



WS-SecurityPolicy Examples

Committee Specification 01

24 November 2009

Specification URIs:

This 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>

Previous Version:

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-01.doc>
(Authoritative)
<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-01.pdf>
<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-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, BEA Systems Inc.
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® 1993–2009. 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.....	12
1.6	Interops and Sample Messages	12
2	Scenarios.....	13
2.1	UsernameToken	13
2.1.1	UsernameToken – no security binding.....	13
2.1.1.1	UsernameToken with plain text password	13
2.1.1.2	UsernameToken without password.....	14
2.1.1.3	UsernameToken with timestamp, nonce and password hash.....	15
2.1.2	Use of SSL Transport Binding.....	16
2.1.2.1	UsernameToken as supporting token	16
2.1.3	(WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, Encrypt.....	18
2.1.3.1	(WSS 1.0) Encrypted UsernameToken with X.509v3	21
2.1.4	(WSS 1.1), User Name with Certificates, Sign, Encrypt	24
2.2	X.509 Token Authentication Scenario Assertions	28
2.2.1	(WSS1.0) X.509 Certificates, Sign, Encrypt.....	28
2.2.2	(WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt.....	31
2.2.2.1	(WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption	34
2.2.3	(WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt.....	38
2.2.4	(WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt.....	41
2.3	SAML Token Authentication Scenario Assertions	47
2.3.1	WSS 1.0 SAML Token Scenarios	48
2.3.1.1	(WSS1.0) SAML1.1 Assertion (Bearer).....	48
2.3.1.2	(WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL	50
2.3.1.3	(WSS1.0) SAML1.1 Assertion (HK) over SSL.....	52
2.3.1.4	(WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional Encrypt	53
2.3.1.5	(WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt	59
2.3.2	WSS 1.1 SAML Token Scenarios	64
2.3.2.1	(WSS1.1) SAML 2.0 Bearer.....	64
2.3.2.2	(WSS1.1) SAML2.0 Sender Vouches over SSL.....	68
2.3.2.3	(WSS1.1) SAML2.0 HoK over SSL.....	70
2.3.2.4	(WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign, Encrypt	74
2.3.2.5	(WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt.....	79
2.4	Secure Conversation Scenarios	104
2.4.1	(WSS 1.0) Secure Conversation bootstrapped by Mutual Authentication with X.509 Certificates	104

3	Conformance	113
A.	Acknowledgements	114
B.	Revision History.....	115

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.

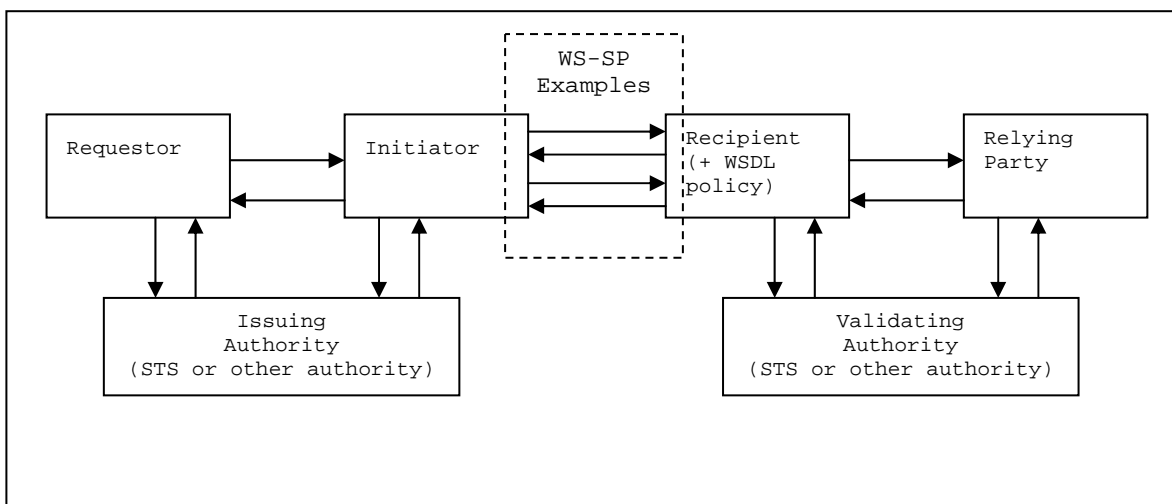
51 The purpose here is not to provide unequivocal answers to all questions regarding liability of
52 transactions, but to give a sense of how the structuring of a request message ties the participating entities
53 to the transaction. Depending on how the WS-SecurityPolicy technology is used, these “ties” 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] [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
249 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 250 [WSS11-SOAPMSG] [http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
251 [SOAPMessageSecurity.pdf](http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)
- 252 [WSS10-USERNAME] [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
253 [profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
- 254 [WSS11-USERNAME] [http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-](http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf)
255 [os-UsernameTokenProfile.pdf](http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf)
- 256 [WSS10-X509-PROFILE] [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
257 [profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
- 258 [WSS11-X509-PROFILE] [http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-](http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf)
259 [spec-os-x509TokenProfile.pdf](http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf)
- 260 [WSS10-SAML11-PROFILE] <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>
- 261 [WSS11-SAML1120-PROFILE] [http://www.oasis-open.org/committees/download.php/16768/wss-](http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-218-spec-os-SAMLSamlTokenProfile.pdf)
262 [v1.1-218-spec-os-SAMLSamlTokenProfile.pdf](http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-218-spec-os-SAMLSamlTokenProfile.pdf)
- 263 [WSS11-LIBERTY-SAML20-PROFILE]
264 [http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-](http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-security-mechanisms-saml-profile-v2.0.pdf)
265 [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)
- 266 [WSS-TC-ALL-TOKEN-PROFILES]
267 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- 268 [WS-SECURITYPOLICY] <http://www.oasis-open.org/specs/index.php#wssecpolv1.2>
- 269 [WS-SECURECONVERSATION] [http://docs.oasis-open.org/ws-sx/ws-](http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf)
270 [secureconversation/200512/ws-secureconversation-1.3-os.pdf](http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf)
- 271 [WS-TRUST] <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf>
- 272 [WS-POLICY] <http://www.w3.org/TR/ws-policy>
- 273 [WS-POLICYATTACHMENT] <http://www.w3.org/TR/ws-policy-attach>
- 274 [WSDL] <http://www.w3.org/TR/wsdl>
- 275 [SSL] <http://www.ietf.org/rfc/rfc2246.txt>
- 276 [SAML11-CORE] [http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)
277 [1.1.pdf](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)
- 278 [SAML20-CORE] <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>
- 279 [XML-DSIG] <http://www.w3.org/TR/xmlsig-core/>
- 280 [XML-ENCR] <http://www.w3.org/TR/xmlenc-core/>

281
282

283 1.4 Non-Normative References

- 284 WSS10-INTEROP-01: [http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-](http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf)
285 [06-merged-changes.pdf](http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf)
- 286
- 287 WSS10-INTEROP-02: [http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-](http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc)
288 [06-merged.doc](http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc)
- 289
- 290 WSS11-INTEROP-01: [http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-](http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc)
291 [draft-01.doc](http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc)
- 292
- 293 WSS10-KERBEROS-INTEROP: [http://www.oasis-open.org/committees/download.php/10991/wss-](http://www.oasis-open.org/committees/download.php/10991/wss-kerberos-interop.doc)
294 [kerberos-interop.doc](http://www.oasis-open.org/committees/download.php/10991/wss-kerberos-interop.doc)
- 295
- 296 WSS10-SAML11-INTEROP: [http://www.oasis-open.org/committees/download.php/7702/wss-saml-](http://www.oasis-open.org/committees/download.php/7702/wss-saml1-interop1-draft-12.doc)
297 [interop1-draft-12.doc](http://www.oasis-open.org/committees/download.php/7702/wss-saml1-interop1-draft-12.doc)
- 298
- 299 WSS11-SAML1120-INTEROP: [http://www.oasis-open.org/committees/download.php/16556/wss-saml2-](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)
300 [interop-draft-v4.doc](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)
- 301
- 302 WSSX-PRE-INTEROP: [http://www.oasis-](http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc)
303 [open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2](http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc)
304 [004-10.doc](http://www.oasis-open.org/committees/download.php/16357/Trust_SecureConversation_Interop.2004-10.doc)
- 305
- 306 WSSX-WSTR-WSSC-INTEROP: [http://www.oasis-open.org/committees/download.php/20954/ws-sx-](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc)
307 [interop-ed-10.doc](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc)
- 308
- 309 WS-SECURE-INTEROP: [http://www.oasis-](http://www.oasis-open.org/committees/document.php?document_id=28803&wg_abbrev=ws-sx)
310 [open.org/committees/document.php?document_id=28803&wg_abbrev=ws-sx](http://www.oasis-open.org/committees/document.php?document_id=28803&wg_abbrev=ws-sx)
- 311
- 312 WSI-SCM-SAMPLEAPPL: [http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-](http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc)
313 [04/SCMSecurityArchitectureWGD5.00.doc](http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc) (login required)
- 314
- 315

316 1.5 Specifications

317 1.6 Interops and Sample Messages

318 2 Scenarios

319 2.1 UsernameToken

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

322 2.1.1 UsernameToken – no security binding

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

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

329 2.1.1.1 UsernameToken with plain text password

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

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

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

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

341

```
342 (P001) <wsp:Policy>  
343 (P002) <sp:SupportingTokens>  
344 (P003) <wsp:Policy>  
345 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-  
346 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient"  
347 (P005) </wsp:Policy>  
348 (P006) </sp:SupportingTokens>  
349 (P007) </wsp:Policy>
```

350

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

352

```
353 (M001) <?xml version="1.0" encoding="utf-8" ?>  
354 (M002) <soap:Envelope xmlns:soap="...">  
355 (M003) <soap:Header>  
356 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">  
357 (M005) <wsse:UsernameToken>  
358 (M006) <wsse:Username>Chris</wsse:Username>  
359 (M007) <wsse:Password Type="http://docs.oasis-  
360 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-  
361 1.0#PasswordText">sirhC</wsse:Password>  
362 (M008) <wsse:Nonce EncodingType="...#Base64Binary"  
363 (M009) >pN...=</wsse:Nonce>
```

```

364 (M010) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
365 (M011) </wsse:UsernameToken>
366 (M012) </wsse:Security>
367 (M013) </soap:Header>
368 (M014) <soap:Body>
369 (M015) <Ping xmlns="http://xmlsoap.org/Ping">
370 (M016) <text>EchoString</text>
371 (M017) </Ping>
372 (M018) </soap:Body>
373 (M019) </soap:Envelope>

```

374

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

381

382 2.1.1.2 UsernameToken without password

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

389

```

390 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
391 (P002) <sp:SupportingTokens>
392 (P003) <wsp:Policy>
393 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
394 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
395 (P005) <wsp:Policy>
396 (P006) <sp:NoPassword/>
397 (P007) </wsp:Policy>
398 (P008) </sp:UsernameToken>
399 (P009) </wsp:Policy>
400 (P010) </sp:SupportingTokens>
401 (P011) </wsp:Policy>

```

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

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

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

406

```

407 (M001) <?xml version="1.0" encoding="utf-8" ?>
408 (M002) <soap:Envelope xmlns:soap="...">
409 (M003) <soap:Header>
410 (M004) <wsse:Security soap:mustUnderstand="1"
411 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
412 wssecurity-secext-1.0.xsd">
413 (M005) <wsse:UsernameToken>
414 (M006) <wsse:Username>Chris</wsse:Username>
415 (M007) </wsse:UsernameToken>
416 (M008) </wsse:Security>
417 (M009) </soap:Header>
418 (M010) <soap:Body>

```

```

419 (M011) <Ping xmlns="http://xmlsoap.org/Ping">
420 (M012) <text>EchoString</text>
421 (M013) </Ping>
422 (M014) </soap:Body>
423 (M015) </soap:Envelope>

```

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

426

427 2.1.1.3 UsernameToken with timestamp, nonce and password hash

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

433

```

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

```

446

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

448

```

449 (M001) <?xml version="1.0" encoding="utf-8" ?>
450 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
451 (M003) <soap:Header>
452 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
453 (M005) <wsse:UsernameToken
454 wsu:Id="uuid-7cee5976-0111-e9c1-e34b-af1e85fa3866">
455 (M006) <wsse:Username>Chris</wsse:Username>
456 (M007) <wsse:Password Type="http://docs.oasis-
457 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
458 1.0#PasswordDigest"
459 >weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==</wsse:Password>
460 (M008) <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
461 (M009) <wsu:Created>2007-05-01T01:15:30Z</wsu:Created>
462 (M010) </wsse:UsernameToken>
463 (M011) </wsse:Security>
464 (M012) </soap:Header>
465 (M013) <soap:Body wsu:Id="uuid-7cee4264-0111-e0cb-8329-af1e85fa3866">
466 (M014) <Ping xmlns="http://xmlsoap.org/Ping">
467 (M015) <text>EchoString</text>
468 (M016) </Ping>
469 (M017) </soap:Body>
470 (M018) </soap:Envelope>

```

471

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

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

477 2.1.2 Use of SSL Transport Binding

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

484

```
485 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
486   <sp:TransportBinding>  
487     <wsp:Policy>  
488       <sp:TransportToken>  
489         <wsp:Policy>  
490           <sp:HttpsToken />  
491         </wsp:Policy>  
492       </sp:TransportToken>  
493       <sp:AlgorithmSuite>  
494         <wsp:Policy>  
495           <sp:Basic256 />  
496         </wsp:Policy>  
497       </sp:AlgorithmSuite>  
498       <sp:Layout>  
499         <wsp:Policy>  
500           <sp:Strict />  
501         </wsp:Policy>  
502       </sp:Layout>  
503       <sp:IncludeTimestamp />  
504     </wsp:Policy>  
505   </sp:TransportBinding>  
506 </wsp:Policy>
```

507 2.1.2.1 UsernameToken as supporting token

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

511

```
512 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
513 (P002)   <sp:TransportBinding>  
514 (P003)     <wsp:Policy>  
515 (P004)       <sp:TransportToken>  
516 (P005)         <wsp:Policy>  
517 (P006)           <sp:HttpsToken/>  
518 (P007)         </wsp:Policy>  
519 (P008)       </sp:TransportToken>  
520 (P009)     <sp:AlgorithmSuite>  
521 (P010)       <wsp:Policy>  
522 (P011)         <sp:Basic256/>  
523 (P012)       </wsp:Policy>  
524 (P013)     </sp:AlgorithmSuite>  
525 (P014)     <sp:Layout>  
526 (P015)       <wsp:Policy>  
527 (P016)         <sp:Strict/>  
528 (P017)       </wsp:Policy>  
529 (P018)     </sp:Layout>  
530 (P019)     <sp:IncludeTimestamp/>
```



```

531      (P020)      </wsp:Policy>
532      (P021)      </sp:TransportBinding>
533      (P022)      <sp:SupportingTokens>
534      (P023)      <wsp:Policy>
535      (P024)      <sp:UsernameToken/>
536      (P025)      </wsp:Policy>
537      (P026)      </sp:SupportingTokens>
538      (P027) </wsp:Policy>

