </Reference>
4983 (RSTS-083) <Reference URI="#_0">
4984 (RSTS-084) <Transforms>
4985 (RSTS-085) <Transform Algorithm=
4986 (RSTS-086) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4987 (RSTS-087) </Transforms>
4988 (RSTS-088) <DigestMethod Algorithm=
4989 (RSTS-089) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4990 (RSTS-090) <DigestValue>z86w+GrzqRZF56ciuz6ogzVXAUA=</DigestValue>
4991 (RSTS-091) </Reference>
4992 (RSTS-092) <Reference URI="#_1">
4993 (RSTS-093) <Transforms>
4994 (RSTS-094) <Transform Algorithm=
4995 (RSTS-095) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4996 (RSTS-096) </Transforms>
4997 (RSTS-097) <DigestMethod Algorithm=
4998 (RSTS-098) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4999 (RSTS-099) <DigestValue>Z9TfdD20a5aGngsyOPEvQuE0urvQ=</DigestValue>
5000 (RSTS-0100) </Reference>
5001 (RSTS-0101) </SignedInfo>
5002 (RSTS-0102) <SignatureValue>Q7HhboPUaZyXqUKgG7NCYlhMTXI=</SignatureValue>
5003 (RSTS-0103) <KeyInfo>
5004 (RSTS-0104) <o:SecurityTokenReference
5005 (RSTS-0105) k:TokenType="http://docs.oasis-open.org/wss/
5006 (RSTS-0106) oasis-wss-soap-message-security-1.1#EncryptedKey"
5007 (RSTS-0107) xmlns:k="http://docs.oasis-open.org/wss/
5008 (RSTS-0108) oasis-wss-wssecurity-secext-1.1.xsd">
5009 (RSTS-0109) <o:KeyIdentifier ValueType=
5010 (RSTS-0110) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5011 (RSTS-0111) 1.1.xsd#EncryptedKeySHA1">
5012 (RSTS-0112) CixQW5yEb3mw6XYqD8Ysvrf8cwI=
5013 (RSTS-0113) </o:KeyIdentifier>
5014 (RSTS-0114) </o:SecurityTokenReference>
5015 (RSTS-0115) </KeyInfo>
5016 (RSTS-0116) </Signature>
5017 (RSTS-0117) </o:Security>
5018 (RSTS-0118) </s:Header>
5019 (RSTS-0119) <s:Body u:Id="_2">
5020 (RSTS-0120) <e:EncryptedData Id="_3"
5021 (RSTS-0121) Type="http://www.w3.org/2001/04/xmlenc#Content">
5022 (RSTS-0122) <e:EncryptionMethod Algorithm=
5023 (RSTS-0123) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5024 (RSTS-0124) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5025 (RSTS-0125) <o:SecurityTokenReference
5026 (RSTS-0126) k:TokenType="http://docs.oasis-open.org/wss/
5027 (RSTS-0127) oasis-wss-soap-message-security-1.1#EncryptedKey"

```

```

5028 (RSTS-0127)          xmlns:k="http://docs.oasis-open.org/wss/
5029 (RSTS-0128)          oasis-wss-wssecurity-secext-1.1.xsd">
5030 (RSTS-0129)          <o:KeyIdentifier ValueType=
5031 (RSTS-0130)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5032 (RSTS-0131)          1.1.xsd#EncryptedKeySHA1">
5033 (RSTS-0131)          CixQW5yEb3mw6XYqD8Ysvrf8cwI=
5034 (RSTS-0132)          </o:KeyIdentifier>
5035 (RSTS-0133)          </o:SecurityTokenReference>
5036 (RSTS-0134)          </KeyInfo>
5037 (RSTS-0135)          <e:CipherData>
5038 (RSTS-0136)          <e:CipherValue>
5039 (RSTS-0137)          <!--base64 encoded octets of encrypted RSTR-->
5040 (RSTS-0138)          <!--
5041 (RSTS-0139)          <t:RequestSecurityTokenResponseCollection>
5042 (RSTS-0140)          <t:RequestSecurityTokenResponse>
5043 (RSTS-0141)          <t:TokenType>
5044 (RSTS-0142)          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5045 (RSTS-0143)          1.1#SAMLV1.1
5046 (RSTS-0143)          </t:TokenType>
5047 (RSTS-0144)          <t:KeySize>256</t:KeySize>
5048 (RSTS-0145)          <t:RequestedAttachedReference>
5049 (RSTS-0146)          <o:SecurityTokenReference>
5050 (RSTS-0147)          <o:KeyIdentifier ValueType=
5051 (RSTS-0148)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5052 (RSTS-0148)          1.0#SAMLAssertionID">
5053 (RSTS-0149)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5054 (RSTS-0150)          </o:KeyIdentifier>
5055 (RSTS-0151)          </o:SecurityTokenReference>
5056 (RSTS-0152)          </t:RequestedAttachedReference>
5057 (RSTS-0153)          <t:RequestedUnattachedReference>
5058 (RSTS-0154)          <o:SecurityTokenReference>
5059 (RSTS-0155)          <o:KeyIdentifier ValueType=
5060 (RSTS-0156)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5061 (RSTS-0156)          1.0#SAMLAssertionID">
5062 (RSTS-0157)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5063 (RSTS-0158)          </o:KeyIdentifier>
5064 (RSTS-0159)          </o:SecurityTokenReference>
5065 (RSTS-0160)          </t:RequestedUnattachedReference>
5066 (RSTS-0161)          <t:Lifetime>
5067 (RSTS-0162)          <u:Created>2005-10-24T20:19:26.526Z</u:Created>
5068 (RSTS-0163)          <u:Expires>2005-10-25T06:24:26.526Z</u:Expires>
5069 (RSTS-0164)          </t:Lifetime>
5070 (RSTS-0165)          <t:RequestedSecurityToken>
5071 (RSTS-0166)          <saml:Assertion MajorVersion="1" MinorVersion="1"
5072 (RSTS-0167)          AssertionID="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16"
5073 (RSTS-0168)          Issuer="Test STS" IssueInstant="2005-10-24T20:24:26.526Z"
5074 (RSTS-0169)          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
5075 (RSTS-0170)          <saml:Conditions NotBefore="2005-10-24T20:19:26.526Z"
5076 (RSTS-0171)          NotOnOrAfter="2005-10-25T06:24:26.526Z">
5077 (RSTS-0172)          <saml:AudienceRestrictionCondition>
5078 (RSTS-0173)          <saml:Audience
5079 (RSTS-0174)          >http://server.example.com/Scenarios5</saml:Audience>
5080 (RSTS-0175)          </saml:AudienceRestrictionCondition>
5081 (RSTS-0176)          </saml:Conditions>
5082 (RSTS-0177)          <saml:Advice>
5083 (RSTS-0178)          </saml:Advice>
5084 (RSTS-0179)          <saml:AttributeStatement>
5085 (RSTS-0180)          <saml:Subject>
5086 (RSTS-0181)          <saml:SubjectConfirmation>
5087 (RSTS-0182)          <saml:ConfirmationMethod>
5088 (RSTS-0183)          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
5089 (RSTS-0184)          </saml:ConfirmationMethod>
5090 (RSTS-0185)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5091 (RSTS-0186)          <e:EncryptedKey

```

```

5092 (RSTS-0187)          xmlns:e="http://www.w3.org/2001/04/xmlenc#">
5093 (RSTS-0188)          <e:EncryptionMethod Algorithm=
5094 (RSTS-0189)          "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
5095 (RSTS-0190)          </e:EncryptionMethod>
5096 (RSTS-0191)          <KeyInfo>
5097 (RSTS-0192)          <o:SecurityTokenReference xmlns:o=
5098 (RSTS-0193)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5099                      secext-1.0.xsd">
5100 (RSTS-0194)          <o:KeyIdentifier ValueType=
5101 (RSTS-0195)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5102                      1.1.xsd#ThumbprintSHA1">
5103 (RSTS-0196)          NQM0IBvuplAtETQvk+6gn8C13wE=
5104 (RSTS-0197)          </o:KeyIdentifier>
5105 (RSTS-0198)          </o:SecurityTokenReference>
5106 (RSTS-0199)          </KeyInfo>
5107 (RSTS-0200)          <e:CipherData>
5108 (RSTS-0201)          <e:CipherValue>
5109 (RSTS-0202)          EEcYjwNoYcJ+20xTYE5e/fixl5KOgzgrfaYAxkDFv/VXiuKfl084h8PmogTfM+azcgAf
5110 (RSTS-0203)          mArVQvOyKWXRb5vmXYfVHLLhZTbXacy+nowSUNnEjp37VDbI3RJ5k6tBHF+ow0NM/P6G
5111 (RSTS-0204)          PNZ9ZqJi1lGDgWJkFsJzNZXNbbMgwuFu3cA=</e:CipherValue>
5112 (RSTS-0205)          </e:CipherData>
5113 (RSTS-0206)          </e:EncryptedKey>
5114 (RSTS-0207)          </KeyInfo>
5115 (RSTS-0208)          </saml:SubjectConfirmation>
5116 (RSTS-0209)          </saml:Subject>
5117 (RSTS-0210)          </saml:AttributeStatement>
5118 (RSTS-0211)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5119 (RSTS-0212)          <SignedInfo>
5120 (RSTS-0213)          <CanonicalizationMethod Algorithm=
5121 (RSTS-0214)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5122 (RSTS-0215)          </CanonicalizationMethod>
5123 (RSTS-0216)          <SignatureMethod Algorithm=
5124 (RSTS-0217)          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
5125 (RSTS-0218)          </SignatureMethod>
5126 (RSTS-0219)          <Reference
5127 (RSTS-0220)          URI="#uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16">
5128 (RSTS-0221)          <Transforms>
5129 (RSTS-0222)          <Transform Algorithm=
5130 (RSTS-0223)          "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
5131 (RSTS-0224)          </Transform>
5132 (RSTS-0225)          <Transform Algorithm=
5133 (RSTS-0226)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5134 (RSTS-0227)          </Transform>
5135 (RSTS-0228)          </Transforms>
5136 (RSTS-0229)          <DigestMethod Algorithm=
5137 (RSTS-0230)          "http://www.w3.org/2000/09/xmldsig#sha1">
5138 (RSTS-0231)          </DigestMethod>
5139 (RSTS-0232)          <DigestValue
5140 (RSTS-0233)          >7nHBrFPsm+LEFAoV4NoQPoEl5Lk=</DigestValue>
5141 (RSTS-0234)          </Reference>
5142 (RSTS-0235)          </SignedInfo>
5143 (RSTS-0236)          <SignatureValue
5144 (RSTS-0237)          >TugV4pTIwCH87bLD4jiMgVGtkbRbtltRlHXJArL34A/YfA4AnGLXB4pJdUsUxMUTbQ
5145 (RSTS-0238)          14PoGgEsdLNg8C77peARELGP1/Tqw7T3u5zBYHxCHCiv2FWBBfeOmwJmgoaBf8XZJ4Al
5146 (RSTS-0239)          yqPq61P61jrQjZJafpHuYpAZnZQsvisiJaBPQ=</SignatureValue>
5147 (RSTS-0240)          <KeyInfo>
5148 (RSTS-0241)          <o:SecurityTokenReference xmlns:o=
5149 (RSTS-0242)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5150                      secext-1.0.xsd">
5151 (RSTS-0243)          <o:KeyIdentifier ValueType=
5152 (RSTS-0244)          http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5153                      1.1.xsd#ThumbprintSHA1
5154 (RSTS-0245)          >W+rqYBmLmVEG//scD7Vo8Kq5G7I=</o:KeyIdentifier>
5155 (RSTS-0246)          </o:SecurityTokenReference>

```