```

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

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

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

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

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

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

```

553 (M001) <?xml version="1.0" encoding="utf-8" ?>
554 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
555 (M003)   <soap:Header>
556 (M004)     <wsse:Security soap:mustUnderstand="1"
557             xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
558             wss-wssecurity-secext-1.0.xsd">
559 (M005)       <wsu:Timestamp wsu:Id="uuid-8066364f-0111-f371-47bf-ba986d2d7dc4">
560 (M006)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
561 (M007)       </wsu:Timestamp>
562 (M008)     <wsse:UsernameToken
563             wsu:Id="uuid-8066368d-0111-e744-f37b-ba986d2d7dc4">
564 (M009)       <wsse:Username>Chris</wsse:Username>
565 (M010)     <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
566 200401-wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
567 (M011)     </wsse:UsernameToken>
568 (M012)   </wsse:Security>
569 (M013) </soap:Header>
570 (M014) <soap:Body wsu:Id="uuid-8066363f-0111-ffdc-de48-ba986d2d7dc4">
571 (M015)   <Ping xmlns="http://xmlsoap.org/Ping">
572 (M016)     <text>EchoString</text>
573 (M017)   </Ping>
574 (M018) </soap:Body>
575 (M019) </soap:Envelope>

```

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

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

579

580

581 2.1.3 (WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, 582 Encrypt

583

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

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

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

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

597 The policy is as follows:

```
598 (P001) <wsp:Policy wsu:Id="wss10_up_cert_policy" >  
599 (P002)   <wsp:ExactlyOne>  
600 (P003)     <wsp>All>  
601 (P004)       <sp:AsymmetricBinding>  
602 (P005)         <wsp:Policy>  
603 (P006)           <sp:InitiatorToken>  
604 (P007)             <wsp:Policy>  
605 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-  
606 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
607 (P009)                 <wsp:Policy>  
608 (P010)                   <sp:WssX509V3Token10/>  
609 (P011)                 </wsp:Policy>  
610 (P012)               </sp:X509Token>  
611 (P013)             </wsp:Policy>  
612 (P014)           </sp:InitiatorToken>  
613 (P015)         <sp:RecipientToken>  
614 (P016)           <wsp:Policy>  
615 (P017)             <sp:X509Token sp:IncludeToken="http://docs.oasis-  
616 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
617 (P018)                 <wsp:Policy>  
618 (P019)                   <sp:WssX509V3Token10/>  
619 (P020)                 </wsp:Policy>  
620 (P021)               </sp:X509Token>  
621 (P022)             </wsp:Policy>  
622 (P023)           </sp:RecipientToken>  
623 (P024)         <sp:AlgorithmSuite>  
624 (P025)           <wsp:Policy>  
625 (P026)             <sp:Basic256/>  
626 (P027)           </wsp:Policy>  
627 (P028)         </sp:AlgorithmSuite>  
628 (P029)       <sp:Layout>  
629 (P030)         <wsp:Policy>  
630 (P031)           <sp:Strict/>  
631 (P032)         </wsp:Policy>  
632 (P033)       </sp:Layout>  
633 (P034)     <sp:IncludeTimestamp/>  
634 (P035)     <sp:OnlySignEntireHeadersAndBody/>  
635 (P036)   </wsp:Policy>  
636 (P037) </sp:AsymmetricBinding>  
637 (P038) <sp:Wss10>  
638 (P039) <wsp:Policy>
```

```

639      (P040)      <sp:MustSupportRefKeyIdentifier/>
640      (P041)      </wsp:Policy>
641      (P042)      </sp:Wss10>
642      (P043)      <sp:SignedEncryptedSupportingTokens>
643      (P044)      <wsp:Policy>
644      (P045)      <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
645      open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
646      (P046)      <wsp:Policy>
647      (P047)      <sp:WssUsernameToken10/>
648      (P048)      </wsp:Policy>
649      (P049)      </sp:UsernameToken>
650      (P050)      </wsp:Policy>
651      (P051)      </sp:SignedEncryptedSupportingTokens>
652      (P052)      </wsp:All>
653      (P053)      </wsp:ExactlyOne>
654      </wsp:Policy>
655
656      (P054) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
657      (P055) <wsp:ExactlyOne>
658      (P056) <wsp:All>
659      (P057) <sp:SignedParts>
660      (P058) <sp:Body/>
661      (P059) </sp:SignedParts>
662      (P060) <sp:EncryptedParts>
663      (P061) <sp:Body/>
664      (P062) </sp:EncryptedParts>
665      (P063) </wsp:All>
666      (P064) </wsp:ExactlyOne>
667      (P065) </wsp:Policy>
668
669      (P066) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
670      (P067) <wsp:ExactlyOne>
671      (P068) <wsp:All>
672      (P069) <sp:SignedParts>
673      (P070) <sp:Body/>
674      (P071) </sp:SignedParts>
675      (P072) <sp:EncryptedParts>
676      (P073) <sp:Body/>
677      (P074) </sp:EncryptedParts>
678      (P075) </wsp:All>
679      (P076) </wsp:ExactlyOne>
680      (P077) </wsp:Policy>

```

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

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

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

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

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

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

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

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

```
705 (M001) <?xml version="1.0" encoding="utf-8" ?>
706 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
707 (M003)   <soap:Header>
708 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
709 (M005)       <xenc:ReferenceList>
710 (M006)         <xenc:DataReference URI="#encUT"/>
711 (M007)         <xenc:DataReference URI="#encBody"/>
712 (M008)       </xenc:ReferenceList>
713 (M009)       <wsu:Timestamp wsu:Id="T0">
714 (M010)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
715 (M011)       </wsu:Timestamp>
716 (M012)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="...#X509v3"
717 EncodingType="...#Base64Binary">
718 (M013)         MIIIEZzCCA9CgAwIBAgIQEmtJZc0...
719 (M014)       </wsse:BinarySecurityToken>
720 (M015)       <xenc:EncryptedData wsu:Id="encUT">
721 (M016)         <ds:KeyInfo>
722 (M017)           <wsse:SecurityTokenReference>
723 (M018)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
724 ValueType="...#X509SubjectKeyIdentifier">
725 (M019)               MIGfMa0GCSq...
726 (M020)             </wsse:KeyIdentifier>
727 (M021)           </wsse:SecurityTokenReference>
728 (M022)         </ds:KeyInfo>
729 (M023)         <xenc:CipherData>
730 (M024)           <xenc:CipherValue>...</xenc:CipherValue>
731 (M025)         </xenc:CipherData>
732 (M026)       </xenc:EncryptedData>
733 (M027)       <ds:Signature>
734 (M028)         <ds:SignedInfo>...
735 (M029)           <ds:Reference URI="#T0">...</ds:Reference>
736 (M030)           <ds:Reference URI="#usernameToken">...</ds:Reference>
737 (M031)           <ds:Reference URI="#body">...</ds:Reference>
738 (M032)         </ds:SignedInfo>
739 (M033)       <ds:SignatureValue>HFLP...</ds:SignatureValue>
740 (M034)       <ds:KeyInfo>
741 (M035)         <wsse:SecurityTokenReference>
742 (M036)           <wsse:Reference URI="#binaryToken"/>
743 (M037)         </wsse:SecurityTokenReference>
744 (M038)       </ds:KeyInfo>
745 (M039)     </ds:Signature>
746 (M040)   </wsse:Security>
747 (M041) </soap:Header>
748 (M042) <soap:Body wsu:Id="body">
749 (M043)   <xenc:EncryptedData wsu:Id="encBody">
750 (M044)     <ds:KeyInfo>
751 (M045)       <wsse:SecurityTokenReference>
752 (M046)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
753 ValueType="...#X509SubjectKeyIdentifier">
754 (M047)           MIGfMa0GCSq...
755 (M048)         </wsse:KeyIdentifier>
756 (M049)       </wsse:SecurityTokenReference>
757 (M050)     </ds:KeyInfo>
758 (M051)     <xenc:CipherData>
759 (M052)       <xenc:CipherValue>...</xenc:CipherValue>
```

```
760 (M053) </xenc:CipherData>
761 (M054) </xenc:EncryptedData>
762 (M055) </soap:Body>
763 (M056) </soap:Envelope>
```

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

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

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

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

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

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

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

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

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

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

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

794

795 **2.1.3.1 (WSS 1.0) Encrypted UsernameToken with X.509v3**

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

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

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

803 This setup is only recommended when the sender cannot provide the “message signature” and it is
804 RECOMMENDED that the receiver employs some security mechanisms external to the message to
805 prevent the spoofing attacks.

806 The policy is as follows:

```
807 (P001) <wsp:Policy wsu:Id="wss10_encrypted_unt_policy" >
808 (P002) <sp:AsymmetricBinding>
809 (P003) <wsp:Policy>
810 (P004) <sp:InitiatorEncryptionToken>
```

```

811 (P005) <wsp:Policy>
812 (P006) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
813 sx/ws-securitypolicy/200702/IncludeToken/Never">
814 (P007) <wsp:Policy>
815 (P008) <sp:WssX509V3Token10/>
816 (P009) </wsp:Policy>
817 (P010) </sp:X509Token>
818 (P011) </wsp:Policy>
819 (P012) </sp:InitiatorEncryptionToken>
820 (P013) <sp:RecipientSignatureToken>
821 (P014) <wsp:Policy>
822 (P015) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
823 sx/ws-securitypolicy/200702/IncludeToken/Never">
824 (P016) <wsp:Policy>
825 (P017) <sp:WssX509V3Token10/>
826 (P018) </wsp:Policy>
827 (P019) </sp:X509Token>
828 (P020) </wsp:Policy>
829 (P021) </sp:RecipientSignatureToken>
830 (P022) <sp:AlgorithmSuite>
831 (P023) <wsp:Policy>
832 (P024) <sp:Basic256/>
833 (P025) </wsp:Policy>
834 (P026) </sp:AlgorithmSuite>
835 (P027) <sp:Layout>
836 (P028) <wsp:Policy>
837 (P029) <sp:Lax/>
838 (P030) </wsp:Policy>
839 (P031) </sp:Layout>
840 (P032) <sp:IncludeTimestamp/>
841 (P033) <sp:OnlySignEntireHeadersAndBody/>
842 (P034) </wsp:Policy>
843 (P035) </sp:AsymmetricBinding>
844 (P036) <sp:Wss10>
845 (P037) <wsp:Policy>
846 (P038) <sp:MustSupportRefKeyIdentifier/>
847 (P039) <sp:MustSupportRefIssuerSerial/>
848 (P040) </wsp:Policy>
849 (P041) </sp:Wss10>
850 (P042) <sp:EncryptedSupportingTokens>
851 (P043) <wsp:Policy>
852 (P044) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-
853 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
854 (P045) <wsp:Policy>
855 (P046) <sp:HashPassword/>
856 (P047) <sp:WssUsernameToken10/>
857 (P048) </wsp:Policy>
858 (P049) </sp:UsernameToken>
859 (P050) </wsp:Policy>
860 (P051) </sp:EncryptedSupportingTokens>
861 (P052) </wsp:Policy>
862
863 (P053) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
864 (P054) <wsp:ExactlyOne>
865 (P055) <wsp:All>
866 (P056) <sp:EncryptedParts>
867 (P057) <sp:Body/>
868 (P058) </sp:EncryptedParts>
869 (P059) </wsp:All>
870 (P060) </wsp:ExactlyOne>
871 (P061) </wsp:Policy>
872
873 (P062) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
874 (P063) <wsp:ExactlyOne>

```

```

875 (P064) <wsp:All>
876 (P065) <sp:SignedParts>
877 (P066) <sp:Body/>
878 (P067) </sp:SignedParts>
879 (P068) </wsp:All>
880 (P069) </wsp:ExactlyOne>
881 (P070) </wsp:Policy>

```

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

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

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

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

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

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

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

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

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

907

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

```

909 (M001) <?xml version="1.0" encoding="utf-8" ?>
910 (M002) <soapenv:Envelope xmlns:soapenv="..." xmlns:xenc="..." . . . >
911 (M003) <soapenv:Header>
912 (M004) <wsse:Security xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..."
913 soapenv:mustUnderstand="1" >
914 (M005) <xenc:EncryptedKey>
915 (M006) <xenc:EncryptionMethod Algorithm=". . .#rsa-oaep-mgf1p">
916 (M007) <ds:DigestMethod Algorithm=". . .#sha1" />
917 (M008) </xenc:EncryptionMethod>
918 (M009) <ds:KeyInfo>
919 (M010) <wsse:SecurityTokenReference >
920 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
921 (M012) Value="...#X509SubjectKeyIdentifier">CuJ. .
922 . =</wsse:KeyIdentifier>
923 (M013) </wsse:SecurityTokenReference>
924 (M014) </ds:KeyInfo>
925 (M015) <xenc:CipherData>
926 (M016) <xenc:CipherValue>dbj...=</xenc:CipherValue>
927 (M017) </xenc:CipherData>
928 (M018) <xenc:ReferenceList>
929 (M019) <xenc:DataReference URI="#encBody" />
930 (M020) <xenc:DataReference URI="#encUnt" />

```

```

931 (M021) </xenc:ReferenceList>
932 (M022) </xenc:EncryptedKey>
933 (M023) <xenc:EncryptedData Id="encUnt" Type="...#Element"
934 MimeType="text/xml" Encoding="UTF-8" >
935 (M024) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
936 (M025) <xenc:CipherData>
937 (M026) <xenc:CipherValue>Kzf...=</xenc:CipherValue>
938 (M027) </xenc:CipherData>
939 (M028) </xenc:EncryptedData>
940 (M029) <wsu:Timestamp >
941 (M030) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
942 (M031) </wsu:Timestamp>
943 (M032) </wsse:Security>
944 (M033) </soapenv:Header>
945 (M034) <soapenv:Body >
946 (M035) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
947 Encoding="UTF-8" >
948 (M036) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
949 (M037) <xenc:CipherData>
950 (M038) <xenc:CipherValue>a/9...B</xenc:CipherValue>
951 (M039) </xenc:CipherData>
952 (M040) </xenc:EncryptedData>
953 (M041) </soapenv:Body>
954 (M042) </soapenv:Envelope>