```

5156 (RSTS-0247)           </KeyInfo>
5157 (RSTS-0248)           </Signature>
5158 (RSTS-0249)           </saml:Assertion>
5159 (RSTS-0250)           </t:RequestedSecurityToken>
5160 (RSTS-0251)           <t:RequestedProofToken>
5161 (RSTS-0252)           <t:BinarySecret
5162 (RSTS-0253)           u:Id="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-14"
5163 (RSTS-0254)           >zT8LWAUwUrIVKA/rkCr0kx1EmKAehcB6TGWJuAgucBM=</t:BinarySecret>
5164 (RSTS-0255)           </t:RequestedProofToken>
5165 (RSTS-0256)           </t:RequestSecurityTokenResponse>
5166 (RSTS-0257)           </t:RequestSecurityTokenResponseCollection>
5167 (RSTS-0258)           -->
5168 (RSTS-0259)           </e:CipherValue>
5169 (RSTS-0260)           </e:CipherData>
5170 (RSTS-0261)           </e:EncryptedData>
5171 (RSTS-0262)           </s:Body>
5172 (RSTS-0263)           </s:Envelope>

```

5173 From here on the description will be less detailed except where new concepts that have not been covered
5174 previously in this example. For instance, tracing the dsig References to their associated elements and
5175 policy requirements will not be detailed since it follows the same patterns that have just been described in
5176 the message above.

5177 The message above is a response from the STS to the Client that contains the requested security tokens
5178 that the Client needs to access the Service.

5179 Lines (RSTS-023)-(RSTS-025) contain a **standalone e:ReferenceList** that has an e:DataReference
5180 (Id="_3") that points to the e:EncryptedData element in the SOAP s:Body (RSTS-0119). This
5181 e:DataReference will be used later in the processing of this message.

5182 Lines (RSTS-026)-(RSTS-031) contain **2 WS-Security 1.1 k:SignatureConfirmation elements** that
5183 indicate to the Client that the STS has processed the data in the Client request and in particular, has
5184 processed the information covered by the 2 dsig Signatures in that request, that are identified by their
5185 respective dsig SignatureValue elements (see lines (MSTS-0134)-(MSTS-0135) and (MSTS-0161)-
5186 (MSTS-0163) in the Client request to the STS above to compare SignatureValues).

5187 Lines (RSTS-032)-(RSTS-0115) contain the **WS-SP message signature** for this message.

5188 Lines (RSTS-0103)-(RSTS-0114) contain the **WS-SP message signature dsig KeyInfo element**, which
5189 contains a WS-Security 1.1 o:SecurityTokenReference, which contains an o:KeyIdentifier of ValueType
5190 "... EncryptedKeySHA1". This is a WS-Security 1.1 construct [[WS_SECURITY_11](#)] used to reference a
5191 security token that is not included in the message, which in this case is the Client-generated ephemeral
5192 key, K1, that the Client used to prepare the request and delivered to the STS in the e:EncryptedKey
5193 element in that request (MSTS-027) that was described in detail above. The contents of the
5194 o:KeyIdentifier (RSTS-0111) consist of the SHA1 of K1. This should be sufficient information to let the
5195 Client know that the STS used its X509 certificate, X509T2, to decrypt K1 from the e:EncryptedKey in the
5196 Client request, which will enable the Client to trust the contents of this STS response.

5197 Lines (RSTS-0119)-(RSTS-0261) contain the **e:EncryptedData**, which is also provided in decrypted form
5198 for instructive purposes. As mentioned above, this element is referred to by the standalone
5199 e:ReferenceList element in the WS-Security header (RSTS-023) and as a result can be not tied to any
5200 particular encryption key as described in [[WS_SECURITY_11](#)], and since referenced will be processed by
5201 WS-Security.

5202 Lines (RSTS-0123)-(RSTS-0134) contain the **e:EncryptedData dsig KeyInfo**, which refers to the same
5203 ephemeral key, K1, as described above for the message signature for this message.

5204 Lines (RSTS-0140)-(RSTS-0256) contain the **decrypted WS-Trust t:RequestSecurityTokenResponse**
5205 from the STS. Briefly, lines (RSTS-0145)-(RSTS-0160) contain WS-Security o:SecurityTokenReference
5206 elements that refer to the SAML 1.1 Assertion that is provided below. These are convenience elements
5207 for the Client to use in subsequent message preparation where the SAML Assertion is used.

5208 Lines (RSTS-0165)-(RSTS-0250) contain **the actual t:RequestedSecurityToken** that has been the main
5209 object of discussion up until this point, which is the SAML 1.1 Assertion, saml:Assertion (RSTS-0166)-
5210 (RSTS-0249), that is the sp:IssuedToken provided by the STS for the Client to use to access the Service.

5211 Lines (RSTS-0170)-(RSTS-0176) of the `saml:Assertion` contain the **saml:Conditions** that specify that this
5212 token is intended only for the Service, identified by the URL in the **saml:Audience** element (RSTS-0174).

5213 Lines (RSTS-0181)-(RSTS-0208) contain the **all-important saml:SubjectConfirmation element**, which
5214 contains the **STS-generated encrypted ephemeral key, K3**, that will be used by the Client and Service
5215 to communicate securely. There are **2 copies of this key, K3**, in this `t:RequestSecurityTokenResponse`.
5216 This copy is for the Service. The Client's copy is described below. In any event, the
5217 `saml:SubjectConfirmation` element contains a `dsig:KeyInfo` (RSTS-0185)-(RSTS-0207), which contains an
5218 `e:EncryptedKey` element (RSTS-0186)-(RSTS-0206), which, in turn, contains a second `dsig:KeyInfo`,
5219 which contains a `WS-Security o:SecurityTokenReference` element that contains an `o:KeyIdentifier` (RSTS-
5220 0194)-(RSTS-0197) that identifies the key, X509T3, the Service's public X509 certificate, that can be
5221 used by the Service to decrypt the ultimate object here, which is the ephemeral key, K3, contained in the
5222 `e:CipherData` (RSTS-0200)-(RSTS-0205). The Service's public X509 certificate is identified by its `WS-
5223 Security 1.1 ThumbprintSHA1` (RSTS-0196). Therefore, when the Service receives this `saml:Assertion`, it
5224 has the ability to obtain the ephemeral key, K3, contained in the `saml:SubjectConfirmation`, with which it
5225 can securely communicate with the Client, based on the assurances provided by the STS.

5226 Lines (RSTS-0211)-(RSTS-0248) contain the **saml:Assertion dsig Signature** that is contained in and
5227 covers the `saml:Assertion` via the `saml:AssertionID`, the value of which appears on lines (RSTS-0220) and
5228 (RSTS-0167).

5229 Lines (RSTS-0240)-(RSTS-0247) contain the **saml:Assertion dsig Signature KeyInfo element**, which
5230 contains the `ThumbprintSHA1` of the **STS public key, X509T2**, which is the same certificate that the
5231 Client referenced in the `e:EncryptedKey` when it made initial contact with the STS above (MSTS-035).
5232 This completes the initial discussion of the characteristics of the IssuedToken `saml:Assertion` that will be
5233 used in the remainder of this example.

5234 Lines (RSTS-0251)-(RSTS-0255) of the `t:RequestSecurityTokenResponse` contains the
5235 **t:RequestedProofToken**, which contains the **Client copy of the STS-generated ephemeral key, K3**,
5236 which will be used in the Client-Service communication to authenticate the client to the Service.

5237 At this point we have completed the setup portion of this example that has enabled secure trusted
5238 communication between the Client and Service as governed by an STS. The exact strength of the
5239 security protecting this example would be the subject of an official security analysis that is beyond the
5240 scope of this document, however, the intent has been to provide sufficient detail that all parties concerned
5241 with such an analysis would have sufficient context to understand and evaluate such an analysis.

5242

5243 Here is an example of a Client request to the Service using the tokens from the STS response:

```

5244
5245 (M001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5246 (M002)   xmlns:a=http://www.w3.org/2005/08/addressing
5247 (M003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5248 (M004)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5249 (M005)   wssecurity-secext-1.0.xsd"
5250 (M006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5251 (M007)   secureconversation/200512"
5252 (M008)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5253 (M009)   wssecurity-utility-1.0.xsd" >
5254 (M010)   <s:Header>
5255 (M011)     <a:Action s:mustUnderstand="1" u:Id="_5">
5256 (M012)       http://example.org/Ping
5257 (M013)     </a:Action>
5258 (M014)     <a:MessageID u:Id="_6">
5259 (M015)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5260 (M016)     </a:MessageID>
5261 (M017)     <a:ReplyTo u:Id="_7">
5262 (M018)       <a:Address>
5263 (M019)         http://www.w3.org/2005/08/addressing/anonymous
5264 (M020)       </a:Address>
5265 (M021)     </a:ReplyTo>
5266 (M022)     <a:To s:mustUnderstand="1" u:Id="_8">
5267 (M023)       http://server.example.com/Scenarios5

```

```

5268 (M021) </a:To>
5269 (M022) <o:Security s:mustUnderstand="1">
5270 (M023) <u:Timestamp
5271 (M024) <u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5272 (M025) <u:Created>2005-10-25T00:47:38.222Z</u:Created>
5273 (M026) <u:Expires>2005-10-25T00:52:38.222Z</u:Expires>
5274 (M027) </u:Timestamp>
5275 (M028) <e:EncryptedKey
5276 (M029) <Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-4">
5277 (M030) <e:EncryptionMethod Algorithm=
5278 (M031) http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5279 (M032) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5280 (M033) <o:SecurityTokenReference>
5281 (M034) <o:KeyIdentifier ValueType=
5282 (M035) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5283 1.1.xsd#ThumbprintSHA1">
5284 (M036) NQM0IBvuplAtETQvk+6gn8C13wE=
5285 (M037) </o:KeyIdentifier>
5286 (M038) </o:SecurityTokenReference>
5287 (M039) </KeyInfo>
5288 (M040) <e:CipherData>
5289 (M041) <e:CipherValue>
5290 (M042) <!-- base64 encoded octets of encrypted key K2 -->
5291 (M043) </e:CipherValue>
5292 (M044) </e:CipherData>
5293 (M045) </e:EncryptedKey>
5294 (M046) <sc:DerivedKeyToken u:Id="_0" >
5295 (M047) <o:SecurityTokenReference>
5296 (M048) <o:Reference
5297 (M049) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5298 (M050) </o:SecurityTokenReference>
5299 (M051) <sc:Offset>0</sc:Offset>
5300 (M052) <sc:Length>24</sc:Length>
5301 (M053) <sc:Nonce>7hI6U16OHavffYgpquHWuQ==</sc:Nonce>
5302 (M054) </sc:DerivedKeyToken>
5303 (M055) <sc:DerivedKeyToken u:Id="_2">
5304 (M056) <o:SecurityTokenReference>
5305 (M057) <o:Reference
5306 (M058) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5307 (M059) </o:SecurityTokenReference>
5308 (M060) <sc:Nonce>OEu+WEEUxPFRQK7SCFAnEQ==</sc:Nonce>
5309 (M061) </sc:DerivedKeyToken>
5310 (M062) <e:ReferenceList>
5311 (M063) <e:DataReference URI="#_4"/>
5312 (M064) </e:ReferenceList>
5313 (M065) <!-- encrypted SAML assertion -->
5314 (M066) <e:EncryptedData Id="_3"
5315 (M067) Type="http://www.w3.org/2001/04/xmlenc#Element">
5316 (M068) <e:EncryptionMethod Algorithm=
5317 (M069) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5318 (M070) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5319 (M071) <!-- encrypted Key K2 -->
5320 (M072) <e:EncryptedKey>
5321 (M073) <e:EncryptionMethod Algorithm=
5322 (M074) "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5323 (M075) <KeyInfo>
5324 (M076) <o:SecurityTokenReference>
5325 (M077) <o:KeyIdentifier ValueType=
5326 (M078) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5327 1.1.xsd#ThumbprintSHA1">
5328 (M079) NQM0IBvuplAtETQvk+6gn8C13wE=
5329 (M080) </o:KeyIdentifier>
5330 (M081) </o:SecurityTokenReference>
5331 (M082) </KeyInfo>

```

```

5332 (M083) <e:CipherData>
5333 (M084) <e:CipherValue>
5334 (M085)
5335 cb7+JW2idPNSarK9quqCe9PQwmW2hoUghuyKRe+I9zOts6HaMcg73LqCWuK/jtdpvNl6
5336 (M086) GT/ZDYfcAJ7NlyMGxSiwi4DUlTOShqS60TYBIKgUKiA+zXNl2koVsy7amcUhPMIT6/fo
5337 (M087) hH+6MZDA4t6jomcyhlCiW8d9IAzSWFkfg2k=
5338 (M088) </e:CipherValue>
5339 (M089) </e:CipherData>
5340 (M090) </e:EncryptedKey>
5341 (M091) </KeyInfo>
5342 (M092) <e:CipherData>
5343 (M093) <e:CipherValue>
5344 (M094) <!-- base64 encoded octets from SAML assertion encrypted
5345 (M095) with the encrypted key K2 above -->
5346 (M096) <!-- SAML assertion element is identical as received in
5347 (M097) RSTR , unencrypted form is omitted for brevity -->
5348 (M098) <!--.....-->
5349 (M099) </e:CipherValue>
5350 (M100) </e:CipherData>
5351 (M101) </e:EncryptedData>
5352 (M102)
5353 (M10103) <sc:DerivedKeyToken u:Id="_9">
5354 (M10104) <o:SecurityTokenReference>
5355 (M10105) <o:KeyIdentifier ValueType=
5356 (M10106) "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5357 1.0#SAMLAssertionID">
5358 (M10107) <b>uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16</b>
5359 (M10108) </o:KeyIdentifier>
5360 (M10109) </o:SecurityTokenReference>
5361 (M10110) <sc:Offset>0</sc:Offset>
5362 (M10111) <sc:Length>24</sc:Length>
5363 (M10112) <sc:Nonce>pgnS/VDSzJn6SFz+Vy23JA==</sc:Nonce>
5364 (M10113) </sc:DerivedKeyToken>
5365 (M10114) <Signature Id="_1" xmlns="http://www.w3.org/2000/09/xmldsig#">
5366 (M10115) <SignedInfo>
5367 (M10116) <CanonicalizationMethod Algorithm=
5368 (M10117) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5369 (M10118) <SignatureMethod Algorithm=
5370 (M10119) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5371 (M10120) <Reference URI="#_3">
5372 (M10121) <Transforms>
5373 (M10122) <Transform Algorithm=
5374 (M10123) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5375 (M10124) </Transforms>
5376 (M10125) <DigestMethod Algorithm=
5377 (M10126) "http://www.w3.org/2000/09/xmldsig#sha1" />
5378 (M10127) <DigestValue>eQdQVGRkVI1YfKJBw7vOYCOeLQw=</DigestValue>
5379 (M10128) </Reference>
5380 (M10129) <Reference URI="#_5">
5381 (M10130) <Transforms>
5382 (M10131) <Transform Algorithm=
5383 (M10132) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5384 (M10133) </Transforms>
5385 (M10134) <DigestMethod Algorithm=
5386 (M10135) "http://www.w3.org/2000/09/xmldsig#sha1" />
5387 (M10136) <DigestValue>xxyKpp5RZ2TebKca2IGOafIgcxk=</DigestValue>
5388 (M10137) </Reference>
5389 (M10138) <Reference URI="#_6">
5390 (M10139) <Transforms>
5391 (M10140) <Transform Algorithm=
5392 (M10141) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5393 (M10142) </Transforms>
5394 (M10143) <DigestMethod Algorithm=
5395 (M10144) "http://www.w3.org/2000/09/xmldsig#sha1" />

```

```

5396 (M0145) <DigestValue>WyGDDyYbL/hQZJfE3Yx2aK3RkK8=</DigestValue>
5397 (M0146) </Reference>
5398 (M0147) <Reference URI="#_7">
5399 (M0148) <Transforms>
5400 (M0149) <Transform Algorithm=
5401 (M0150) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5402 (M0151) </Transforms>
5403 (M0152) <DigestMethod Algorithm=
5404 (M0153) "http://www.w3.org/2000/09/xmldsig#sha1" />
5405 (M0154) <DigestValue>AEOH0t2KYR8mivgqUGDrgMtxgEQ=</DigestValue>
5406 (M0155) </Reference>
5407 (M0156) <Reference URI="#_8">
5408 (M0157) <Transforms>
5409 (M0158) <Transform Algorithm=
5410 (M0159) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5411 (M0160) </Transforms>
5412 (M0161) <DigestMethod Algorithm=
5413 (M0162) "http://www.w3.org/2000/09/xmldsig#sha1" />
5414 (M0163) <DigestValue>y8n6Dxd3DbD6TR6d6H/oVWsV4yE=</DigestValue>
5415 (M0164) </Reference>
5416 (M0165) <Reference
5417 (M0166) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5418 (M0167) <Transforms>
5419 (M0168) <Transform Algorithm=
5420 (M0169) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5421 (M0170) </Transforms>
5422 (M0171) <DigestMethod Algorithm=
5423 (M0172) "http://www.w3.org/2000/09/xmldsig#sha1" />
5424 (M0173) <DigestValue>/Cc+bGkkeQ6j1VXZx8PGgmF6MjI=</DigestValue>
5425 (M0174) </Reference>
5426 (M0175) </SignedInfo>
5427 (M0176) <!--base64 encoded signature value -->
5428 (M0177) <SignatureValue>EyKUHUuffPUPE/ZjaFrMJJ5KLKY=</SignatureValue>
5429 (M0178) <KeyInfo>
5430 (M0179) <o:SecurityTokenReference>
5431 (M0180) <o:Reference URI="#_0" />
5432 (M0181) </o:SecurityTokenReference>
5433 (M0182) </KeyInfo>
5434 (M0183) </Signature>
5435 (M0184) <!-- signature over the primary signature above
5436 (M0185) using the key derived from the proof-key, K3,
5437 (M0186) associated with SAML assertion -->
5438 (M0187) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5439 (M0188) <SignedInfo>
5440 (M0189) <CanonicalizationMethod Algorithm=
5441 (M0190) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5442 (M0191) <SignatureMethod Algorithm=
5443 (M0192) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5444 (M0193) <Reference URI="#_1">
5445 (M0194) <Transforms>
5446 (M0195) <Transform Algorithm=
5447 (M0196) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5448 (M0197) </Transforms>
5449 (M0198) <DigestMethod Algorithm=
5450 (M0199) "http://www.w3.org/2000/09/xmldsig#sha1" />
5451 (M0200) <DigestValue>TMSmLlgeUn8cxyb6OYe5Q2nUuxY=</DigestValue>
5452 (M0201) </Reference>
5453 (M0202) </SignedInfo>
5454 (M0203) <SignatureValue>Fh4NyOpAi+NqVFiHBgHWyvzah9I=</SignatureValue>
5455 (M0204) <KeyInfo>
5456 (M0205) <o:SecurityTokenReference>
5457 (M0206) <o:Reference URI="#_9" />
5458 (M0207) </o:SecurityTokenReference>
5459 (M0208) </KeyInfo>

```