```

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

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

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

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

```

968 (M043) <wsse:UsernameToken wsu:Id="usernameToken">
969 (M044) <wsse:Username>Chris</wsse:Username>
970 (M045) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
971 wss-username-token-profile-1.0#PasswordDigest">oY...=</wsse:Password>
972 (M046) <wsse:Nonce EncodingType="...#Base64Binary">pN...=</wsse:Nonce>
973 (M047) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
974 (M048) </wsse:UsernameToken>

```

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

978

979 **2.1.4 (WSS 1.1), User Name with Certificates, Sign, Encrypt**

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

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

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

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

989 The policy is as follows:

```
990      (P001)    <wsp:Policy wsu:Id="WSS11UsernameWithCertificates_policy">
991      (P002)      <wsp:ExactlyOne>
992      (P003)        <wsp>All>
993      (P004)          <sp:SymmetricBinding>
994      (P005)            <wsp:Policy>
995      (P006)              <sp:ProtectionToken>
996      (P007)                <wsp:Policy>
997      (P008)                  <sp:X509Token sp:IncludeToken="http://docs.oasis-
998      open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
999      (P009)                    <wsp:Policy>
1000     (P010)                      <sp:RequireThumbprintReference/>
1001     (P011)                      <sp:WssX509V3Token11/>
1002     (P012)                    </wsp:Policy>
1003     (P013)                  </sp:X509Token>
1004     (P014)                </wsp:Policy>
1005     (P015)              </sp:ProtectionToken>
1006     (P016)            <sp:AlgorithmSuite>
1007     (P017)              <wsp:Policy>
1008     (P018)                <sp:Basic256/>
1009     (P019)              </wsp:Policy>
1010     (P020)            </sp:AlgorithmSuite>
1011     (P021)          <sp:Layout>
1012     (P022)            <wsp:Policy>
1013     (P023)              <sp:Strict/>
1014     (P024)            </wsp:Policy>
1015     (P025)          </sp:Layout>
1016     (P026)        <sp:IncludeTimestamp/>
1017     (P027)      <sp:OnlySignEntireHeadersAndBody/>
1018     (P028)    </wsp:Policy>
1019     (P029)  </sp:SymmetricBinding>
1020     (P030) <sp:SignedEncryptedSupportingTokens>
1021     (P031) <wsp:Policy>
1022     (P032) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
1023     open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1024     (P033) <wsp:Policy>
1025     (P034) <sp:WssUsernameToken11/>
1026     (P035) </wsp:Policy>
1027     (P036) </sp:UsernameToken>
1028     (P037) </wsp:Policy>
1029     (P038) </sp:SignedEncryptedSupportingTokens>
1030     (P039) <sp:Wss11>
1031     (P040) <wsp:Policy>
1032     (P041) <sp:MustSupportRefKeyIdentifier/>
1033     (P042) <sp:MustSupportRefIssuerSerial/>
1034     (P043) <sp:MustSupportRefThumbprint/>
1035     (P044) <sp:MustSupportRefEncryptedKey/>
1036     (P045) </wsp:Policy>
1037     (P046) </sp:Wss11>
1038     (P047) </wsp>All>
1039     (P048) </wsp:ExactlyOne>
1040     (P049) </wsp:Policy>
1041
1042     (P050) <wsp:Policy wsu:Id="UsernameForCertificates_input_policy">
1043     (P051) <wsp:ExactlyOne>
1044     (P052) <wsp>All>
1045     (P053) <sp:SignedParts>
1046     (P054) <sp:Body/>
1047     (P055) </sp:SignedParts>
```

```

1048 (P056) <sp:EncryptedParts>
1049 (P057) <sp:Body/>
1050 (P058) </sp:EncryptedParts>
1051 (P059) </wsp:All>
1052 (P060) </wsp:ExactlyOne>
1053 (P061) </wsp:Policy>
1054
1055 (P062) <wsp:Policy wsu:Id="UsernameForCertificate_output_policy">
1056 (P063) <wsp:ExactlyOne>
1057 (P064) <wsp:All>
1058 (P065) <sp:SignedParts>
1059 (P066) <sp:Body/>
1060 (P067) </sp:SignedParts>
1061 (P068) <sp:EncryptedParts>
1062 (P069) <sp:Body/>
1063 (P070) </sp:EncryptedParts>
1064 (P071) </wsp:All>
1065 (P072) </wsp:ExactlyOne>
1066 (P073) </wsp:Policy>

```

1067 Lines (P004) – (P297) contain a SymmetricBinding assertion which indicates the use of one token to both
1068 sign and encrypt a message.

1069 Lines (P006) – (P015) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P012)
1070 indicate that the protection token must be an X.509 token that must never be included in any messages in
1071 the message exchange. Line (P010) dictates the X.509 token must be an X.509v3 security token as
1072 described in the WS-Security 1.1 X.509 Token Profile. Line (P011) dicates a thumbprint reference must
1073 be used to identify the token in any message.

1074 Line (P026) requires the inclusion of a timestamp.

1075 Lines (P030) – (P038) contain a SignedEncryptedSupportingTokens assertion which identifies the
1076 inclusion of an additional token which must be included in the message signature and encrypted. Lines
1077 (P032) – (P036) indicate that the supporting token must be a UsernameToken and must be included in all
1078 messages to the recipient. Line (P034) dictates the UsernameToken must conform to the WS-Security 1.1
1079 UsernameToken Profile.

1080 Lines (P040) – (P046) contain some WS-Security 1.1 related interoperability requirements, specifically
1081 support for key identifier, issuer serial number, thumbprint, and encrypted key references.

1082 Lines (P050) – (P061) contain a policy that is attached to the input message. Lines (P053) – (P055)
1083 require that the body of the input message must be signed. Lines (P056) – (P058) require the body of the
1084 input message must be encrypted.

1085 Lines (P062) – (P073) contain a policy that is attached to the output message. Lines (P065) – (P067)
1086 require that the body of the output message must be signed. Lines (P068) – (P070) require the body of
1087 the output message must be encrypted.

1088 An example of an input message that conforms to the above stated policy is as follows.

```

1089 (M001) <?xml version="1.0" encoding="utf-8" ?>
1090 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1091 (M003) <soap:Header>
1092 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..."
1093 xmlns:wsu="...">
1094 (M005) <xenc:EncryptedKey wsu:Id="EK">
1095 (M006) <xenc:EncryptionMethod
1096 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
1097 (M007) <ds:KeyInfo>
1098 (M008) <wsse:SecurityTokenReference>
1099 (M009) <wsse:KeyIdentifier ValueType="http://docs.oasis-
1100 open.org/wss/oasis-wss-soap-message-security-1.1#ThumbPrintSHA1">
1101 (M010) LKiQ/CmFrJDJqCLFcjIhIsmZ/+0=
1102 (M011) </wsse:KeyIdentifier>
1103 (M012) </wsse:SecurityTokenReference>
1104 (M013) </ds:KeyInfo>
1105 (M014) </xenc:EncryptedKey>

```

```

1106 (M015) <xenc:ReferenceList>
1107 (M016) <xenc:DataReference URI="#encUT" />
1108 (M017) <xenc:DataReference URI="#encBody" />
1109 (M018) </xenc:ReferenceList>
1110 (M019) <wsu:Timestamp wsu:Id="T0">
1111 (M020) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1112 (M021) </wsu:Timestamp>
1113 (M022) <xenc:EncryptedData wsu:Id="encUT">
1114 (M023) <xenc:EncryptionMethod
1115 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1116 (M024) <ds:KeyInfo>
1117 (M025) <wsse:SecurityTokenReference>
1118 (M026) <wsse:Reference URI="#EK" />
1119 (M027) </wsse:SecurityTokenReference>
1120 (M028) </ds:KeyInfo>
1121 (M029) <xenc:CipherData>
1122 (M030) <xenc:CipherValue>...</xenc:CipherValue>
1123 (M031) </xenc:CipherData>
1124 (M032) </xenc:EncryptedData>
1125 (M033) <ds:Signature>
1126 (M034) <ds:SignedInfo>...
1127 (M035) <ds:Reference URI="#T0">...</ds:Reference>
1128 (M036) <ds:Reference URI="#usernameToken">...</ds:Reference>
1129 (M037) <ds:Reference URI="#body">...</ds:Reference>
1130 (M038) </ds:SignedInfo>
1131 (M039) <ds:SignatureValue>HFLP...</ds:SignatureValue>
1132 (M040) <ds:KeyInfo>
1133 (M041) <wsse:SecurityTokenReference>
1134 (M042) <wsse:Reference URI="#EK" />
1135 (M043) </wsse:SecurityTokenReference>
1136 (M044) </ds:KeyInfo>
1137 (M045) </ds:Signature>
1138 (M046) </wsse:Security>
1139 (M047) </soap:Header>
1140 (M048) <soap:Body wsu:Id="body">
1141 (M049) <xenc:EncryptedData wsu:Id="encBody">
1142 (M050) <xenc:EncryptionMethod
1143 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1144 (M051) <ds:KeyInfo>
1145 (M052) <wsse:SecurityTokenReference>
1146 (M053) <wsse:Reference URI="#EK" />
1147 (M054) </wsse:SecurityTokenReference>
1148 (M055) </ds:KeyInfo>
1149 (M056) <xenc:CipherData>
1150 (M057) <xenc:CipherValue>...</xenc:CipherValue>
1151 (M058) </xenc:CipherData>
1152 (M059) </xenc:EncryptedData>
1153 (M060) </soap:Body>
1154 (M061) </soap:Envelope>

```

1155 Lines (M005) – (M014) contain the encrypted symmetric key as required by the use of the
1156 SymmetricBinding assertion starting on line (P004) with an X509Token ProtectionToken assertion. Line
1157 (M006) references the KwRsaOaep Asymmetric Key Wrap algorithm dictated by the Basic 256 Algorithm
1158 Suite assertion on line (P018). Lines (M009) – (M011) hold a KeyIdentifier of the protection token used to
1159 encrypt the symmetric key as required by the SymmetricBinding assertion. Because the ProtectionToken
1160 assertion disallowed the token from being inserted into the message and instead required a thumbprint
1161 reference, a thumbprint reference is included to identify the token.

1162 Line (M016) is an encryption data reference that references the encrypted supporting UsernameToken on
1163 lines (M022) – (M032). The encryption was required by the SignedEncryptedSupportingTokens assertion
1164 on line (P038). Line (M023) references the Aes256 Encryption algorithm dictated by the Basic 256
1165 Algorithm Suite assertion on line (P018). The encrypted symmetric key is used to encrypt the
1166 UsernameToken as referenced on line (M026).

1167 Line (M017) is an encryption data reference that references the encrypted body of the message on lines
1168 (M049) – (M059). The encryption was required by the EncryptedParts assertion of the input message
1169 policy. The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line
1170 (M053).

1171 Lines (M019) – (M021) contain a timestamp for the message as required by the IncludeTimestamp
1172 assertion.

1173 Lines (M033) – (M045) contain the message signature.

1174 Line (M035) indicates the message timestamp is included in the signature as required by the
1175 IncludeTimestamp assertion definition.

1176 Line (M036) indicates the supporting UsernameToken is included in the signature as required by the
1177 SignedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is
1178 included below to better illustrate the reference.

```
1179 (M062) <wsse:UsernameToken wsu:Id="usernameToken" >  
1180 (M063) <wsse:Username>Chris</wsse:Username>  
1181 (M064) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-  
1182 200401-wss-username-token-profile-1.0#PasswordText ">sirhC</wsse:Password>  
1183 (M065) </wsse:UsernameToken>
```

1184 Line (M037) indicates the message body is included in the signature as required by the SignedParts
1185 assertion of the input message policy.

1186 Line (M042) references the encrypted symmetric key for signing as dictated by the SymmetricBinding
1187 assertion.

1188 2.2 X.509 Token Authentication Scenario Assertions

1189 2.2.1 (WSS1.0) X.509 Certificates, Sign, Encrypt

1190 This use-case corresponds to the situation where both parties have X.509v3 certificates (and public-
1191 private key pairs). The requestor identifies itself to the service. The message exchange is integrity
1192 protected and encrypted.

1193 This modeled by use of an asymmetric security binding assertion.

1194 The message level policies in this and subsequent sections cover a different scope of the web service
1195 definition than the security binding level policy and so appear as separate policies and are attached at
1196 WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a
1197 set of coordinated policies one with endpoint subject and two with message subjects to achieve this use
1198 case.

1199 The policies are as follows:

```
1200 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >  
1201 (P002) <wsp:ExactlyOne>  
1202 (P003) <wsp:All>  
1203 (P004) <sp:AsymmetricBinding>  
1204 (P005) <wsp:Policy>  
1205 (P006) <sp:InitiatorToken>  
1206 (P007) <wsp:Policy>  
1207 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1208 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient" >  
1209 (P009) <wsp:Policy>  
1210 (P010) <sp:WssX509V3Token10/>  
1211 (P011) </wsp:Policy>  
1212 (P012) </sp:X509Token>  
1213 (P013) </wsp:Policy>  
1214 (P014) </sp:InitiatorToken>  
1215 (P015) <sp:RecipientToken>  
1216 (P016) <wsp:Policy>  
1217 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1218 sx/ws-securitypolicy/200702/IncludeToken/Never" >
```

```

1219 (P018) <wsp:Policy>
1220 (P019) <sp:WssX509V3Token10/>
1221 (P020) </wsp:Policy>
1222 (P021) </sp:X509Token>
1223 (P022) </wsp:Policy>
1224 (P023) </sp:RecipientToken>
1225 (P024) <sp:AlgorithmSuite>
1226 (P025) <wsp:Policy>
1227 (P026) <sp:Basic256/>
1228 (P027) </wsp:Policy>
1229 (P028) </sp:AlgorithmSuite>
1230 (P029) <sp:Layout>
1231 (P030) <wsp:Policy>
1232 (P031) <sp:Strict/>
1233 (P032) </wsp:Policy>
1234 (P033) </sp:Layout>
1235 (P034) <sp:IncludeTimestamp/>
1236 (P035) <sp:OnlySignEntireHeadersAndBody/>
1237 (P036) </wsp:Policy>
1238 (P037) </sp:AsymmetricBinding>
1239 (P038) <sp:Wss10>
1240 (P039) <wsp:Policy>
1241 (P040) <sp:MustSupportRefKeyIdentifier/>
1242 (P041) </wsp:Policy>
1243 (P042) </sp:Wss10>
1244 (P043) </wsp:All>
1245 (P044) </wsp:ExactlyOne>
1246 (P045) </wsp:Policy>
1247
1248 (P046) <wsp:Policy wsu:Id="WSS10Anonymous_with_Certificates_input_policy">
1249 (P047) <wsp:ExactlyOne>
1250 (P048) <wsp:All>
1251 (P049) <sp:SignedParts>
1252 (P050) <sp:Body/>
1253 (P051) </sp:SignedParts>
1254 (P052) <sp:EncryptedParts>
1255 (P053) <sp:Body/>
1256 (P054) </sp:EncryptedParts>
1257 (P055) </wsp:All>
1258 (P056) </wsp:ExactlyOne>
1259 (P057) </wsp:Policy>
1260
1261 (P058) <wsp:Policy wsu:Id="WSS10anonymous_with_certs_output_policy">
1262 (P059) <wsp:ExactlyOne>
1263 (P060) <wsp:All>
1264 (P061) <sp:SignedParts>
1265 (P062) <sp:Body/>
1266 (P063) </sp:SignedParts>
1267 (P064) <sp:EncryptedParts>
1268 (P065) <sp:Body/>
1269 (P066) </sp:EncryptedParts>
1270 (P067) </wsp:All>
1271 (P068) </wsp:ExactlyOne>
1272 (P069) </wsp:Policy>

```

1273 Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token
1274 must be used for the message signature and the recipient's token must be used for message encryption.

1275 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)
1276 indicate that the initiator token must be an X.509 token that must be included with all messages sent to
1277 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the
1278 WS-Security 1.0 X.509 Token Profile.

1279 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)
1280 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token

1281 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to
 1282 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the
 1283 token in any messages where the token is used.

1284 Line (P034) requires the inclusion of a timestamp.

1285 Lines (P046) – (P057) contain a policy that is attached to the input message. Lines (P049) – (P051)
 1286 require that the body of the input message must be signed. Lines (P052) – (P054) require the body of the
 1287 input message must be encrypted.

1288 Lines (P058) – (P069) contain a policy that is attached to the output message. Lines (P061) – (P063)
 1289 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of
 1290 the output message must be encrypted.

1291 An example of an input message that conforms to the above stated policy is as follows.

```

1292 (M001) <?xml version="1.0" encoding="utf-8" ?>
1293 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1294 (M003)   <soap:Header>
1295 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
1296 (M005)       <xenc:EncryptedKey >
1297 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1298 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1299 (M008)         </xenc:EncryptionMethod>
1300 (M009)         <ds:KeyInfo>
1301 (M010)           <wsse:SecurityTokenReference >
1302 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1303 (M012)             Value="...#X509SubjectKeyIdentifier">
1304 (M013)               MlGFMa0GCSq...
1305 (M014)             </wsse:KeyIdentifier>
1306 (M015)           </ds:KeyInfo>
1307 (M016)           <xenc:CipherData>
1308 (M017)             <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1309 (M018)           </xenc:CipherData>
1310 (M019)           <xenc:ReferenceList>
1311 (M020)             <xenc:DataReference URI="#encBody"/>
1312 (M021)           </xenc:ReferenceList>
1313 (M022)         </xenc:EncryptedKey>
1314 (M023)         <wsu:Timestamp wsu:Id="T0">
1315 (M024)           <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1316 (M025)         </wsu:Timestamp>
1317 (M026)         <wsse:BinarySecurityToken wsu:Id="binaryToken" Value="...#X509v3"
1318 (M027)         EncodingType="...#Base64Binary">
1319 (M028)           MIIEZzCCA9CgAwIBAgIQEmtJZc0...
1320 (M029)         </wsse:BinarySecurityToken>
1321 (M030)         <ds:Signature>
1322 (M031)           <ds:SignedInfo>...
1323 (M032)             <ds:Reference URI="#T0">...</ds:Reference>
1324 (M033)             <ds:Reference URI="#body">...</ds:Reference>
1325 (M034)           </ds:SignedInfo>
1326 (M035)           <ds:SignatureValue>HFLP...</ds:SignatureValue>
1327 (M036)         <ds:KeyInfo>
1328 (M037)           <wsse:SecurityTokenReference>
1329 (M038)             <wsse:Reference URI="#binaryToken"/>
1330 (M039)           </wsse:SecurityTokenReference>
1331 (M040)         </ds:KeyInfo>
1332 (M041)       </ds:Signature>
1333 (M042)     </wsse:Security>
1334 (M043)   </soap:Header>
1335 (M044)   <soap:Body wsu:Id="body">
1336 (M045)     <xenc:EncryptedData wsu:Id="encBody">
1337 (M046)       <xenc:CipherData>
1338 (M047)         <xenc:CipherValue>...</xenc:CipherValue>
1339 (M048)       </xenc:CipherData>
1340 (M049)     </xenc:EncryptedData>
1341 (M050)   </soap:Body>

```

1342 (M049) </soap:Envelope>

1343 Line (M019) is an encryption data reference that references the encrypted body of the message on lines
 1344 (M043) – (M047). The encryption was required by the EncryptedParts assertion of the input message
 1345 policy. Lines (M011) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as
 1346 required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the
 1347 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included
 1348 token.

1349 Lines (M022) – (M024) contain a timestamp for the message as required by the IncludeTimestamp
 1350 assertion.

1351 Lines (M025) – (M027) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as
 1352 required by the InitiatorToken assertion.

1353 Lines (M028) – (M039) contain the message signature.

1354 Line (M030) indicates the message timestamp is included in the signature as required by the
 1355 IncludeTimestamp assertion definition.

1356 Line (M031) indicates the message body is included in the signature as required by the SignedParts
 1357 assertion of the input message policy.

1358 Note that the initiator’s BinarySecurityToken is not included in the message signature as it was not
 1359 required by policy.

1360 Lines (M035) – (M037) references the initiator’s BinarySecurityToken included in the message for
 1361 identifying the key used for signing as dictated by the AsymmetricBinding assertion.

1362 2.2.2 (WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt

1363 This scenario is based on WSS Interop, Scenario 3, [Web Services Security: Interop 1](#), Draft 06, Editor,
 1364 Hal Lockhart, BEA Systems

1365 This use case corresponds to the situation where both parties have X.509v3 certificates (and public-
 1366 private key pairs). The requestor wishes to identify itself to the service using its X.509 credential (strong
 1367 authentication). The message exchange needs to be integrity protected and encrypted as well. The
 1368 difference from previous use case is that the X509 token inserted by the client is included in the message
 1369 signature (see <ProtectTokens />).

1370 The policy is as follows:

```

1371 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1372 (P002)   <wsp:ExactlyOne>
1373 (P003)     <wsp:All>
1374 (P004)       <sp:AsymmetricBinding>
1375 (P005)         <wsp:Policy>
1376 (P006)           <sp:InitiatorToken>
1377 (P007)             <wsp:Policy>
1378 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1379 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1380 (P009)                 <wsp:Policy>
1381 (P010)                   <sp:WssX509V3Token10/>
1382 (P011)                 </wsp:Policy>
1383 (P012)               </sp:X509Token>
1384 (P013)             </wsp:Policy>
1385 (P014)           </sp:InitiatorToken>
1386 (P015)         <sp:RecipientToken>
1387 (P016)           <wsp:Policy>
1388 (P017)             <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1389 sx/ws-securitypolicy/200702/IncludeToken/Never">
1390 (P018)               <wsp:Policy>
1391 (P019)                 <sp:WssX509V3Token10/>
1392 (P020)               </wsp:Policy>
1393 (P021)             </sp:X509Token>
1394 (P022)           </wsp:Policy>
1395 (P023)         </sp:RecipientToken>

```

```

1396 (P024) <sp:AlgorithmSuite>
1397 (P025) <wsp:Policy>
1398 (P026) <sp:Basic256/>
1399 (P027) </wsp:Policy>
1400 (P028) </sp:AlgorithmSuite>
1401 (P029) <sp:Layout>
1402 (P030) <wsp:Policy>
1403 (P031) <sp:Strict/>
1404 (P032) </wsp:Policy>
1405 (P033) </sp:Layout>
1406 (P034) <sp:IncludeTimestamp/>
1407 (P035) <sp:ProtectTokens />
1408 (P036) <sp:OnlySignEntireHeadersAndBody/>
1409 (P037) </wsp:Policy>
1410 (P038) </sp:AsymmetricBinding>
1411 (P039) <sp:Wss10>
1412 (P040) <wsp:Policy>
1413 (P041) <sp:MustSupportRefKeyIdentifier/>
1414 (P042) </wsp:Policy>
1415 (P043) </sp:Wss10>
1416 (P044) </wsp:All>
1417 (P045) </wsp:ExactlyOne>
1418 (P046) </wsp:Policy>
1419
1420 (P047) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1421 (P048) <wsp:ExactlyOne>
1422 (P049) <wsp:All>
1423 (P050) <sp:SignedParts>
1424 (P051) <sp:Body/>
1425 (P052) </sp:SignedParts>
1426 (P053) <sp:EncryptedParts>
1427 (P054) <sp:Body/>
1428 (P055) </sp:EncryptedParts>
1429 (P056) </wsp:All>
1430 (P057) </wsp:ExactlyOne>
1431 (P058) </wsp:Policy>
1432
1433 (P059) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1434 (P060) <wsp:ExactlyOne>
1435 (P061) <wsp:All>
1436 (P062) <sp:SignedParts>
1437 (P063) <sp:Body/>
1438 (P064) </sp:SignedParts>
1439 (P065) <sp:EncryptedParts>
1440 (P066) <sp:Body/>
1441 (P067) </sp:EncryptedParts>
1442 (P068) </wsp:All>
1443 (P069) </wsp:ExactlyOne>
1444 (P070) </wsp:Policy>

```

1445 Lines (P004) – (P038) contain the AsymmetricBinding assertion which indicates that the initiator's token
1446 must be used for the message signature and the recipient's token must be used for message encryption.

1447 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)
1448 indicate that the initiator token must be an X.509 token that must be included with all messages sent to
1449 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the
1450 WS-Security 1.0 X.509 Token Profile.

1451 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)
1452 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token
1453 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to
1454 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the
1455 token in any messages where the token is used.

1456 Line (P034) requires the inclusion of a timestamp.

1457 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the
 1458 token used to generate that signature.

1459 Lines (P047) – (P058) contain a policy that is attached to the input message. Lines (P050) – (P052)
 1460 require that the body of the input message must be signed. Lines (P053) – (P055) require the body of the
 1461 input message must be encrypted.

1462 Lines (P059) – (P070) contain a policy that is attached to the output message. Lines (P062) – (P064)
 1463 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of
 1464 the output message must be encrypted.

1465 An example of an input message that conforms to the above stated policy is as follows.

```

1466 (M001) <?xml version="1.0" encoding="utf-8" ?>
1467 (M002) <soapenv:Envelope xmlns:soapenv="..."
1468 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="..." xmlns:xenc="..." ...>
1469 (M003)   <soapenv:Header>
1470 (M004)     <wsse:Security xmlns:wsse="..." xmlns:ds="..."
1471 soapenv:mustUnderstand="1">
1472 (M005)       <xenc:EncryptedKey >
1473 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1474 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1475 (M008)         </xenc:EncryptionMethod>
1476 (M009)         <ds:KeyInfo>
1477 (M010)           <wsse:SecurityTokenReference >
1478 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1479 ValueType="...#X509SubjectKeyIdentifier">CuJd...=</wsse:KeyIdentifier>
1480 (M012)           </wsse:SecurityTokenReference>
1481 (M013)         </ds:KeyInfo>
1482 (M014)       <xenc:CipherData>
1483 (M015)         <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1484 (M016)       </xenc:CipherData>
1485 (M017)       <xenc:ReferenceList>
1486 (M018)         <xenc:DataReference URI="#encBody"/>
1487 (M019)       </xenc:ReferenceList>
1488 (M020)     </xenc:EncryptedKey>
1489 (M021)     <wsu:Timestamp wsu:Id="Timestamp" >
1490 (M022)       <wsu:Created>2007-03-26T16:53:39Z</wsu:Created>
1491 (M023)     </wsu:Timestamp>
1492 (M024)     <wsse:BinarySecurityToken wsu:Id="bst" ValueType="...#X509v3"
1493 EncodingType="...#Base64Binary">MIID...=</wsse:BinarySecurityToken>
1494 (M025)     <ds:Signature>
1495 (M026)       <ds:SignedInfo>
1496 (M027)         <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#"/>
1497 (M028)         <ds:SignatureMethod Algorithm="...#rsa-sha1"/>
1498 (M029)         <ds:Reference URI="#Timestamp">
1499 (M030)           <ds:Transforms>... </ds:Transforms>
1500 (M031)           <ds:DigestMethod Algorithm="...#sha1"/>
1501 (M032)           <ds:DigestValue>+g0I...=</ds:DigestValue>
1502 (M033)         </ds:Reference>
1503 (M034)         <ds:Reference URI="#Body">...</ds:Reference>
1504 (M035)         <ds:Reference URI="#bst">...</ds:Reference>
1505 (M036)       </ds:SignedInfo>
1506 (M037)       <ds:SignatureValue>RRT...=</ds:SignatureValue>
1507 (M038)     </ds:Signature>
1508 (M039)     <ds:KeyInfo>
1509 (M040)       <wsse:SecurityTokenReference >
1510 (M041)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1511 ValueType="...#X509SubjectKeyIdentifier">Xeg5...=</wsse:KeyIdentifier>
1512 (M042)       </wsse:SecurityTokenReference>
1513 (M043)     </ds:KeyInfo>
1514 (M044)   </ds:Signature>
1515 (M045) </wsse:Security>
1516 (M046) </soapenv:Header>
1517 (M047) <soapenv:Body wsu:Id="Body" >
1518 (M048)   <xenc:EncryptedData Id="encBody" Type="...#Content"
1519   MimeTypes="text/xml" Encoding="UTF-8" >

```

```

1519 (M049) <xenc:EncryptionMethod Algorithm="...#aes256-cbc" />
1520 (M050) <xenc:CipherData>
1521 (M051) <xenc:CipherValue>W84fn...l</xenc:CipherValue>
1522 (M052) </xenc:CipherData>
1523 (M053) </xenc:EncryptedData>
1524 (M054) </soapenv:Body>
1525 (M055) </soapenv:Envelope>

```

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

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

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

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

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

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

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

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

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

1548 **2.2.2.1 (WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption**

1549 This scenario is based on WSS Interop, Scenario 4, [Web Services Security: Interop 2](#).

1550 A common variation on the previous example is where X.509 Certificates are still used for authentication,
1551 but a mutually agreed upon symmetric key is used for encryption.

1552 In this use case the policy is the same except that the InitiatorToken and RecipientToken are now each
1553 just SignatureTokens.