```

5460 (M0209)          </Signature>
5461 (M0210)          </o:Security>
5462 (M0211)          </s:Header>
5463 (M0212)          <s:Body u:Id="_3">
5464 (M0213)          <e:EncryptedData Id="_4"
5465 (M0214)              Type="http://www.w3.org/2001/04/xmlenc#Content">
5466 (M0215)          <e:EncryptionMethod Algorithm=
5467 (M0216)              "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5468 (M0217)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5469 (M0218)          <o:SecurityTokenReference >
5470 (M0219)              <o:Reference URI="#_2"/>
5471 (M0220)          </o:SecurityTokenReference>
5472 (M0221)          </KeyInfo>
5473 (M0222)          <e:CipherData>
5474 (M0223)              <e:CipherValue>
5475 (M0224)                  <!-- base64 encoded octets of encrypted body content-->
5476 (M0225)              </e:CipherValue>
5477 (M0226)          </e:CipherData>
5478 (M0227)          </e:EncryptedData>
5479 (M0228)          </s:Body>
5480 (M0229)          </s:Envelope>

```

5481 The message above is a request from the Client to the Service using the tokens provided by the STS
5482 above.

5483 Lines (M022)-(M0210) contain the **WS-Security o:Security header** for this request.

5484 Lines (M028)-(M045) contain an **e:EncryptedKey** that contains the **ephemeral key, K2**, for the Client-
5485 Service communication. The service must use its X509T3 (M034)-(M037) to decrypt this Client-generated
5486 ephemeral key, K2.

5487 Lines (M046)-(M054) contain a **WS-SecureConversation sc:DerivedKeyToken**, that contains the
5488 information required to **generate the signing key, DKT1(K2)** [[WS_SECURE_CONV](#)], including an
5489 sc:Nonce (M053), and a WS-Security SecurityTokenReference that contains a direct reference (M049) to
5490 the e:EncryptedKey, K2 (M042). This derived key, DKT1(K2), is used to sign the message in the
5491 message signature element (M0114)-(M0183).

5492 Lines (M055)-(M054) provide similar **sc:DerivedKeyToken** constructs the **generate the encrypting key**
5493 **DKT2(K2)**. This derived key, DKT2(K2), is used to encrypt the message body, resulting in the
5494 EncryptedData element in the s:Body, on lines (M0213)-(M0227).

5495 Lines (M062)-(M064) provide an **e:ReferenceList to reference the e:EncryptedData in the s:Body**
5496 (M0213), which contains the Client request for the Service. The s:Body payload data in this Client request
5497 is not of interest to the security properties of this example and will not be shown in decrypted form.

5498 Lines (M066)-(M0101) contain an **e:EncryptedData element that contains the saml:Assertion**
5499 described in the STS response above. This e:EncryptedData contains a dsig KeyInfo, which, in turn,
5500 contains an e:EncryptedKey element (M072)-(M090), which contains the Client-generated ephemeral
5501 key, K2, which was used by the Client to directly encrypt the saml:Assertion. The encryption key, K2, may
5502 be decrypted, again using the Service public X509 certificate, again identified by its ThumbprintSHA1,
5503 (M077)-(M080).

5504 Lines (M0103)-(M0113) contain a **third sc:DerivedKeyToken** that contains the information necessary for
5505 the Service to **generate the endorsing signing key, DKT3(K2)**.

5506 Lines (M0114)-(M0183) contains the **message signature**. It is **signed using** the key identified in the dsig
5507 KeyInfo (M0178)-(M0182), which contains a reference to the **derived signing key, DKT1(K2)**, which may
5508 be constructed by the service using the sc:DerivedKeyToken (M046) described above.

5509 Lines (M0187)-(M0209) contain the **message endorsing signature**. It is **signed using the client proof**
5510 **key, K3**, that was generated by the STS. Note that the Service should compare this key, K3, with the one
5511 in the saml:SubjectConfirmation in the decrypted saml:Assertion to verify that the Client is using the same
5512 proof key, K3, that is contained in the saml:Assertion that authenticates this request.

5513 Lines (M0213)-(M0227) contain the **SOAP s:Body message payload, e:EncryptedData**, which may be
5514 decrypted using the sc:DerivedKeyToken (M055) to generate the decryption key DKT2(K2).

5515

5516 Here is an example response from the Service to the Client:

5517

```
(R001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5519 (R002)   xmlns:a=http://www.w3.org/2005/08/addressing
5520 (R003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5521 (R004)   xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5522 1.1.xsd"
5523 (R005)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5524 wssecurity-secext-1.0.xsd"
5525 (R006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5526 secureconversation/200512"
5527 (R007)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5528 wssecurity-utility-1.0.xsd">
5529 (R008)   <s:Header>
5530 (R009)     <a:Action s:mustUnderstand="1" u:Id="_6">
5531 (R010)       http://example.org/PingResponse
5532 (R011)     </a:Action>
5533 (R012)     <a:RelatesTo u:Id="_7">
5534 (R013)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5535 (R014)     </a:RelatesTo>
5536 (R015)     <a:To s:mustUnderstand="1" u:Id="_8">
5537 (R016)       http://www.w3.org/2005/08/addressing/anonymous
5538 (R017)     </a:To>
5539 (R018)     <o:Security s:mustUnderstand="1">
5540 (R019)       <u:Timestamp u:Id="uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5541 (R020)         <u:Created>2005-10-25T00:47:38.921Z</u:Created>
5542 (R021)         <u:Expires>2005-10-25T00:52:38.921Z</u:Expires>
5543 (R022)       </u:Timestamp>
5544 (R023)       <sc:DerivedKeyToken u:Id="_0" >
5545 (R024)         <o:SecurityTokenReference
5546 (R025)           k:TokenType="http://docs.oasis-open.org/wss/
5547 (R026)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5548 (R027)           xmlns:k="http://docs.oasis-open.org/wss/
5549 (R028)             oasis-wss-wssecurity-secext-1.1.xsd">
5550 (R029)             <o:KeyIdentifier ValueType=
5551 (R030)             "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5552 1.1.xsd#EncryptedKeySHA1"
5553 (R031)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5554 (R032)           </o:SecurityTokenReference>
5555 (R033)           <sc:Offset>0</sc:Offset>
5556 (R034)           <sc:Length>24</sc:Length>
5557 (R035)           <sc:Nonce>KFjylGb73BubLul0ZGgx+w==</sc:Nonce>
5558 (R036)         </sc:DerivedKeyToken>
5559 (R037)       <sc:DerivedKeyToken u:Id="_3">
5560 (R038)         <o:SecurityTokenReference
5561 (R039)           k:TokenType="http://docs.oasis-open.org/wss/
5562 (R040)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5563 (R041)           xmlns:k="http://docs.oasis-open.org/wss/
5564 (R042)             oasis-wss-wssecurity-secext-1.1.xsd">
5565 (R043)             <o:KeyIdentifier ValueType=
5566 (R044)             "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5567 1.1.xsd#EncryptedKeySHA1"
5568 (R045)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5569 (R046)           </o:SecurityTokenReference>
5570 (R047)           <sc:Nonce>omyh+Eg6XIa8q3V5IkHiXg==</sc:Nonce>
5571 (R048)         </sc:DerivedKeyToken>
5572 (R049)       <e:ReferenceList>
5573 (R050)         <e:DataReference URI="#_5"/>
5574 (R051)       </e:ReferenceList>
5575 (R052)       <k:SignatureConfirmation u:Id="_1"
5576 (R053)         Value="EyKUHUuffPUPE/ZjaFrMJJ5KLKY="/>
5577 (R054)       <k:SignatureConfirmation u:Id="_2"
```

```

5578 (R055) Value="Fh4NyOpAi+NqVFiHBgHWyvzah9I="/>
5579 (R056) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5580 (R057) <SignedInfo>
5581 (R058) <CanonicalizationMethod Algorithm=
5582 (R059) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5583 (R060) <SignatureMethod Algorithm=
5584 (R061) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5585 (R062) <Reference URI="#_4">
5586 (R063) <Transforms>
5587 (R064) <Transform Algorithm=
5588 (R065) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5589 (R066) </Transforms>
5590 (R067) <DigestMethod Algorithm=
5591 (R068) "http://www.w3.org/2000/09/xmldsig#sha1" />
5592 (R069) <DigestValue>y/oItF5TcTOFan7SavhZTTTv48M=</DigestValue>
5593 (R070) </Reference>
5594 (R071) <Reference URI="#_6">
5595 (R072) <Transforms>
5596 (R073) <Transform Algorithm=
5597 (R074) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5598 (R075) </Transforms>
5599 (R076) <DigestMethod Algorithm=
5600 (R077) "http://www.w3.org/2000/09/xmldsig#sha1" />
5601 (R078) <DigestValue>X4UIaLWnaAWTriw4UJ/SFDgm090=</DigestValue>
5602 (R079) </Reference>
5603 (R080) <Reference URI="#_7">
5604 (R081) <Transforms>
5605 (R082) <Transform Algorithm=
5606 (R083) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5607 (R084) </Transforms>
5608 (R085) <DigestMethod Algorithm=
5609 (R086) "http://www.w3.org/2000/09/xmldsig#sha1" />
5610 (R087) <DigestValue>vqy8/D4CDCaI1nnd4wl1Qjyp+qM=</DigestValue>
5611 (R088) </Reference>
5612 (R089) <Reference URI="#_8">
5613 (R090) <Transforms>
5614 (R091) <Transform Algorithm=
5615 (R092) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5616 (R093) </Transforms>
5617 (R094) <DigestMethod Algorithm=
5618 (R095) "http://www.w3.org/2000/09/xmldsig#sha1" />
5619 (R096) <DigestValue>H1lvLAr5g8pbZ6jfZ+2WNYiNjiM=</DigestValue>
5620 (R097) </Reference>
5621 (R098) <Reference
5622 (R099) URI="#uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5623 (R100) <Transforms>
5624 (R101) <Transform Algorithm=
5625 (R102) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5626 (R103) </Transforms>
5627 (R104) <DigestMethod Algorithm=
5628 (R105) "http://www.w3.org/2000/09/xmldsig#sha1" />
5629 (R106) <DigestValue>dr0g6hycoc884i+BD8FYCJGbbbE=</DigestValue>
5630 (R107) </Reference>
5631 (R108) <Reference URI="#_1">
5632 (R109) <Transforms>
5633 (R110) <Transform Algorithm=
5634 (R111) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5635 (R112) </Transforms>
5636 (R113) <DigestMethod Algorithm=
5637 (R114) "http://www.w3.org/2000/09/xmldsig#sha1" />
5638 (R115) <DigestValue>Rv3N7VNfAqpn0khr3F/qQZmE/l4=</DigestValue>
5639 (R116) </Reference>
5640 (R117) <Reference URI="#_2">
5641 (R118) <Transforms>

```