1554 A second variation in this use case is that the mutually agreed upon symmetric encryption key is
1555 characterized as an “IssuedToken” with a mutually agreed upon URI used to identify the out of band (not
1556 included in the message) token, which is included in a SupportingTokens element.

1557 The policy is as follows:

```

1558 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1559 (P002) <wsp:ExactlyOne>
1560 (P003) <wsp:All>
1561 (P004) <sp:AsymmetricBinding>
1562 (P005) <wsp:Policy>
1563 (P006) <sp:InitiatorSignatureToken>
1564 (P007) <wsp:Policy>
1565 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1566 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1567 (P009) <wsp:Policy>
1568 (P010) <sp:WssX509V3Token10/>
1569 (P011) </wsp:Policy>
1570 (P012) </sp:X509Token>
1571 (P013) </wsp:Policy>

```

```

1572 (P014) </sp:InitiatorSignatureToken>
1573 (P015) <sp:RecipientSignatureToken>
1574 (P016) <wsp:Policy>
1575 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1576 sx/ws-securitypolicy/200702/IncludeToken/Never">
1577 (P018) <wsp:Policy>
1578 (P019) <sp:WssX509V3Token10/>
1579 (P020) </wsp:Policy>
1580 (P021) </sp:X509Token>
1581 (P022) </wsp:Policy>
1582 (P023) </sp:RecipientSignatureToken>
1583 (P024) <sp:AlgorithmSuite>
1584 (P025) <wsp:Policy>
1585 (P026) <sp:Basic256/>
1586 (P027) </wsp:Policy>
1587 (P028) </sp:AlgorithmSuite>
1588 (P029) <sp:Layout>
1589 (P030) <wsp:Policy>
1590 (P031) <sp:Strict/>
1591 (P032) </wsp:Policy>
1592 (P033) </sp:Layout>
1593 (P034) <sp:IncludeTimestamp/>
1594 (P035) <sp:ProtectTokens />
1595 (P036) <sp:OnlySignEntireHeadersAndBody/>
1596 (P037) </wsp:Policy>
1597 (P038) </sp:AsymmetricBinding>
1598 (P039) <sp:SupportingTokens>
1599 (P040) <wsp:Policy>
1600 (P041) <sp:IssuedToken>
1601 (P042) <sp:Issuer>SomeMutuallyAgreedURI</sp:Issuer>
1602 (P043) <sp:RequireExternalReference/>
1603 (P044) </sp:IssuedToken>
1604 (P045) <sp:EncryptedParts>
1605 (P046) <sp:Body/>
1606 (P047) </sp:EncryptedParts>
1607 (P048) </wsp:Policy>
1608 (P049) </sp:SupportingTokens>
1609 (P050) <sp:Wss10>
1610 (P051) <wsp:Policy>
1611 (P052) <sp:MustSupportRefKeyIdentifier/>
1612 (P053) </wsp:Policy>
1613 (P054) </sp:Wss10>
1614 (P055) </wsp:All>
1615 (P056) </wsp:ExactlyOne>
1616 (P057) </wsp:Policy>
1617
1618 (P058) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1619 (P059) <wsp:ExactlyOne>
1620 (P060) <wsp:All>
1621 (P061) <sp:SignedParts>
1622 (P062) <sp:Body/>
1623 (P063) </sp:SignedParts>
1624 (P064) <sp:EncryptedParts>
1625 (P065) <sp:Body/>
1626 (P066) </sp:EncryptedParts>
1627 (P067) </wsp:All>
1628 (P068) </wsp:ExactlyOne>
1629 (P069) </wsp:Policy>
1630
1631 (P070) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1632 (P071) <wsp:ExactlyOne>
1633 (P072) <wsp:All>
1634 (P073) <sp:SignedParts>
1635 (P074) <sp:Body/>

```

```

1636 (P075)      </sp:SignedParts>
1637 (P076)      <sp:EncryptedParts>
1638 (P077)      <sp:Body/>
1639 (P078)      </sp:EncryptedParts>
1640 (P079)      </wsp:All>
1641 (P080)      </wsp:ExactlyOne>
1642 (P081) </wsp:Policy>

```

1643 Lines (P004) – (P038) contain the AsymmetricBindingAssertion which indicates that the initiator’s token
1644 must be used for the message signature and the recipient’s token must be used for message encryption.

1645 Lines (P006) – (P014) contain the InitiatorSignatureToken assertion. Within that assertion lines (P008) –
1646 (P012) indicate that the initiator token must be an X.509 token that must be included with all messages
1647 sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as
1648 described in the WS-Security 1.0 X.509 Token Profile. By specifying that this is an
1649 InitiatorSignatureToken, it will only be used to sign the message and not used for encryption.

1650 Lines (P015) – (P023) contain the RecipientSignatureToken assertion. Within that assertion lines (P017)
1651 – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0
1652 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead,
1653 according to the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to
1654 identify the token in any messages where the token is used. By specifying that this is an
1655 RecipientSignatureToken, it will only be used to sign the response and not used for encryption.

1656 Line (P034) requires the inclusion of a timestamp.

1657 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the
1658 token or token reference associated with the generation of that signature.

1659 Lines (P039) – (P049) contain a SupportingTokens assertion that contains an IssuedToken (P041) –
1660 (P044), which contains an Issuer element (P042) that identifies an explicit URI
1661 (“SomeMutuallyAgreedURI”) that must be used to indicate what token will be used. (Since the content of
1662 the IssuedToken element is specific to the Issuer, any mechanism can be used to identify the key, and in
1663 this case the simplest method of identifying the issuer with the explicit key has been chosen only to
1664 illustrate one possible method, but not to recommend as opposed to any other method.) Line (P043),
1665 RequireExternalReference indicates that the IssuedToken requires a token that is referenced external to
1666 the message. Lines (P045) – (P047) indicate that the token is to be used for encrypting parts of the
1667 message, explicitly, line (P046), the Body of the message.

1668 Lines (P058) – (P069) contain a policy that is attached to the input message. Lines (P061) – (P063)
1669 require that the body of the input message must be signed. Lines (P064) – (P066) require the body of the
1670 input message must be encrypted.

1671 Lines (P070) – (P081) contain a policy that is attached to the output message. Lines (P073) – (P075)
1672 require that the body of the output message must be signed. Lines (P076) – (P078) require the body of
1673 the output message must be encrypted.

1674

1675 The following example message is derived from the WSS Interop2 document.

1676

```

1677 (M001)      <?xml version="1.0" encoding="utf-8" ?>
1678 (M002)      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1679 (M003)          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1680 (M004)          xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
1681 (M005)      <soap:Header>
1682 (M006)          <wsse:Security soap:mustUnderstand="1"
1683 (M007)              xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext" >
1684 (M008)              <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" >
1685 (M009)                  <xenc:DataReference URI="#enc" />
1686 (M010)          </xenc:ReferenceList>
1687 (M011)          <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1688 (M012)              wsu:Id="timestamp" >
1689 (M013)              <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1690 (M014)          </wsu:Timestamp>
1691 (M015)          <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1692 (M016)              EncodingType="wsse:Base64Binary" >

```

```

1693 (M017)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1694 (M018)      wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1695 (M019)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1696 (M020)      <SignedInfo>
1697 (M021)      <CanonicalizationMethod
1698 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1699 (M023)      <SignatureMethod
1700 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1701 (M025)      <Reference URI="#body">
1702 (M026)      <Transforms>
1703 (M027)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1704 (M028)      </Transforms>
1705 (M029)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1706 (M030)      <DigestValue>QTV...dw</DigestValue>
1707 (M031)      </Reference>
1708 (M032)      <Reference URI="#myCert">
1709 (M033)      <Transforms>
1710 (M034)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1711 (M035)      </Transforms>
1712 (M036)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1713 (M037)      <DigestValue>XYZ...ab</DigestValue>
1714 (M038)      </Reference>
1715 (M039)      <Reference URI="#timestamp">
1716 (M040)      <Transforms>
1717 (M041)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1718 (M042)      </Transforms>
1719 (M043)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1720 (M044)      <DigestValue>XYZ...ab</DigestValue>
1721 (M045)      </Reference>
1722 (M046)      </SignedInfo>
1723 (M047)      <SignatureValue>H+x0...gUw</SignatureValue>
1724 (M048)      <KeyInfo>
1725 (M049)      <wsse:SecurityTokenReference>
1726 (M050)      <wsse:Reference URI="#myCert" />
1727 (M051)      </wsse:SecurityTokenReference>
1728 (M052)      </KeyInfo>
1729 (M053)      </Signature>
1730 (M054)      </wsse:Security>
1731 (M055)      </soap:Header>
1732 (M056)      <soap:Body wsu:Id="body"
1733 (M057)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1734 (M058)      <xenc:EncryptedData Id="enc"
1735 (M059)      Type="http://www.w3.org/2001/04/xmlenc#Content"
1736 (M060)      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1737 (M061)      <xenc:EncryptionMethod
1738 (M062)      Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
1739 (M063)      <xenc:KeyInfo>
1740 (M064)      <xenc:KeyName>SomeMutuallyAgreedURI</xenc:KeyName>
1741 (M065)      </xenc:KeyInfo>
1742 (M066)      <xenc:CipherData>
1743 (M067)      <xenc:CipherValue>AYb...Y8</xenc:CipherValue>
1744 (M068)      </xenc:CipherData>
1745 (M069)      </xenc:EncryptedData>
1746 (M070)      </soap:Body>
1747 (M071)      </soap:Envelope>

```

1748 In the above example, the main point of attention is line (M064), where the KeyName of the
1749 EncryptedData element is "SomeMutuallyAgreedURI". As indicated above the specific techniques used to
1750 identify and apply this token are totally within agreed upon methods define by the mutual parties.

1751 Lines (M005) – (M055) contain the SOAP Header element, which contains the WS-Security header
1752 (M006) – (M054).

1753 The ReferenceList (M008) – (M010) contains an internal reference (M009), "enc", identifying the
1754 EncryptedData element Id, line (M058). This EncryptedData (M058) – (M069) inside the SOAP Body,
1755 (M056) – (M070), was required to be encrypted by the policy (P058) – (P069) as described above.

1756 The Timestamp (M011) – (M014) is required by the policy line (P034).

1757 The BinarySecurityToken (M015) – (M018) contains the actual certificate used to sign the request as
1758 required by the policy, (P008), IncludeToken/AlwaysToRecipient.

1759 The Signature (M019) – (M053) contains a SignedInfo (M020) – (M046) that identifies the “#body” (M025)
1760 the “#myCert” (M032), and the “#timestamp” (M039) as the elements covered by the signature. The
1761 Reference (M032) – (M038) with URI=“#myCert” that covers the BinarySecurityToken (M015) – (M018)
1762 was required by ProtectTokens in the policy (P035).

1763 The KeyInfo (M048) – (M052) uses a SecurityTokenReference to point to the “myCert”
1764 BinarySecurityToken.

1765 The EncryptedData (M058) – (M069) was described above, where it was pointed out that line (M064)
1766 contains the external reference (“SomeMutuallyAgreedURI”) to the mutually shared symmetric key used
1767 for encryption.

1768 2.2.3 (WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt

1769 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”
1770 [WS-SECURE-INTEROP] (see also sec 2.1.4)

1771 In this use case the Request is signed using DerivedKeyToken1(K), then encrypted using a
1772 DerivedKeyToken2(K) where K is ephemeral key protected for the server's certificate. Response is signed
1773 using DKT3(K), (if needed) encrypted using DKT4(K). The requestor does not wish to identify himself; the
1774 message exchange is protected using derived symmetric keys. As a simpler, but less secure, alternative,
1775 ephemeral key K (instead of derived keys) could be used for message protection by simply omitting the
1776 sp:RequireDerivedKeys assertion.

1777 The policy is as follows:

```
1778 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">  
1779 (P002)   <wsp:ExactlyOne>  
1780 (P003)     <wsp:All>  
1781 (P004)       <sp:SymmetricBinding>  
1782 (P005)         <wsp:Policy>  
1783 (P006)           <sp:ProtectionToken>  
1784 (P007)             <wsp:Policy>  
1785 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1786 sx/ws-securitypolicy/200702/IncludeToken/Never">  
1787 (P009)                 <wsp:Policy>  
1788 (P010)                   <sp:RequireDerivedKeys/>  
1789 (P011)                   <sp:RequireThumbprintReference/>  
1790 (P012)                   <sp:WssX509V3Token11/>  
1791 (P013)                 </wsp:Policy>  
1792 (P014)               </sp:X509Token>  
1793 (P015)             </wsp:Policy>  
1794 (P016)           </sp:ProtectionToken>  
1795 (P017)         <sp:AlgorithmSuite>  
1796 (P018)           <wsp:Policy>  
1797 (P019)             <sp:Basic256/>  
1798 (P020)           </wsp:Policy>  
1799 (P021)         </sp:AlgorithmSuite>  
1800 (P022)       <sp:Layout>  
1801 (P023)         <wsp:Policy>  
1802 (P024)           <sp:Strict/>  
1803 (P025)         </wsp:Policy>  
1804 (P026)       </sp:Layout>  
1805 (P027)     <sp:IncludeTimestamp/>  
1806 (P028)     <sp:OnlySignEntireHeadersAndBody/>  
1807 (P029)   </wsp:Policy>  
1808 (P030) </sp:SymmetricBinding>  
1809 (P031) <sp:Wss11>  
1810 (P032)   <wsp:Policy>  
1811 (P033)     <sp:MustSupportRefKeyIdentifier/>  
1812 (P034)     <sp:MustSupportRefIssuerSerial/>  
1813 (P035)     <sp:MustSupportRefThumbprint/>  
1814 (P036)     <sp:MustSupportRefEncryptedKey/>  
1815 (P037)     <sp:RequireSignatureConfirmation/>  
1816 (P038)   </wsp:Policy>
```

```

1817 (P039) </sp:Wss11>
1818 (P040) </wsp:All>
1819 (P041) </wsp:ExactlyOne>
1820 (P042) </wsp:Policy>
1821
1822 (P043) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_input_policy">
1823 (P044) <wsp:ExactlyOne>
1824 (P045) <wsp:All>
1825 (P046) <sp:SignedParts>
1826 (P047) <sp:Body/>
1827 (P048) </sp:SignedParts>
1828 (P049) <sp:EncryptedParts>
1829 (P050) <sp:Body/>
1830 (P051) </sp:EncryptedParts>
1831 (P052) </wsp:All>
1832 (P053) </wsp:ExactlyOne>
1833 (P054) </wsp:Policy>
1834
1835 (P055) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_output_policy">
1836 (P056) <wsp:ExactlyOne>
1837 (P057) <wsp:All>
1838 (P058) <sp:SignedParts>
1839 (P059) <sp:Body/>
1840 (P060) </sp:SignedParts>
1841 (P061) <sp:EncryptedParts>
1842 (P062) <sp:Body/>
1843 (P063) </sp:EncryptedParts>
1844 (P064) </wsp:All>
1845 (P065) </wsp:ExactlyOne>
1846 (P066) </wsp:Policy>

```

1847 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token
1848 must be used for both message signature and message encryption.

1849 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)
1850 indicate that the ProtectionToken must be an X.509 token that MUST NOT be included with any message
1851 sent between the Initiator and Recipient.

1852 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an
1853 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the
1854 MustSupportRefThumbprint assertion on line (P035) and RequireThumbprintReference on line (P011), a
1855 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token
1856 is used.

1857 Line (P027) requires the inclusion of a timestamp.

1858 Lines (P031) – (P039) contain some WS-Security 1.1 related interoperability requirements, specifically
1859 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires
1860 signature confirmation on the response.

1861 Lines (P043) – (P054) contain a policy that is attached to the input message. Lines (P045) – (P048)
1862 require that the body of the input message must be signed. Lines (P049) – (P051) require the body of the
1863 input message must be encrypted.

1864 Lines (P055) – (P066) contain a policy that is attached to the output message. Lines (P057) – (P060)
1865 require that the body of the output message must be signed. Lines (P061) – (P063) require the body of
1866 the output message must be encrypted.

1867 An example of an input message that conforms to the above stated policy is as follows.

1868 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,
1869 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the
1870 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-
1871 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```

1872 (M001) <?xml version="1.0" encoding="utf-8" ?>
1873 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">

```

```

1874 (M003) <env:Header>
1875 (M004) <wsse:Security xmlns:wsse="..." xmlns:wssell="..." xmlns:wsc="..."
1876 env:mustUnderstand="1">
1877 (M005) <xenc:EncryptedKey Id="encKey" >
1878 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
1879 (M007) <ds:DigestMethod Algorithm...#sha1" />
1880 (M008) </xenc:EncryptionMethod>
1881 (M009) <ds:KeyInfo >
1882 (M010) <wsse:SecurityTokenReference >
1883 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1884 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
1885 (M012) </wsse:SecurityTokenReference>
1886 (M013) </ds:KeyInfo>
1887 (M014) <xenc:CipherData>
1888 (M015) <xenc:CipherValue>TE...=</xenc:CipherValue>
1889 (M016) </xenc:CipherData>
1890 (M017) </xenc:EncryptedKey>
1891 (M018) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey1">
1892 (M019) wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1893 (M020) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey" />
1894 (M021) </wsse:SecurityTokenReference>
1895 (M022) <wsc:Generation>0</wsc:Generation>
1896 (M023) <wsc:Length>32</wsc:Length>
1897 (M024) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1898 (M025) <wsc:Nonce>39...=</wsc:Nonce>
1899 (M026) </wsc:DerivedKeyToken>
1900 (M027) <xenc:ReferenceList>
1901 (M028) <xenc:DataReference URI="#encBody" />
1902 (M029) </xenc:ReferenceList>
1903 (M030) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey2">
1904 (M031) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1905 (M032) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey" />
1906 (M033) </wsse:SecurityTokenReference>
1907 (M034) <wsc:Generation>0</wsc:Generation>
1908 (M035) <wsc:Length>32</wsc:Length>
1909 (M036) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1910 (M037) <wsc:Nonce>...=</wsc:Nonce>
1911 (M038) </wsc:DerivedKeyToken>
1912 (M039) <wsu:Timestamp wsu:Id="Timestamp" >
1913 (M040) <wsu:Created>2007-03-26T23:43:13Z</wsu:Created>
1914 (M041) </wsu:Timestamp>
1915 (M042) <ds:Signature >
1916 (M043) <ds:SignedInfo>
1917 (M044) ...
1918 (M045) <ds:Reference URI="#Timestamp">...</ds:Reference>
1919 (M046) <ds:Reference URI="#Body">...</ds:Reference>
1920 (M047) </ds:SignedInfo>
1921 (M048) <ds:SignatureValue>Yu...=</ds:SignatureValue>
1922 (M049) <ds:KeyInfo>
1923 (M050) <wsse:SecurityTokenReference >
1924 (M051) <wsse:Reference URI="#DKey2" ValueType=".../dk" />
1925 (M052) </wsse:SecurityTokenReference>
1926 (M053) </ds:KeyInfo>
1927 (M054) </ds:Signature>
1928 (M055) </wsse:Security>
1929 (M056) </env:Header>
1930 (M057) <env:Body wsu:Id="Body" >
1931 (M058) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
1932 Encoding="UTF-8" >
1933 (M059) <xenc:EncryptionMethod Algorithm="...#aes256-cbc" />
1934 (M060) <ds:KeyInfo >
1935 (M061) <wsse:SecurityTokenReference >
1936 (M062) <wsse:Reference URI="#DKey1" ValueType=".../dk" />
1937 (M063) </wsse:SecurityTokenReference>
1938 (M064) </ds:KeyInfo>
1939 (M065) <xenc:CipherData>
1940 (M066) <xenc:CipherValue>p53...f</xenc:CipherValue>
1941 (M067) </xenc:CipherData>
1942 (M068) </xenc:EncryptedData>
1943 (M069) </env:Body>

```


1944 (M070) </env:Envelope>

1945

1946 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1
1947 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.
1948 Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is
1949 used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value
1950 type of #ThumbprintSHA1 for Thumbprint Reference, it is required by the policy line (P011) of the
1951 Thumbprint Reference.

1952 Lines (M018) – (M022) is the first wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines
1953 (M005) – (M017), which is referenced using the WSS1.1 #EncryptedKey SecurityTokenReference
1954 mechanism contained in lines (M019) – (M021), and used for body encryption and it is referenced on line
1955 (M062).

1956 Lines (M027) – (M029) is an encryption data reference that references the encrypted body of the
1957 message on lines (M058) – (M068). The encryption was required by the EncryptedParts assertion of the
1958 input message policy.

1959 Lines (M030) – (M038) is the second wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines
1960 (M005) – (M017) and used for signature referenced on line (M051).

1961 Lines (M039) – (M041) contain a timestamp for the message as required by the IncludeTimestamp
1962 assertion.

1963 Lines (M042) – (M054) contain the message signature.

1964 Line (M045) indicates the message timestamp is included in the signature as required by the
1965 IncludeTimestamp assertion definition.

1966 Line (M046) indicates the message body is included in the signature as required by the SignedParts
1967 assertion of the input message policy.

1968 2.2.4 (WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt

1969 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”
1970 [\[WS-SECURE-INTEROP\]](#) (see also sec 2.1.4)

1971 Client and server X509 certificates are used for client and server authorization respectively. Request is
1972 signed using K, then encrypted using K, K is ephemeral key protected for server's certificate. Signature
1973 corresponding to K is signed using client certificate. Response is signed using K, encrypted using K,
1974 encrypted key K is not included in response. Alternatively, derived keys can be used for each of the
1975 encryption and signature operations by simply adding an sp:RequireDerivedKeys assertion.

1976 The policy is as follows:

1977

```
1978 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">  
1979 (P002)   <wsp:ExactlyOne>  
1980 (P003)     <wsp:All>  
1981 (P004)       <sp:SymmetricBinding>  
1982 (P005)         <wsp:Policy>  
1983 (P006)           <sp:ProtectionToken>  
1984 (P007)             <wsp:Policy>  
1985 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1986 sx/ws-securitypolicy/200702/IncludeToken/Never">  
1987 (P009)                 <wsp:Policy>  
1988 (P010)                   <sp:RequireDerivedKeys/>  
1989 (P011)                     <sp:RequireThumbprintReference/>  
1990 (P012)                       <sp:WssX509V3Token11/>  
1991 (P013)                         </wsp:Policy>  
1992 (P014)                           </sp:X509Token>  
1993 (P015)                             </wsp:Policy>  
1994 (P016)                               </sp:ProtectionToken>  
1995 (P017)                                 <sp:AlgorithmSuite>  
1996 (P018)                                   <wsp:Policy>
```

```

1997 (P019) <sp:Basic256/>
1998 (P020) </wsp:Policy>
1999 (P021) </sp:AlgorithmSuite>
2000 (P022) <sp:Layout>
2001 (P023) <wsp:Policy>
2002 (P024) <sp:Strict/>
2003 (P025) </wsp:Policy>
2004 (P026) </sp:Layout>
2005 (P027) <sp:IncludeTimestamp/>
2006 (P028) <sp:OnlySignEntireHeadersAndBody/>
2007 (P029) </wsp:Policy>
2008 (P030) </sp:SymmetricBinding>
2009 (P031) <sp:EndorsingSupportingTokens>
2010 (P032) <wsp:Policy>
2011 (P033) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2012 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2013 (P034) <wsp:Policy>
2014 (P035) <sp:RequireThumbprintReference/>
2015 (P036) <sp:WssX509V3Token11/>
2016 (P037) </wsp:Policy>
2017 (P038) </sp:X509Token>
2018 (P039) </wsp:Policy>
2019 (P040) </sp:EndorsingSupportingTokens>
2020 (P041) <sp:Wss11>
2021 (P042) <wsp:Policy>
2022 (P043) <sp:MustSupportRefKeyIdentifier/>
2023 (P044) <sp:MustSupportRefIssuerSerial/>
2024 (P045) <sp:MustSupportRefThumbprint/>
2025 (P046) <sp:MustSupportRefEncryptedKey/>
2026 (P047) <sp:RequireSignatureConfirmation/>
2027 (P048) </wsp:Policy>
2028 (P049) </sp:Wss11>
2029 (P050) </wsp:All>
2030 (P051) </wsp:ExactlyOne>
2031 (P052) </wsp:Policy>
2032
2033 (P053) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_input_policy">
2034 (P054) <wsp:ExactlyOne>
2035 (P055) <wsp:All>
2036 (P056) <sp:SignedParts>
2037 (P057) <sp:Body/>
2038 (P058) </sp:SignedParts>
2039 (P059) <sp:EncryptedParts>
2040 (P060) <sp:Body/>
2041 (P061) </sp:EncryptedParts>
2042 (P062) </wsp:All>
2043 (P063) </wsp:ExactlyOne>
2044 (P064) </wsp:Policy>
2045
2046 (P065) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_output_policy">
2047 (P066) <wsp:ExactlyOne>
2048 (P067) <wsp:All>
2049 (P068) <sp:SignedParts>
2050 (P069) <sp:Body/>
2051 (P070) </sp:SignedParts>
2052 (P071) <sp:EncryptedParts>
2053 (P072) <sp:Body/>
2054 (P073) </sp:EncryptedParts>
2055 (P074) </wsp:All>
2056 (P075) </wsp:ExactlyOne>
2057 (P076) </wsp:Policy>

```

2058 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token
2059 must be used for both message signature and message encryption.

2060 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)
2061 indicate that the initiator token must be an X.509 token that must not be included with all messages sent
2062 between the recipient and Recipient.

2063 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an
2064 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the
2065 MustSupportRefThumbprint assertion on line (P043) and RequireThumbprintReference on line (P011), a
2066 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token
2067 is used.

2068 Line (P027) requires the inclusion of a timestamp.

2069 Lines (P031) – (P040) contain the EndorsingSupportingTokens assertion which indicates that the message
2070 signature should be endorsed by client's X509 certificate on line (P033) – (P038). Line (P033)
2071 IncludeToken=".../AlwaysToRecipient" indicates the endorsing is only required when the message is sent to recipient.
2072 Line (P036) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security
2073 1.1 X.509 Token Profile. and RequireThumbprintReference on line (P035), a Thumbprint Reference of
2074 KeyIdentifier must be used to identify the token in any messages where the token is used.

2075 Lines (P041) – (P049) contain some WS-Security 1.1 related interoperability requirements, specifically
2076 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires
2077 signature confirmation on the response.

2078 Lines (P053) – (P064) contain a policy that is attached to the input message. Lines (P055) – (P058)
2079 require that the body of the input message must be signed. Lines (P059) – (P061) require the body of the
2080 input message must be encrypted.

2081 Lines (P065) – (P076) contain a policy that is attached to the output message. Lines (P067) – (P070)
2082 require that the body of the output message must be signed. Lines (P071) – (P073) require the body of
2083 the output message must be encrypted.

2084 An example of an input message that conforms to the above stated policy is as follows.

2085 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,
2086 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the
2087 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-
2088 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
2089 (M001) <?xml version="1.0" encoding="utf-8" ?>
2090 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
2091 (M003)   <env:Header>
2092 (M004)     <wsse:Security xmlns:wsse="..." xmlns:wssell="..." env:mustUnderstand="1">
2093 (M005)       <xenc:EncryptedKey Id="encKey" >
2094 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
2095 (M007)           <ds:DigestMethod Algorithm="...#sha1" />
2096 (M008)         </xenc:EncryptionMethod>
2097 (M009)         <ds:KeyInfo >
2098 (M010)           <wsse:SecurityTokenReference >
2099 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2100 (M012)               ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
2101 (M013)             </wsse:SecurityTokenReference>
2102 (M014)           </ds:KeyInfo>
2103 (M015)         <xenc:CipherData>
2104 (M016)           <xenc:CipherValue>eI...=</xenc:CipherValue>
2105 (M017)         </xenc:CipherData>
2106 (M018)       </xenc:EncryptedKey>
2107 (M019)       <wsse:BinarySecurityToken ValueType="...#X509v3"
2108 (M020)       EncodingType="...#Base64Binary">MI...=</wsse:BinarySecurityToken>
2109 (M020)       <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
2110 (M020)       wsu:Id="derivedKeyToken1">
2111 (M021)         <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey" >
2112 (M022)           <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
2113 (M023)         </wsse:SecurityTokenReference>
2114 (M024)       <wsc:Generation>0</wsc:Generation>
2115 (M025)       <wsc:Length>32</wsc:Length>
2116 (M026)       <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2117 (M027)       <wsc:Nonce>+R...=</wsc:Nonce>
2118 (M028)     </wsc:DerivedKeyToken>
```

```

2119 (M029) <wsu:Timestamp wsu:Id="Timestamp" >
2120 (M030) <wsu:Created>2007-03-31T04:27:21Z</wsu:Created>
2121 (M031) </wsu:Timestamp>
2122 (M032) <nl:ReferenceList xmlns:nl=".../xmlenc#">
2123 (M033) <nl:DataReference URI="#encBody"/>
2124 (M034) </nl:ReferenceList>
2125 (M035) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_shal"
2126 wsu:Id="derivedKeyToken2">
2127 (M036) <wsse:SecurityTokenReference wsse1:TokenType="...#EncryptedKey" >
2128 (M037) <wsse:Reference ValueType="...EncryptedKey" URI="#encKey"/>
2129 (M038) </wsse:SecurityTokenReference>
2130 (M039) <wsc:Generation>0</wsc:Generation>
2131 (M040) <wsc:Length>32</wsc:Length>
2132 (M041) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2133 (M042) <wsc:Nonce>wL...=</wsc:Nonce>
2134 (M043) </wsc:DerivedKeyToken>
2135 (M044) <ds:Signature Id="messageSignature">
2136 (M045) <ds:SignedInfo>
2137 (M046) ...
2138 (M047) <ds:Reference URI="#Timestamp">
2139 (M048) ...
2140 (M049) </ds:Reference>
2141 (M050) <ds:Reference URI="#Body">
2142 (M051) ...
2143 (M052) </ds:Reference>
2144 (M053) </ds:SignedInfo>
2145 (M054) <ds:SignatureValue>abcdefg</ds:SignatureValue>
2146 (M055) <ds:KeyInfo>
2147 (M056) <wsse:SecurityTokenReference >
2148 (M057) <wsse:Reference URI="#derivedKeyToken2" ValueType=".../dk"/>
2149 (M058) </wsse:SecurityTokenReference>
2150 (M059) </ds:KeyInfo>
2151 (M060) </ds:Signature>
2152 (M061) <ds:Signature >
2153 (M062) <ds:SignedInfo>
2154 (M063) ...
2155 (M064) <ds:Reference URI="#messageSignature">
2156 (M065) ...
2157 (M066) </ds:Reference>
2158 (M067) </ds:SignedInfo>
2159 (M068) <ds:SignatureValue>hijklmnop</ds:SignatureValue>
2160 (M069) <ds:KeyInfo>
2161 (M070) <wsse:SecurityTokenReference >
2162 (M071) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2163 ValueType="...#ThumbprintSHA1">Pj...=</wsse:KeyIdentifier>
2164 (M072) </wsse:SecurityTokenReference>
2165 (M073) </ds:KeyInfo>
2166 (M074) </ds:Signature>
2167 (M075) </wsse:Security>
2168 (M076) </env:Header>
2169 (M077) <env:Body wsu:Id="Body" >
2170 (M078) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
2171 Encoding="UTF-8" >
2172 (M079) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
2173 (M080) <ds:KeyInfo >
2174 (M081) <wsse:SecurityTokenReference >
2175 (M082) <wsse:Reference URI="#derivedKeyToken1" ValueType=".../dk"/>
2176 (M083) </wsse:SecurityTokenReference>
2177 (M084) </ds:KeyInfo>
2178 (M085) <xenc:CipherData>
2179 (M086) <xenc:CipherValue>70...=</xenc:CipherValue>
2180 (M087) </xenc:CipherData>
2181 (M088) </xenc:EncryptedData>
2182 (M089) </env:Body>
2183 (M090) </env:Envelope>
2184 (M091)