```

5642 (R0119)         <Transform Algorithm=
5643 (R0120)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
5644 (R0121)         </Transforms>
5645 (R0122)         <DigestMethod Algorithm=
5646 (R0123)         "http://www.w3.org/2000/09/xmldsig#sha1" />
5647 (R0124)         <DigestValue>X2pxEnYPM8cMLrbhNqPgs8xk+a4=</DigestValue>
5648 (R0125)         </Reference>
5649 (R0126)         </SignedInfo>
5650 (R0127)         <SignatureValue>I2jQuDTWWQiNJy/ziyg8AFYO/z4=</SignatureValue>
5651 (R0128)         <KeyInfo>
5652 (R0129)         <o:SecurityTokenReference>
5653 (R0130)         <o:Reference URI="#_0" />
5654 (R0131)         </o:SecurityTokenReference>
5655 (R0132)         </KeyInfo>
5656 (R0133)         </Signature>
5657 (R0134)         </o:Security>
5658 (R0135)         </s:Header>
5659 (R0136)         <s:Body u:Id="_4">
5660 (R0137)         <e:EncryptedData Id="_5"
5661 (R0138)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5662 (R0139)         <e:EncryptionMethod Algorithm=
5663 (R0140)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
5664 (R0141)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5665 (R0142)         <o:SecurityTokenReference>
5666 (R0143)         <o:Reference URI="#_3" />
5667 (R0144)         </o:SecurityTokenReference>
5668 (R0145)         </KeyInfo>
5669 (R0146)         <e:CipherData>
5670 (R0147)         <e:CipherValue>
5671 (R0148)         <!--base64 encoded octets of encrypted body content-->
5672 (R0149)         </e:CipherValue>
5673 (R0150)         </e:CipherData>
5674 (R0151)         </e:EncryptedData>
5675 (R0152)         </s:Body>
5676 (R0153)         </s:Envelope>

```

5677 The message above is a response from the Service to the Client using the tokens based on the
5678 saml:Assertion IssuedToken from the STS.

5679 Lines (R018)-(R0134) contain the WS-Security o:Security header in the SOAP s:Header.

5680 Lines (R023)-(R036) contain an **sc:DerivedKeyToken** that may be used to **construct the derived**
5681 **signing key DKT4(K2)**, which uses the Client-generated ephemeral key, K2, that the Service received in
5682 the Client request (M028)-(M045) above, and is now used in the response to the client, similar to the way
5683 the STS used K1 to respond to the Client, except that in this case the Service will use derived keys
5684 DKT4(K2) and DKT5(K2) in addition to K2 in the response. This sc:DerivedKeyToken contains a WS-
5685 Security o:SecurityTokenReference (R024)-(R032) that uses the WS-Security 1.1 mechanism
5686 EncryptedKeySHA1 to identify the Client-generated ephemeral key, K2, as the key to use to derive
5687 DKT4(K2) along with the sc:Nonce provided (R035) (and the label as described in
5688 [\[WS_SECURE_CONV\]](#)).

5689 Lines (R037)-(R048) contain a **second sc:DerivedKeyToken** that may be used by the Client to
5690 **construct the derived encryption key, DKT5(K2)**. The same EncryptedKeySHA1 mechanism is used
5691 by the Client to construct DKT5(K2) as described for DKT4(K2).

5692 Lines (R049)-(R051) contain an **e:ReferenceList** containing an e:DataReference to the EncryptedData
5693 element in the s:Body (R0136).

5694 Lines (R052)-(R055) contain 2 WS-Security 1.1 k:SignatureConfirmation elements confirming the dsig
5695 Signatures that were in the client request above (M0203) on the endorsing signature and (M0177) on the
5696 message signature of the Client request. The dsig SignatureValues compare respectively to assure the
5697 Client that the data from the Client request was processed and is what is being referred to in this
5698 response.

5699 Lines (R056)-(R0134) contain the message signature, which is signed as referenced in the dsig KeyInfo
5700 (R0128)-(R0132) by DKT4(K2), which may be constructed as described above using the
5701 sc:DerivedKeyToken element (R023)-(R036).

5702 Lines (R0137)-(R0151) contain the Service response payload to the Client which may be decrypted using
5703 DKT5(K2) using the sc:DerivedKeyToken element (R037)-(R048).

5704 2.4 Secure Conversation Scenarios

5705 2.4.1 (WSS 1.0) Secure Conversation bootstrapped by Mutual 5706 Authentication with X.509 Certificates

5707 This scenario corresponds to the situation where both parties have an X.509 certificate (and public/private
5708 key pair). Because of the volume of messages from each source, the Requestor/Initiator uses WS-
5709 SecureConversation to establish a new session key and uses the session key for integrity and/or
5710 confidentiality protection. This improves performance, by using less expensive symmetric key operations
5711 and improves security by reducing the exposure of the long term secret.

5712 The recipient models this scenario by using the symmetric binding that includes a
5713 SecureConversationToken assertion to describe the token type accepted by this endpoint. This token
5714 assertion further contains a bootstrap policy to indicate the security binding that is used by requestors that
5715 want a security context token issued by this service. The bootstrap policy affects the Request Security
5716 Token Request (RST) and Request Security Token Request Response (RSTR) messages sent between
5717 the Initiator and the Recipient to establish the security context. It is modeled by use of an asymmetric
5718 binding assertion for this scenario because both parties mutually authenticate each other using their
5719 X.509 certificates.

5720 The message level policies cover a different scope of the web service definition than the security binding
5721 level policy and so appear as separate policies and are attached at Message Policy Subject. These are
5722 shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint
5723 subject (WSS10SecureConversation_policy) and two (WSS10SecureConversation_input_policy,
5724 WSS10SecureConversation_output_policy) with message subjects to achieve this use case.

5725

```
5726 (P001) <wsp:Policy wsu:Id="WSS10SecureConversation_policy">  
5727 (P002)   <wsp:ExactlyOne>  
5728 (P003)     <wsp>All>  
5729 (P004)       <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-  
5730 securitypolicy/200702">  
5731 (P005)         <wsp:Policy>  
5732 (P006)           <sp:ProtectionToken>  
5733 (P007)             <wsp:Policy>  
5734 (P008)               <sp:SecureConversationToken  
5735 sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken  
5736 /AlwaysToRecipient">  
5737 (P009)                 <wsp:Policy>  
5738 (P010)                   <sp:RequireDerivedKeys/>  
5739 (P011)                   <sp:BootstrapPolicy>  
5740 (P012)                   <wsp:Policy>  
5741 (P013)                   <wsp:ExactlyOne>  
5742 (P014)                   <wsp>All>  
5743 (P015)                     <sp:AsymmetricBinding  
5744 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
5745 (P016)                       <wsp:Policy>  
5746 (P017)                         <sp:InitiatorToken>  
5747 (P018)                           <wsp:Policy>  
5748 (P019)                             <sp:X509Token  
5749 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/  
5750 AlwaysToRecipient">  
5751 (P020)                               <wsp:Policy>  
5752 (P021)                                 <sp:WssX509V3Token10/>  
5753 (P022)                                   </wsp:Policy>
```

```

5754 (P023) </sp:X509Token>
5755 (P024) </wsp:Policy>
5756 (P025) </sp:InitiatorToken>
5757 (P026) <sp:RecipientToken>
5758 (P027) <wsp:Policy>
5759 (P028) <sp:X509Token
5760 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
5761 AlwaysToInitiator">
5762 (P029) <wsp:Policy>
5763 (P030) <sp:WssX509V3Token10/>
5764 (P031) </wsp:Policy>
5765 (P032) </sp:X509Token>
5766 (P033) </wsp:Policy>
5767 (P034) </sp:RecipientToken>
5768 (P035) <sp:AlgorithmSuite>
5769 (P036) <wsp:Policy>
5770 (P037) <sp:TripleDesRsa15/>
5771 (P038) </wsp:Policy>
5772 (P039) </sp:AlgorithmSuite>
5773 (P040) <sp:Layout>
5774 (P041) <wsp:Policy>
5775 (P042) <sp:Strict/>
5776 (P043) </wsp:Policy>
5777 (P044) </sp:Layout>
5778 (P045) <sp:IncludeTimestamp/>
5779 (P046) <sp:OnlySignEntireHeadersAndBody/>
5780 (P047) </wsp:Policy>
5781 (P048) </sp:AsymmetricBinding>
5782 (P049) <sp:Wss10>
5783 (P050) <wsp:Policy>
5784 (P051) <sp:MustSupportRefKeyIdentifier/>
5785 (P052) </wsp:Policy>
5786 (P053) </sp:Wss10>
5787 (P054) <sp:SignedParts>
5788 (P055) <sp:Body/>
5789 (P056) <sp:Header Name="Action"
5790 Namespace="http://www.w3.org/2005/08/addressing"/>
5791 (P057) </sp:SignedParts>
5792 (P058) <sp:EncryptedParts>
5793 (P059) <sp:Body/>
5794 (P060) </sp:EncryptedParts>
5795 (P061) </wsp:All>
5796 (P062) </wsp:ExactlyOne>
5797 (P063) </wsp:Policy>
5798 (P064) </sp:BootstrapPolicy>
5799 (P065) </wsp:Policy>
5800 (P066) </sp:SecureConversationToken>
5801 (P067) </wsp:Policy>
5802 (P068) </sp:ProtectionToken>
5803 (P069) <sp:AlgorithmSuite>
5804 (P070) <wsp:Policy>
5805 (P071) <sp:Basic256/>
5806 (P072) </wsp:Policy>
5807 (P073) </sp:AlgorithmSuite>
5808 (P074) <sp:Layout>
5809 (P075) <wsp:Policy>
5810 (P076) <sp:Strict/>
5811 (P077) </wsp:Policy>
5812 (P078) </sp:Layout>
5813 (P079) <sp:IncludeTimestamp/>
5814 (P080) <sp:OnlySignEntireHeadersAndBody/>
5815 (P081) </wsp:Policy>
5816 (P082) </sp:SymmetricBinding>
5817 (P083) <sp:Trust13>

```