```

2185

2186 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1
2187 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.

2188 Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token
 2189 from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.
 2190 In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint
 2191 Reference, and it is required by the policy line (P011) of the Thumbprint Reference.

2192 Line (M019) is an X509 BinarySecurityToken that contains the actual X509 certificate that is used by the
 2193 endorsing signature described below. The token is required to be present in the message based on the
 2194 line (P033) that requires this token for the message sent to the recipient.

2195 Lines (M020) – (M027) is the first derived key token. It is derived from the EncryptedKey of lines (M005) –
 2196 (M017) and used for body encryption referenced on line (M081).

2197 Lines (M028) – (M030) contain a Timestamp element for the message as required by the
 2198 IncludeTimestamp assertion.

2199 Lines (M031) – (M033) contain an encryption data reference that references the encrypted body of the
 2200 message on lines (M077) – (M087). The encryption was required by the EncryptedParts assertion of the
 2201 input message policy.

2202 Lines (M034) – (M042) is the second derived key token. It is derived from the EncryptedKey of lines
 2203 (M005) – (M017) and used by the signature that references it on line (M056).

2204 Lines (M043) – (M059) contain the message signature. Lines (M054) – (M058) is its KeyInfo block that
 2205 indicates the second derived key token should be used to verify the message signature.

2206 Line (M057) indicates the message timestamp is included in the signature as required by the
 2207 IncludeTimestamp assertion definition.

2208 Line (M060) indicates the message body is included in the signature as required by the SignedParts
 2209 assertion of the input message policy.

2210 Lines (M060) – (M073) contain the endorsing signature. It signs the message signature, referenced on
 2211 line (M063), per EndorsingSupportingTokens assertion policy requirement on lines (P031) – (P040). Line
 2212 (M070), the KeyIdentifier, has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is
 2213 required by the policy line (P035) of the Thumbprint Reference. This Thumbprint Reference must match
 2214 the Thumbprint of the X509 Certificate contained in the BinarySecurityToken on line (M019), based on the
 2215 IncludeToken requirement line (P033).

2216

2217 An example of an output message that conforms to the above stated policy follows:

2218

```

2219 (R001) <env:Envelope xmlns:env="..." xmlns:wsu="...">
2220 (R002)   <env:Header>
2221 (R003)     <wsse:Security env:mustUnderstand="1" xmlns:wsse="...">
2222 (R004)       <wsu:Timestamp wsu:Id="Timestamp" >
2223 (R005)         <wsu:Created>2007-03-31T04:27:25Z</wsu:Created>
2224 (R006)       </wsu:Timestamp>
2225 (R007)     <wsc:DerivedKeyToken wsu:Id="derivedKeyToken1" xmlns:wsc=".../sc">
2226 (R008)       <wsse:SecurityTokenReference>
2227 (R009)         <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2228 (R010)           >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2229 (R011)       </wsse:SecurityTokenReference>
2230 (R012)     <wsc:Offset>0</wsc:Offset>
2231 (R013)     <wsc:Length>24</wsc:Length>
2232 (R014)     <wsc:Nonce>NfSNXZLYAA8mocQz19KWjg==</wsc:Nonce>
2233 (R015)     </wsc:DerivedKeyToken>
2234 (R016)     <wsc:DerivedKeyToken wsu:Id="derivedKeyToken2" xmlns:wsc=".../sc">
2235 (R017)       <wsse:SecurityTokenReference>
2236 (R018)         <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2237 (R019)           >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2238 (R020)       </wsse:SecurityTokenReference>
2239 (R021)     <wsc:Nonce>1F/CtyQ9d1Ro8E3+uZYmgQ==</wsc:Nonce>
2240 (R022)     </wsc:DerivedKeyToken>
2241 (R023)     <enc:ReferenceList xmlns:enc=".../xmlenc#">
2242 (R024)       <enc:DataReference URI="#encBody" />
2243 (R025)     </enc:ReferenceList>
2244 (R026)     <wsse11:SignatureConfirmation wsu:Id="sigconf1"
2245 (R027)       Value="abcdefg=" xmlns:wsse11="..." />

```

```

2246 (R028) <wssell:SignatureConfirmation wsu:Id="sigconf2"
2247 (R029) Value="hijklmnop=" xmlns:wssell="..." />
2248 (R030) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
2249 (R031) <SignedInfo>
2250 (R032) <CanonicalizationMethod
2251 (R033) Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2252 (R034) <SignatureMethod
2253 (R035) Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-shal" />
2254 (R036) <Reference URI="#msgBody">
2255 (R037) ...
2256 (R038) </Reference>
2257 (R039) <Reference URI="#Timestamp">
2258 (R040) ...
2259 (R041) </Reference>
2260 (R042) <Reference URI="#sigconf1">
2261 (R043) ...
2262 (R044) </Reference>
2263 (R045) <Reference URI="#sigconf2">
2264 (R046) ...
2265 (R047) </Reference>
2266 (R048) </SignedInfo>
2267 (R049) <SignatureValue>3rAxsfJ2LjF7liRQX2EH/0DBmzE=</SignatureValue>
2268 (R050) <KeyInfo>
2269 (R051) <wsse:SecurityTokenReference>
2270 (R052) <wsse:Reference URI="#derivedKeyToken1" />
2271 (R053) </wsse:SecurityTokenReference>
2272 (R054) </KeyInfo>
2273 (R055) </Signature>
2274 (R056) </wsse:Security>
2275 (R057) </env:Header>
2276 (R058) <env:Body wsu:Id="msgBody">
2277 (R059) <enc:EncryptedData Id="encBody" Type="...xmlenc#Content"
2278 (R060) xmlns:enc=".../xmlenc#">
2279 (R061) <enc:EncryptionMethod Algorithm=".../xmlenc#aes256-cbc" />
2280 (R062) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
2281 (R063) <wsse:SecurityTokenReference xmlns:wsse="...">
2282 (R064) <wsse:Reference URI="#derivedKeyToken2" />
2283 (R065) </wsse:SecurityTokenReference>
2284 (R066) </KeyInfo>
2285 (R067) <enc:CipherData>
2286 (R068) <enc:CipherValue>y+eV...pu</enc:CipherValue>
2287 (R069) </enc:CipherData>
2288 (R070) </enc:EncryptedData>
2289 (R071) </env:Body>
2290 (R072) </env:Envelope>
2291

```

2292 Lines (R001)-(R072) contain the Response message returned to the Initiator.

2293 Lines (R004)-(R006) contain the Timestamp required by the Policy (P027).

2294 Lines (R007)-(R015) and (R016)-(R022) contain the information necessary for the Initiator to derive the
2295 keys using the WS-SecureConversation shared secret, the identified key (R010) for DK1 and (R019) for
2296 DK2, which reference the same key, and the Nonce (R014) for DK1 and (R021) for DK2, which are
2297 different for the two derived keys.

2298 Lines (R023)-(R025) contain a ReferenceList that indicates the message Body (R058) contains the data
2299 to be encrypted. The encryption was required by the output policy (P079).

2300 Lines (R026)-(R027) contain the SignatureConfirmation for the SignatureValue in the request message
2301 (M054) which was required to be included by the policy (P047) RequireSignatureConfirmation.

2302 Lines (R028)-(R029) contain the SignatureConfirmation for the 2nd SignatureValue in the request
2303 message (M068), which is also required by the policy (P047).

2304 Lines (R030)-(R055) contain the response message signature that covers the Timestamp element
2305 (R039)->(R004) required to be signed by the policy (P027), the two SignatureConfirmation elements
2306 (R042)->(R026) and (R045)->(R028), which are required to be signed because they are required
2307 elements of the policy (P047), and the message Body (R036)->(R058), which is required to be signed by
2308 the output message policy (P076). The signature may be verified using derivedKeyToken1 as indicated
2309 on line (R052).

2310 Lines (R059)-(R070) contain the encrypted data as required by the policy (P079) and may be decrypted
2311 using derivedKeyToken2 as indicated on line (R064).

2312 2.3 SAML Token Authentication Scenario Assertions

2313 For SAML, the combination of SAML and WSS version numbers is supported (WssSamlV11Token10,
2314 WssSamlV11Token11, WssSamlV20Token11).

2315 Instead of explicitly including the SAML Assertion, a wsse:KeyIdentifier reference can also be used. To
2316 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)
2317 [securitypolicy/200702/IncludeToken/Never](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never).

2318 In all of the SAML scenarios, the SAML Assertion ConfirmationMethod expected to be used by the
2319 Initiator is communicated implicitly by the context of the sp: security binding in use and type of sp:
2320 element containing the SamlToken. There are 3 general SAML ConfirmationMethod use cases covered:
2321 holder-of-key (hk), sender-vouches (sv), and bearer (br).

2322

2323 For the **holder-of-key** case, there is always a contained reference (or value) in the SAML
2324 Assertion to key material that may be used for message protection in the scenario. The hk case
2325 can be assumed if the WssSamlV**Token** element appears either in the sp:InitiatorToken
2326 element of the sp:AsymmetricBinding element, or in the sp:ProtectionToken element of the
2327 sp:SymmetricBinding element. In the sp:TransportBinding case, if the WssSamlV**Token**
2328 element appears in an sp:EndorsingSupportingTokens or sp:SignedEndorsingSupportingTokens
2329 element, the SAML Assertion type can also be assumed to be hk.

2330 The **sender-vouches** case can be assumed if the WssSamlV**Token** version will always
2331 appear in an sp:SignedSupportingTokens element to indicate that the SAML Authority associated
2332 with this token is the Initiator that signs the message.

2333 The **bearer** case can be assumed if the WssSamlV**Token** version appears in an
2334 sp:SupportingTokens element (it may be signed as a SignedPart, but it is not involved in the
2335 message protection).

2336

2337 It is recognized that other uses cases might exist, where these guidelines might need further elaboration
2338 in order to address all contingencies, however that is outside the scope of the current version of this
2339 document.

2340 Note: in the discussions below the term “SamlToken” or “SamlToken assertion” refers to the
2341 policy requirement that a “SAML Assertion” be used as that token.

2342 Note: SAML Assertions have issuers. In addition, these Assertions are generally signed with an
2343 X509 certificate. In general, the SAML IssuingAuthority may be regarded as the Subject of the
2344 X509 certificate, which is trusted by a RelyingParty. In general, as well, there is usually a known
2345 mapping between the Issuer identified within the SAML Assertion and the Subject of the X509
2346 certificate used to sign that Assertion. It will be assumed in this document that the SAML
2347 IssuingAuthority may be identified by either of these identities and that, in general, a RelyingParty
2348 will check the details as necessary.

2349 The concept of the sp:“message signature” within the context of the **sp:TransportBinding** is not clearly
2350 identified within the current version of [[WS-SecurityPolicy](#)], however, there are certain inferences that are
2351 used in the use cases that follow that are identified here for reference.

- 2352 • Based on the diagrams in section 8 of [[WS-SecurityPolicy](#)], which show messages with a
2353 “message signature” followed by a diagram showing use of transport security the only difference
2354 is that the message signature diagrams contain a block labelled “Sig1”, which is the message
2355 signature, and the transport security diagrams do not have this block.
- 2356 • Considering the fact that when [[SSL](#)] Client Certificates are used for client authentication, that the
2357 client uses the client certificate private key to sign hash of SSL handshake messages in an SSL
2358 CertificateVerify message that is part of the SSL protocol, the assumption is made that
2359 subsequent data sent by the client on this link is effectively protected for the purposes of data

- 2360 integrity and confidentiality, and that the data sent on the link is authorized to be sent by the
 2361 holder of the private key associated with the client certificate.
- 2362 • Based on the considerations in the bullets above, and with intention of maintaining functional
 2363 consistency with the WS-SecurityPolicy notions of sp: message signature,
 2364 sp:SignedSupportingTokens, and sp:SignedEndorsingSupportingTokens, use of client certificates
 2365 with SSL will be considered equivalent to having a “virtual message signature” provided by the
 2366 Initiator’s client certificate which covers all the data in the SOAP request that the Initiator sends
 2367 on the SSL link.
 - 2368 • As such, in the above context, the client certificate may be regarded as providing both a virtual
 2369 Signing function for tokens and Timestamp that appear in the wsse:Security header, as well as a
 2370 virtual Endorsing function for tokens that contain the client certificate or a reference to the client
 2371 certificate that appear in the wsse:Security header.