```

5818 (P084) <wsp:Policy>
5819 (P085) <sp:RequireClientEntropy/>
5820 (P086) <sp:RequireServerEntropy/>
5821 (P087) </wsp:Policy>
5822 (P088) </sp:Trust13>
5823 (P089)
5824 (P090) </wsp:All>
5825 (P091) </wsp:ExactlyOne>
5826 (P092) </wsp:Policy>
5827
5828 (P093) <wsp:Policy wsu:Id="WSS10SecureConversation_input_policy">
5829 (P094) <wsp:ExactlyOne>
5830 (P095) <wsp:All>
5831 (P096) <sp:SignedParts>
5832 (P097) <sp:Header Name="Action"
5833 <sp:Header Name="To"
5834 <sp:Header Name="MessageID"
5835 <sp:Header Name="MessageID"
5836 <sp:Header Name="MessageID"
5837 <sp:Header Name="MessageID"
5838 <sp:Header Name="MessageID"
5839 <sp:Header Name="MessageID"
5840 <sp:Header Name="MessageID"
5841 <sp:Header Name="MessageID"
5842 <sp:Header Name="MessageID"
5843 <sp:Header Name="MessageID"
5844 <sp:Header Name="MessageID"
5845 <sp:Header Name="MessageID"
5846 <sp:Header Name="MessageID"
5847 <sp:Header Name="MessageID"
5848 <sp:Header Name="MessageID"
5849 <sp:Header Name="MessageID"
5850 <sp:Header Name="MessageID"
5851 <sp:Header Name="MessageID"
5852 <sp:Header Name="MessageID"
5853 <sp:Header Name="MessageID"
5854 <sp:Header Name="MessageID"
5855 <sp:Header Name="MessageID"
5856 <sp:Header Name="MessageID"
5857 <sp:Header Name="MessageID"
5858 <sp:Header Name="MessageID"
5859 <sp:Header Name="MessageID"
5860 <sp:Header Name="MessageID"
5861 <sp:Header Name="MessageID"
5862 <sp:Header Name="MessageID"
5863 <sp:Header Name="MessageID"
5864 <sp:Header Name="MessageID"
5865 <sp:Header Name="MessageID"
5866 <sp:Header Name="MessageID"
5867 <sp:Header Name="MessageID"
5868 <sp:Header Name="MessageID"
5869 <sp:Header Name="MessageID"
5870 <sp:Header Name="MessageID"
5871 <sp:Header Name="MessageID"
5872 <sp:Header Name="MessageID"
5873 <sp:Header Name="MessageID"
5874 <sp:Header Name="MessageID"
5875 <sp:Header Name="MessageID"
5876 <sp:Header Name="MessageID"

```

5861 Lines (P004) – (P082) indicate that the service uses the symmetric security binding to protect messages,
5862 using a Security Context Token (Lines (P008) – (P066)) to sign messages in both directions. The actual
5863 basis for the signatures should be keys derived from that Security Context Token as stated by the
5864 RequireDerivedKey assertion in Line (P010). Messages must include a Timestamp element (Line (P079))
5865 and the signature value must be calculated over the entire body and header elements (Line (P080)).

5866 Lines (P083) – (P088) contain the Trust13 assertion which defines the requirements for the WS-Trust
5867 related message exchange as part of the SC bootstrap. Line (P085) indicates that the Initiator (Client) has
5868 to provide entropy to be used as key material for the requested proof token. Line (P086) requires the
5869 same from the recipient (Server) which results in computed key from both client and server entropy
5870 values.

5871 Lines (P011) – (P065) contain the bootstrap policy for the initial creation of the security context between
5872 the communication parties before it is being used. It contains an AsymmetricBinding assertion (Lines
5873 (P015) – (P048)) which indicates that the initiator's (WS-Security 1.0 X.509 Certificate) token (Lines
5874 (P017) – (P025)) used by the recipient to verify the signature in the RST must be included in the message
5875 (P019). The exchange of entropy and key material in the body of the RST and RSTR messages is
5876 protected by the EncryptedParts assertion of the bootstrap policy in lines (P058) – (P061).

5877 According to the MustSupportKeyRefIdentifier assertions in Line (P051), an X.509 Key Identifier must be
5878 used to identify the token. Lines (P054) – (P057) require that the body and the WS-Addressing Action
5879 header is signed on each message (RST, RSTR) within the bootstrap process.

5880 An example of an RST message sent from the initiator to the recipient according to the bootstrap policy
5881 defined by this policy is as follows:

```
5882 (M001) <?xml version="1.0" encoding="utf-8" ?>
5883 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
5884         xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="...">
5885 (M003)   <soap:Header>
5886 (M004)     <wsa:Action wsu:Id="action">
5887 (M005)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
5888 (M006)     </wsa:Action>
5889 (M007)     <wsse:Security>
5890 (M008)       <wsu:Timestamp wsu:Id="timestamp">
5891 (M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5892 (M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5893 (M011)       </wsu:Timestamp>
5894 (M012)       <wsse:BinarySecurityToken wsu:Id="clientToken"
5895             Value="..."#X509v3" EncodingType="..."#Base64Binary">
5896 (M013)         MIICZDCCAc2gAwIBAgIRALSOLzt7...
5897 (M014)       </wsse:BinarySecurityToken>
5898 (M015)       <ds:Signature xmlns:ds="...">
5899 (M016)         <ds:SignedInfo>
5900 (M017)           <ds:CanonicalizationMethod
5901             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5902 (M018)           <ds:SignatureMethod
5903             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
5904 (M019)           <ds:Reference URI="#action">
5905 (M020)             <ds:Transforms>
5906 (M021)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
5907                 exc-c14n#" />
5908 (M022)             </ds:Transforms>
5909 (M023)             <ds:DigestMethod
5910                 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
5911 (M024)             <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
5912 (M025)           </ds:Reference>
5913 (M026)           <ds:Reference URI="#timestamp">...</ds:Reference>
5914 (M027)           <ds:Reference URI="#body">...</ds:Reference>
5915 (M028)         </ds:SignedInfo>
5916 (M029)         <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
5917 (M030)         <ds:KeyInfo>
5918 (M031)           <wsse:SecurityTokenReference>
5919 (M032)             <wsse:Reference URI="#clientToken"
5920                 Value="..."#X509v3" />
5921 (M033)           </wsse:SecurityTokenReference>
5922 (M034)           </ds:KeyInfo>
5923 (M035)         </ds:Signature>
5924 (M036)         <xenc:EncryptedKey>
5925 (M037)           <xenc:EncryptionMethod Algorithm="..."#rsa-1_5" />
5926 (M038)           <ds:KeyInfo>
5927 (M039)             <wsse:SecurityTokenReference>
5928 (M040)               <wsse:KeyIdentifier
5929 (M041)                 Value="..."#X509v3SubjectKeyIdentifier">AtETQ...
5930 (M042)               </wsse:KeyIdentifier>
5931 (M043)             </wsse:SecurityTokenReference>
5932 (M044)           </ds:KeyInfo>
5933 (M045)           <xenc:CipherData>
5934 (M046)             <xenc:CipherValue>
5935 (M047)               <!-- encrypted key -->
5936 (M048)             </xenc:CipherValue>
5937 (M049)           </xenc:CipherData>
5938 (M050)           <xenc:ReferenceList>
5939 (M051)             <xenc:DataReference URI="#request" />
```

```

5940 (M052) </xenc:ReferenceList>
5941 (M053) </xenc:EncryptedKey>
5942 (M054) </wsse:Security>
5943 (M055) </soap:Header>
5944 (M056) <soap:Body wsu:Id="body">
5945 (M057) <xenc:EncryptedData Id="request" Type="...#Content">
5946 (M058) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
5947 (M059) <xenc:CipherData>
5948 (M060) <xenc:CipherValue>
5949 (M061) <!-- encrypted RST request -->
5950 (M062) <!-- ### Begin unencrypted RST request
5951 (M063) <wst:RequestSecurityToken>
5952 (M064) <wst:TokenType>
5953 (M065) http://docs.oasis-open.org/ws-sx/ws-secureconversation/
5954 200512/sct
5955 (M066) </wst:TokenType>
5956 (M067) <wst:RequestType>
5957 (M068) http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
5958 (M069) </wst:RequestType>
5959 (M070) <wsp:AppliesTo
5960 xmlns:wsp="http://www.w3.org/ns/ws-policy">
5961 (M071) <wsp:EndpointReference>
5962 (M072) <wsp:Address>
5963 (M073) http://acme.com/ping/
5964 (M074) </wsp:Address>
5965 (M075) </wsp:EndpointReference>
5966 (M076) </wsp:AppliesTo>
5967 (M077) <wst:Entropy>
5968 (M078) <wst:BinarySecret>cr9b0cb1wCEyY9ehaGK33e41h9s=
5969 (M079) </wst:BinarySecret>
5970 (M080) </wst:Entropy>
5971 (M081) </wst:RequestSecurityToken>
5972 (M082) ### End unencrypted RST request -->
5973 (M083) </xenc:CipherValue>
5974 (M084) </xenc:CipherData>
5975 (M085) </xenc:EncryptedData>
5976 (M086) </soap:Body>
5977 (M087) </soap:Envelope>

```

- 5978 Line (M005) indicates to the recipient that the SCT Binding of WS-Trust is used.
- 5979 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 5980 Lines (M012) – (M014) contain the initiator’s X.509 token as required by the InitiatorToken assertion’s value (“.../AlwaysToRecipient”).
- 5981
- 5982 Lines (M015) – (M035) hold the message signature.
- 5983 Lines (M036) – (M053) hold the encrypted symmetric key used to encrypt the RST request in the body of the message according to the bootstrap policy. It contains a Subject Key Identifier of the X.509 Certificate used to encrypt the key and not the certificate itself as required by the RecipientToken assertion’s value (“.../Never”) in (P028).
- 5984
- 5985
- 5986
- 5987 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 5988
- 5989 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.
- 5990
- 5991 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.
- 5992
- 5993 Lines (M056) – (M086) hold the Security Token Reference pointing to the initiators X.509 certificate.
- 5994 Lines (M038) – (M058) contain the SOAP Body of the message with the encrypted RST element. Lines (M062) – (M082) show the unencrypted content of the RST request. It specifies the Token Type (Lines (M064) – (M066)) and the Request Type (Lines (M067) – (M069)). According to the Trust13 assertion, it also includes entropy provided by the initiator as indicated by Lines (M077) – (M080).
- 5995
- 5996
- 5997

5998 An example of an RSTR message sent from the recipient to the initiator according to the bootstrap policy
 5999 defined by this policy is as follows:

```

6000 (M001) <?xml version="1.0" encoding="UTF-8"?>
6001 (M002) <soap:Envelope xmlns:soap="..." xmlns:wst="..." xmlns:wsc="..."
6002         xmlns:xenc="...">
6003 (M003)   <soap:Header>
6004 (M004)     <wsa:Action wsu:Id="action">
6005 (M005)       http://docs.oasis-open.org/ws-sx/ws-
6006 trust/200512/RSTRC/IssueFinal
6007 (M006)     </wsa:Action>
6008 (M007)     <wsse:Security>
6009 (M008)       <wsu:Timestamp wsu:Id="timestamp">
6010 (M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6011 (M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6012 (M011)       </wsu:Timestamp>
6013 (M012)       <wsse:BinarySecurityToken wsu:Id="serviceToken"
6014             ValueType="...#X509v3" EncodingType="...#Base64Binary">
6015             MIIASDPiOASDsaöAgIRALSOLzt7...
6016 (M014)       </wsse:BinarySecurityToken>
6017 (M015)       <ds:Signature xmlns:ds="...">
6018 (M016)         <ds:SignedInfo>
6019 (M017)           <ds:CanonicalizationMethod
6020                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
6021 (M018)           <ds:SignatureMethod
6022                 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
6023 (M019)           <ds:Reference URI="#action">
6024 (M020)             <ds:Transforms>
6025 (M021)               <ds:Transform
6026                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
6027 (M022)             </ds:Transforms>
6028 (M023)           <ds:DigestMethod
6029                 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
6030 (M024)           <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6031 (M025)           </ds:Reference>
6032 (M026)           <ds:Reference URI="#timestamp">...</ds:Reference>
6033 (M027)           <ds:Reference URI="#body">...</ds:Reference>
6034 (M028)         </ds:SignedInfo>
6035 (M029)         <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6036 (M030)         <ds:KeyInfo>
6037 (M031)           <wsse:SecurityTokenReference>
6038 (M032)             <wsse:Reference URI="#myToken" ValueType="...#X509v3" />
6039 (M033)           </wsse:SecurityTokenReference>
6040 (M034)         </ds:KeyInfo>
6041 (M035)       </ds:Signature>
6042 (M036)       <xenc:EncryptedKey>
6043 (M037)         <xenc:EncryptionMethod Algorithm="...#rsa-1_5" />
6044 (M038)         <ds:KeyInfo>
6045 (M039)           <wsse:SecurityTokenReference>
6046 (M040)             <wsse:KeyIdentifier
6047 (M041)               ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
6048 (M042)             </wsse:KeyIdentifier>
6049 (M043)           </wsse:SecurityTokenReference>
6050 (M044)         </ds:KeyInfo>
6051 (M045)         <xenc:CipherData>
6052 (M046)           <xenc:CipherValue>
6053 (M047)             <!-- encrypted key -->
6054 (M048)           </xenc:CipherValue>
6055 (M049)         </xenc:CipherData>
6056 (M050)         <xenc:ReferenceList>
6057 (M051)           <xenc:DataReference URI="#response" />
6058 (M052)         </xenc:ReferenceList>
6059 (M053)       </xenc:EncryptedKey>
6060 (M054)     </wsse:Security>
6061 (M055)   </soap:Header>

```

```

6062 (M056) <soap:Body wsu:Id="body">
6063 (M057) <xenc:EncryptedData Id="response" Type="...#Content">
6064 (M058) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
6065 (M059) <xenc:CipherData>
6066 (M060) <xenc:CipherValue>
6067 (M061) <!-- encrypted RSTR -->
6068 (M062) <!-- ### Begin unencrypted RSTR
6069 (M063) <wst:RequestSecurityTokenResponseCollection>
6070 (M064) <wst:RequestSecurityTokenResponse>
6071 (M065) <wst:RequestedSecurityToken>
6072 (M066) <wsc:SecurityContextToken>
6073 (M067) <wsc:Identifier>uuid:...</wsc:Identifier>
6074 (M068) </wsc:SecurityContextToken>
6075 (M069) </wst:RequestedSecurityToken>
6076 (M070) <wst:RequestedProofToken>
6077 (M071) <xenc:EncryptedKey Id="ek049ea390c90011dbba4e00304852867e">
6078 (M072) <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
6079 (M073) <ds:KeyInfo>
6080 (M074) <wsse:SecurityTokenReference>
6081 (M075) <wsse:KeyIdentifier
Value="...#X509SubjectKeyIdentifier"
EncodingType="...#Base64Binary">
6082 (M076) Wjw2gDCBye6NJAh0lPCyUldvTN8=
6083 (M077) </wsse:KeyIdentifier>
6084 (M078) </wsse:SecurityTokenReference>
6085 (M079) </ds:KeyInfo>
6086 (M080) <xenc:CipherData>
6087 (M081) <xenc:CipherValue>hhx9TcaVL6XwBN1...</xenc:CipherValue>
6088 (M082) </xenc:CipherData>
6089 (M083) </xenc:EncryptedKey>
6090 (M084) </wst:RequestedProofToken>
6091 (M085) <wsp:AppliesTo
xmlns:wsp="http://www.w3.org/ns/ws-policy">
6092 (M086) <wsp:EndpointReference>
6093 (M087) <wsp:Address>
6094 (M088) http://acme.com/ping/
6095 (M089) </wsp:Address>
6096 (M090) </wsp:EndpointReference>
6097 (M091) </wsp:AppliesTo>
6098 (M092) <wst:Entropy>
6099 (M093) <wst:BinarySecret>asdhjwkjqwe123498SDFALasd=
6100 (M094) </wst:BinarySecret>
6101 (M095) </wst:Entropy>
6102 (M096) </wst:RequestSecurityTokenResponse>
6103 (M097) </wst:RequestSecurityTokenResponseCollection>
6104 (M098) ### End unencrypted RSTR -->
6105 (M099) </xenc:CipherValue>
6106 (M100) </xenc:CipherData>
6107 (M101) </xenc:EncryptedData>
6108 (M102) </soap:Body>
6109 (M103) </soap:Envelope>

```

- 6114 Line (M005) indicates to the initiator that this is the final response to the RST in an RSTRC.
- 6115 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 6116 Lines (M012) – (M014) contain the recipient's X.509 token as required by the RecipientToken assertion's value (".../AlwaysToInitiator").
- 6117
- 6118 Lines (M015) – (M035) hold the message signature.
- 6119 Lines (M036) – (M053) hold the encrypted symmetric key used to encrypt the RSTR response in the body of the message according to the bootstrap policy.
- 6120
- 6121 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 6122

6123 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp
6124 assertion definition.

6125 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts
6126 assertion.

6127 Lines (M031) – (M033) hold the Security Token Reference pointing to the requestor’s X.509 certificate.

6128 Lines (M056) – (M102) contain the SOAP Body of the message with the encrypted RSTR collection
6129 element. Commented out is the unencrypted form of the RSTR collection in Lines (M063) – (M097), which
6130 includes one RSTR (Lines (M064) – (M096)). Lines (M065) – (M069) contain the new Security Context
6131 Token. The accompanying RequestedProofToken in Lines (M070) – (M084) include the encrypted secret
6132 key (Lines (M071) – (M083)) of the security context. The key is encrypted using the initiator’s public key
6133 that was already used to verify its signature in the incoming request.

6134 According to the Trust13 assertion, the response includes entropy provided by the recipient as indicated
6135 by Lines (M092) – (M095).

6136 An Example of an SCT-secured application message sent from the initiator to the recipient according to
6137 the message input policy (WSS10SecureConversation_input_policy) is as follows:

```

6138 (M001) <?xml version="1.0" encoding="UTF-8"?>
6139 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
6140 (M003)   xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="..." xmlns:wsm="...">
6141 (M004)   <soap:Header>
6142 (M005)     <wsa:To wsu:Id="to">http://acme.com/ping/</wsa:To>
6143 (M006)     <wsa:MessageID wsu:Id="msgid">
6144 (M007)       http://acme.com/guid/12318731edh-CA47-1067-B31D-10662DA
6145 (M008)     </wsa:MessageID>
6146 (M009)     <wsa:Action wsu:Id="action">urn:Ping</wsa:Action>
6147 (M010)     <wsse:Security>
6148 (M011)       <wsu:Timestamp wsu:Id="timestamp">
6149 (M012)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6150 (M013)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6151 (M014)       </wsu:Timestamp>
6152 (M015)       <wsc:SecurityContextToken wsu:Id="SCT">
6153 (M016)         <wsc:Identifier>uuid:91f50600-60cc-11da-8e67-000000000000
6154 (M017)         </wsc:Identifier>
6155 (M018)       </wsc:SecurityContextToken>
6156 (M019)       <wsc:DerivedKeyToken wsu:Id="DKT">
6157 (M020)         <wsse:SecurityTokenReference>
6158 (M021)           <wsse:Reference URI="#SCT" ValueType="http://docs.oasis-
6159 (M022)             open.org/ws-sx/ws-secureconversation/200512/sct"/>
6160 (M023)         </wsse:SecurityTokenReference>
6161 (M024)         <wsc:Offset>0</wsc:Offset>
6162 (M025)         <wsc:Length>24</wsc:Length>
6163 (M026)         <wsc:Label>
6164 (M027)           WSSecure ConversationWSSecure Conversation
6165 (M028)         </wsc:Label>
6166 (M029)         <wsc:Nonce>yln04kFBJesy2U2SQL6ezI3SCak=</wsc:Nonce>
6167 (M030)       </wsc:DerivedKeyToken>
6168 (M031)       <ds:Signature xmlns:ds="...">
6169 (M032)         <ds:SignedInfo>
6170 (M033)           <ds:CanonicalizationMethod
6171 (M034)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6172 (M035)           <ds:SignatureMethod
6173 (M036)             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6174 (M037)           <ds:Reference URI="#msgid">
6175 (M038)             <ds:Transforms>
6176 (M039)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
6177 (M040)                 exc-c14n#"/>
6178 (M041)             </ds:Transforms>
6179 (M042)           <ds:DigestMethod
6180 (M043)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6181 (M044)           <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6182 (M045)         </ds:Reference>

```

```

6183 (M039) <ds:Reference URI="#to">...</ds:Reference>
6184 (M040) <ds:Reference URI="#action">...</ds:Reference>
6185 (M041) <ds:Reference URI="#timestamp">...</ds:Reference>
6186 (M042) <ds:Reference URI="#body">...</ds:Reference>
6187 (M043) </ds:SignedInfo>
6188 (M044) <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6189 (M045) <ds:KeyInfo>
6190 (M046) <wsse:SecurityTokenReference>
6191 (M047) <wsse:Reference URI="#DKT" ValueType="http://docs.oasis-
6192 open.org/ws-sx/ws-secureconversation/200512/dk"/>
6193 (M048) </wsse:SecurityTokenReference>
6194 (M049) </ds:KeyInfo>
6195 (M050) </ds:Signature>
6196 (M051) </wsse:Security>
6197 (M052) </soap:Header>
6198 (M053) <soap:Body wsu:Id="body">
6199 (M054) <pns:Ping xmlns:pns="http://tempuri.org/">
6200 (M055) <pns:Text>abc</pns:Text>
6201 (M056) </pns:Ping>
6202 (M057) </soap:Body>
6203 (M058) </soap:Envelope>

```

- 6204
- 6205 Lines (M004) – (M008) contain the WS-Addressing headers according to the UsingAddressing assertion.
- 6206 Lines (M010) – (M013) hold the Timestamp as specified by the IncludeTimestamp assertion within the
- 6207 SymmetricBinding assertion.
- 6208 Lines (M014) – (M017) contain the Security Context Token which has been issued by the recipient in the
- 6209 previous message. Based on this token, the initiator included a Derived Key Token (Lines (M018) –
- 6210 (M027)) as indicated by the RequireDerivedKey assertion in the policy.
- 6211 Lines (M028) – (M050) hold the message signature that uses the Derived Key Token (Lines (M046) –
- 6212 (M048)) to sign the WS-Addressing headers (Lines (M032) – (M040)), the timestamp (Line (M041)) and
- 6213 the body (Line (M042)) of the message, according to the SignedParts assertion of the message input
- 6214 policy (WSS10SecureConversation_input_policy).

6215 **3 Conformance**

6216 This document contains non-normative examples of usage of WS-SecurityPolicy and other related
6217 specifications.

6218 Therefore **there are no conformance statements that apply to this document.**

6219 Refer to the referenced specifications [[References](#)], which will individually contain conformance
6220 requirements for WS-SecurityPolicy and related specifications.

6221

6222

A. Acknowledgements

6223 The following individuals have participated in the creation of this specification and are gratefully
6224 acknowledged:

6225 **Original Contributors:**

6226 Ashok Malhotra, Oracle

6227 Prateek Mishra, Oracle

6228 Ramana Turlapati, Oracle

6229 **Participants:**

6230 Don Adams, Tibco

6231 Moushmi Banerjee, Oracle

6232 Symon Chang, Oracle

6233 Henry Chung, IBM

6234 Paul Cotton, Microsoft

6235 Marc Goodner, Microsoft

6236 Martin Gudgin, Microsoft

6237 Tony Gullotta, SOA

6238 Jiandong Guo, Sun

6239 Frederick Hirsch, Nokia

6240 Chris Kaler, Microsoft

6241 Rich Levinson, Oracle

6242 Chunlong Liang, IBM

6243 Hal Lockhart, Oracle

6244 Mike Lyons, Layer 7

6245 Ashok Malhotra, Oracle

6246 Carlo Milono, Tibco

6247 Prateek Mishra, Oracle

6248 Anthony Nadalin, Microsoft

6249 Martin Raeppe, SAP AG

6250 Bruce Rich, IBM

6251 Ramana Turlapati, Oracle

6252 Greg Whitehead, Hewlett-Packard

6253 Ching-Yun (C.Y.) Chao, IBM

6254

6255 **TC Members during the development of this specification:**

6256 Don Adams, Tibco Software Inc.

6257 Jan Alexander, Microsoft Corporation

6258 Steve Anderson, BMC Software

6259 Donal Arundel, IONA Technologies

6260 Howard Bae, Oracle Corporation

6261 Abbie Barbir, Nortel Networks Limited

6262 Charlton Barreto, Adobe Systems

6263 Michael Botha, Software AG, Inc.

6264 Toufic Boubez, Layer 7 Technologies Inc.

6265 Norman Brickman, Mitre Corporation
6266 Melissa Brumfield, Booz Allen Hamilton
6267 Geoff Bullen, Microsoft Corporation
6268 Lloyd Burch, Novell
6269 Scott Cantor, Internet2
6270 Greg Carpenter, Microsoft Corporation
6271 Steve Carter, Novell
6272 Symon Chang, Oracle Corporation Ching-Yun (C.Y.) Chao, IBM
6273 Martin Chapman, Oracle Corporation
6274 Kate Cherry, Lockheed Martin
6275 Henry (Hyenvui) Chung, IBM
6276 Luc Clement, Systinet Corp.
6277 Paul Cotton, Microsoft Corporation
6278 Glen Daniels, Sonic Software Corp.
6279 Peter Davis, Neustar, Inc.
6280 Duane DeCouteau, Veterans Health Administration
6281 Martijn de Boer, SAP AG
6282 Werner Dittmann, Siemens AG
6283 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory
6284 Fred Dushin, IONA Technologies
6285 Petr Dvorak, Systinet Corp.
6286 Colleen Evans, Microsoft Corporation
6287 Ruchith Fernando, WSO2
6288 Mark Fussell, Microsoft Corporation
6289 Vijay Gajjala, Microsoft Corporation
6290 Marc Goodner, Microsoft Corporation
6291 Hans Granqvist, VeriSign
6292 Martin Gudgin, Microsoft Corporation
6293 Tony Gullotta, SOA Software Inc.
6294 Jiandong Guo, Sun Microsystems
6295 Phillip Hallam-Baker, VeriSign
6296 Patrick Harding, Ping Identity Corporation
6297 Heather Hinton, IBM
6298 Frederick Hirsch, Nokia Corporation
6299 Jeff Hodges, Neustar, Inc.
6300 Will Hopkins, Oracle Corporation
6301 Alex Hristov, Otecia Incorporated
6302 John Hughes, PA Consulting
6303 Diane Jordan, IBM
6304 Venugopal K, Sun Microsystems
6305 Chris Kaler, Microsoft Corporation
6306 Dana Kaufman, Forum Systems, Inc.
6307 Paul Knight, Nortel Networks Limited
6308 Ramanathan Krishnamurthy, IONA Technologies
6309 Christopher Kurt, Microsoft Corporation
6310 Kelvin Lawrence, IBM
6311 Hubert Le Van Gong, Sun Microsystems
6312 Jong Lee, Oracle Corporation
6313 Rich Levinson, Oracle Corporation
6314 Tommy Lindberg, Dajeil Ltd.
6315 Mark Little, JBoss Inc.
6316 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.
6317 Eve Maler, Sun Microsystems
6318 Ashok Malhotra, Oracle Corporation
6319 Anand Mani, CrimsonLogic Pte Ltd
6320 Jonathan Marsh, Microsoft Corporation
6321 Robin Martherus, Oracle Corporation

6322 Miko Matsumura, Infravio, Inc.
6323 Gary McAfee, IBM
6324 Michael McIntosh, IBM
6325 John Merrells, Sxip Networks SRL
6326 Jeff Mischkinsky, Oracle Corporation
6327 Prateek Mishra, Oracle Corporation
6328 Bob Morgan, Internet2
6329 Vamsi Motukuru, Oracle Corporation
6330 Raajmohan Na, EDS
6331 Anthony Nadalin, IBM
6332 Andrew Nash, Reactivity, Inc.
6333 Eric Newcomer, IONA Technologies
6334 Duane Nickull, Adobe Systems
6335 Toshihiro Nishimura, Fujitsu Limited
6336 Rob Philpott, RSA Security
6337 Denis Pilipchuk, Oracle Corporation.
6338 Darren Platt, Ping Identity Corporation
6339 Martin Raepple, SAP AG
6340 Nick Ragouzis, Enosis Group LLC
6341 Prakash Reddy, CA
6342 Alain Regnier, Ricoh Company, Ltd.
6343 Irving Reid, Hewlett-Packard
6344 Bruce Rich, IBM
6345 Tom Rutt, Fujitsu Limited
6346 Maneesh Sahu, Actional Corporation
6347 Frank Siebenlist, Argonne National Laboratory
6348 Joe Smith, Apani Networks
6349 Davanum Srinivas, WSO2
6350 David Staggs, Veterans Health Administration
6351 Yakov Sverdlov, CA
6352 Gene Thurston, AmberPoint
6353 Victor Valle, IBM
6354 Asir Vedamuthu, Microsoft Corporation
6355 Greg Whitehead, Hewlett-Packard
6356 Ron Williams, IBM
6357 Corinna Witt, Oracle Corporation
6358 Kyle Young, Microsoft Corporation
6359

6361

B. Revision History

6362 [optional; should not be included in OASIS Standards]

6363

Revision	Date	Editor	Changes Made
0.09	23-Feb-07	Rich Levinson	Updated doc format to latest OASIS template, added Symon Chang's encrypted UsernameToken scenario
0.10	6-Mar-07	Rich Levinson Tony Gullotta Symon Chang Martin Raepple	Added sample messages and explanatory text to several examples. Line numbered each example w Pxxx for the Policy, Mxxx for the sample message. Intent is to do all examples, this version is to get feedback along with progress as each example stands alone. Completed examples: 2.1.1.2, 2.1.2.1, 2.1.3, 2.1.4, 2.2.1, and 2.5.1. Partially completed examples that have sample messages: 2.1.1.1, 2.1.1.3, 2.3.1.2, 2.3.1.4, 2.3.1.5, and 2.3.2.2, 2.3.2.4, 2.3.2.5

6364