2372 The net effect of the above assumptions for the SAML use cases in this section that use the
 2373 **sp:TransportBinding** is that if a **client certificate is required** to be used with the **sp:HttpsToken**
 2374 assertion then:

- 2375 1. A SAML **sender-vouches** Assertion contained in a wsse:Security header block may be
 2376 considered to be an **sp:SignedSupportingToken** when used in an sp:TransportBinding
 2377 using an sp:HttpsToken assertion that contains an sp:RequireClientCertificate assertion,
 2378 because the client certificate is being used as the IssuingAuthority behind the SAML
 2379 sender-vouches Assertion, which requires the IssuingAuthority to sign the combined
 2380 message and Assertion.
- 2381 2. A SAML **holder-of-key** Assertion contained in a wsse:Security header block may be
 2382 considered to be an **sp:SignedEndorsingSupportingToken** when used in an
 2383 sp:TransportBinding using an sp:HttpsToken assertion that contains an
 2384 sp:RequireClientCertificate assertion AND that the either the client certificate or a
 2385 reference to it is contained in the saml:SubjectConfirmationData/ds:KeyInfo element of
 2386 the SAML holder-of-key Assertion.
- 2387 3. A SAML **bearer** Assertion contained in a wsse:Security header block may only be
 2388 considered to be an **sp:SupportingToken**, because they are only “incidentally” covered
 2389 by the virtual message signature as a “pass through” token provided by the Requestor to
 2390 be evaluated by the Recipient that is being “passed through” by the Initiator, but which
 2391 the Initiator takes no responsibility.

2392 Ultimately, the responsibilities associated with the granting access to resources is determined by the
 2393 agreements between RelyingParties and IssuingAuthorities. Those agreements need to take into
 2394 consideration the security characteristics of the bindings which support access requests as to what kind
 2395 of bindings are required to “adequately protect” the requests for the associated business purposes. These
 2396 examples are intended to show how SAML Assertions can be used in a variety of WS-SecurityPolicy
 2397 contexts and how the different SAML token types (hk, sv, bearer) may be used in different configurations
 2398 relating the IssuingAuthority, the Requestor, the Initiator, the Recipient, and the RelyingParty.

2399 **2.3.1 WSS 1.0 SAML Token Scenarios**

2400 **2.3.1.1 (WSS1.0) SAML1.1 Assertion (Bearer)**

2401 Initiator adds a SAML assertion (bearer) representing the Requestor to the SOAP security header.

2402 Since the SamlToken is listed in the SupportingTokens element, it will not explicitly be covered by a
 2403 message signature. The Initiator may infer that a Saml Bearer Assertion is acceptable to meet this
 2404 requirement, because the Initiator is not required to explicitly cover a SupportingToken with a signature.

2405 The SAML assertion itself could be signed. The IssuingAuthority in this scenario is the Issuer (and signer,
 2406 if the Assertion is signed) of the SAML Assertion. The Initiator simply passes the token through and is not
 2407 actively involved in the trust relationship between the IssuingAuthority that issued the SAML Assertion
 2408 and the Requestor who is the Subject of the SAML Assertion.

2409 This scenario might be used in a SAML Federation application where a browser-based user with a SAML
2410 Assertion indicating that user's SSO (Single Sign On) credential has submitted a request to a portal using
2411 the Assertion as a credential (either directly or indirectly via a SAML Artifact [SAML11]), and the portal as
2412 Initiator is generating a web service request on behalf of this user, but the trust that the Recipient has for
2413 the Requestor is based on the Assertion and its IssuingAuthority, not the Initiator who delivered the
2414 request.

2415

```
2416 (P001) <wsp:Policy>  
2417 (P002)   <sp:SupportingTokens>  
2418 (P003)   <wsp:Policy>  
2419 (P004)   <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
2420         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
2421 (P005)   <wsp:Policy>  
2422 (P006)   <sp:WssSamlV11Token10/>  
2423 (P007)   </wsp:Policy>  
2424 (P008)   </sp:SamlToken>  
2425 (P009)   </wsp:Policy>  
2426 (P010)   </sp:SupportingTokens>  
2427 (P011) </wsp:Policy>
```

2428

2429 Lines (P001)-(P011) contain the endpoint wsp:Policy, which contains no bindings because none is
2430 required to access the service protected by this policy, however the service does require that the
2431 Requestor provide a Supporting Token in the sp:SupportingTokens element (P002)-(P010), which must
2432 be an sp:SamlToken (P004)-(P008), which must be an sp:WssSamlV11Token10 (P006), which means
2433 that the SamlToken must be a SAML 1.1 saml:Assertion that is sent in compliance with the
2434 [WSS10-SAML11-PROFILE].

2435 As explained in section 2.3 above, the fact that the sp:SamlToken is simply in an sp:SupportingTokens
2436 element indicates to the Initiator that a SAML bearer Assertion is what is expected to accompany the
2437 request.

2438 The following is a sample request taken from [WSS11-SAML1120-PROFILE] that complies with the
2439 WSS 1.0 compatible policy above

2440

```
2441 (M001) <S12:Envelope xmlns:S12="...">  
2442 (M002)   <S12:Header>  
2443 (M003)     <wsse:Security xmlns:wsse="...">  
2444 (M004)       <saml:Assertion xmlns:saml="...">  
2445 (M005)         AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"  
2446 (M006)         IssueInstant="2003-04-17T00:46:02Z"  
2447 (M007)         Issuer="www.opensaml.org"  
2448 (M008)         MajorVersion="1"  
2449 (M009)         MinorVersion="1">  
2450 (M010)       <saml:AuthenticationStatement>  
2451 (M011)         <saml:Subject>  
2452 (M012)           <saml:NameIdentifier  
2453 (M013)             NameQualifier="www.example.com"  
2454 (M014)             Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">  
2455 (M015)               uid=joe,ou=people,ou=saml-demo,o=baltimore.com  
2456 (M016)             </saml:NameIdentifier>  
2457 (M017)           <saml:SubjectConfirmation>  
2458 (M018)             <saml:ConfirmationMethod>  
2459 (M019)               urn:oasis:names:tc:SAML:1.0:cm:bearer  
2460 (M020)             </saml:ConfirmationMethod>  
2461 (M021)           </saml:SubjectConfirmation>  
2462 (M022)         </saml:Subject>  
2463 (M023)       </saml:AuthenticationStatement>  
2464 (M024)     </saml:Assertion>  
2465 (M025)   </wsse:Security>  
2466 (M026) </S12:Header>  
2467 (M027) <S12:Body>
```

```
2468 (M028) . . .
2469 (M029) </S12:Body>
2470 (M030) </S12:Envelope>
```

2471

2472 Lines (M001)-(M030) contains the SOAP:Envelope, which contains the SOAP:Header (M002)-(M026)
2473 and the SOAP:Body (M027)-(M029).

2474 The SOAP Header contains a wsse:Security header block (M003)-(M025) which simply contains the
2475 saml:Assertion.

2476 The saml:Assertion (M004)-(M024) is Version 1.1 (M008)-(M009), which is required by the policy
2477 sp:WssSamlV11Token10 assertion (P006). The Assertion contains a saml:AuthenticationStatement
2478 (M010)-(M023) that contains a saml:NameIdentifier (M012)-(M016) that identifies the saml:Subject, who is
2479 the Requestor (who submitted the request from a browser) and it contains a saml:SubjectConfirmation
2480 element (M017)-(M021), which specifies the saml:ConfirmationMethod to be "bearer" (M019), which
2481 meets the policy requirement (P006).

2482 For general context to relate this scenario to Figure 1, the Requestor at a browser will have obtained
2483 either the saml:Assertion above or an Artifact identifying that Assertion from an IssuingAuthority and sent
2484 it to the portal in the context of some request the Requestor is trying to make. The portal recognizes that
2485 to meet the needs of this request that it must invoke a web service that has publicized the policy above
2486 (P001)-(P011). Therefore, the portal will now take on the role of Initiator (Fig 1) and assemble a SOAP
2487 request (M001)-(M030) and submit it to the service as described in detail above.

2488 2.3.1.2 (WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL

2489 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #2](#)
2490 section 4.4.4]

2491 Initiator adds a SAML Assertion (sv) to the SOAP Security Header. Because the TransportBinding
2492 requires a Client Certificate AND the SAML token is in a SignedSupportingTokens element, when the
2493 Initiator uses the Client Certificate to protect the message under SSL, the Initiator may be considered as
2494 effectively "signing" the SAML sv Assertion and acting as a SAML Authority (i.e. the issuer of the sv
2495 Assertion). By including the sv assertion in the header and using the Client Certificate to protect the
2496 message, the Initiator takes responsibility for binding the Requestor, who is the Subject of the Assertion
2497 to the contents of the message.

2498 Note: because SSL does not retain cryptographic protection of the message after the message is
2499 delivered, messages protected using only this mechanism cannot be used as the basis for
2500 non-repudiation.

2501

```
2502 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2503 wsu:Id="Wss10SamlSvV11Tran_policy">
2504 (P002) <sp:TransportBinding>
2505 (P003) <wsp:Policy>
2506 (P004) <sp:TransportToken>
2507 (P005) <wsp:Policy>
2508 (P006) <sp:HttpsToken>
2509 (P007) <wsp:Policy>
2510 (P008) <sp:RequireClientCertificate>
2511 (P009) </wsp:Policy>
2512 (P010) </sp:HttpsToken>
2513 (P011) </wsp:Policy>
2514 (P012) </sp:TransportToken>
2515 (P013) <sp:AlgorithmSuite>
2516 (P014) <wsp:Policy>
2517 (P015) <sp:Basic256 />
2518 (P016) </wsp:Policy>
2519 (P017) </sp:AlgorithmSuite>
2520 (P018) <sp:Layout>
2521 (P019) <wsp:Policy>
2522 (P020) <sp:Strict />
```

```

2523 (P021)      </wsp:Policy>
2524 (P022)      </sp:Layout>
2525 (P023)      <sp:IncludeTimestamp />
2526 (P024)      </wsp:Policy>
2527 (P025)     </sp:TransportBinding>
2528 (P026)     <sp:SignedSupportingTokens>
2529 (P027)      <wsp:Policy>
2530 (P028)      <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
2531 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient" >
2532 (P029)      <wsp:Policy>
2533 (P030)      <sp:WssSamlV11Token10/>
2534 (P031)      </wsp:Policy>
2535 (P032)      </sp:SamlToken>
2536 (P033)     </wsp:Policy>
2537 (P034)     </sp:SignedSupportingTokens>
2538 (P035) </wsp:Policy>

```

2539

2540 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
2541 by a secure transport protocol such as SSL or TLS.

2542 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
2543 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
2544 transport token using the Basic256 algorithm suite (P015).

2545 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of
2546 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section
2547 4.3.1].

2548 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)
2549 since there are no cryptographic tokens required to be present in the WS-Security header.

2550 A timestamp (P023) is required to be included in an acceptable message.

2551 Lines (P026)-(P034) contain a SignedSupportingTokens assertion, which indicates that the referenced
2552 token is effectively "signed" by the combination of usage of the client certificate for SSL authentication
2553 and the cryptographic protection of SSL. However, the "signed" characteristic will not be present after the
2554 message is delivered from the transport layer to the recipient.

2555 Lines (P028)-(P032) indicate the signed token is a SAML Assertion (sender-vouches as described above
2556 in section 2.3) and on line (P030) that it is a SAML 1.1 Assertion and that the WS-Security 1.0 SAML
2557 Profile [[WSS10-SAML11-PROFILE](#)] is being used.

2558 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP](#) "Scenario #2 -
2559 Sender-Vouches: Unsigned: SSL" section 4.4.4]

2560 Here is the example request from that scenario:

```

2561 (M001) <?xml version="1.0" encoding="utf-8" ?>
2562 (M002) <soap:Envelope
2563 (M003)   xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
2564 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2565 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2566 (M006)   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401
2567 (M007)   -wss-wssecurity-secext-1.0.xsd"
2568 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
2569 (M009)   -wssecurity-utility-1.0.xsd">
2570 (M010) <soap:Header>
2571 (M011)   <wss:Security soap:mustUnderstand="1">
2572 (M012)     <wsu:Timestamp>
2573 (M013)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2574 (M014)     </wsu:Timestamp>
2575 (M015)     <saml:Assertion
2576 (M016)       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2577 (M017)       MajorVersion="1" MinorVersion="1"
2578 (M018)       AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
2579 (M019)       Issuer=www.opensaml.org

```

```

2580 (M020) IssueInstant="2002-06-19T16:58:33.173Z">
2581 (M021) <saml:Conditions
2582 (M022) NotBefore="2002-06-19T16:53:33.173Z"
2583 (M023) NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2584 (M024) <saml:AuthenticationStatement
2585 (M025) AuthenticationMethod=
2586 (M026) "urn:oasis:names:tc:SAML:1.0:am:password"
2587 (M027) AuthenticationInstant="2002-06-19T16:57:30.000Z">
2588 (M028) <saml:Subject>
2589 (M029) <saml:NameIdentifier
2590 (M030) NameQualifier=www.example.com
2591 (M031) Format="">
2592 (M032) uid=joe,ou=people,ou=saml-demo,o=example.com
2593 (M033) </saml:NameIdentifier>
2594 (M034) <saml:SubjectConfirmation>
2595 (M035) <saml:ConfirmationMethod>
2596 (M036) urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2597 (M037) </saml:ConfirmationMethod>
2598 (M038) </saml:SubjectConfirmation>
2599 (M039) </saml:Subject>
2600 (M040) </saml:AuthenticationStatement>
2601 (M041) </saml:Assertion>
2602 (M042) </wsse:Security>
2603 (M043) </soap:Header>
2604 (M044) <soap:Body>
2605 (M045) <Ping xmlns="http://xmlsoap.org/Ping">
2606 (M046) <text>EchoString</text>
2607 (M047) </Ping>
2608 (M048) </soap:Body>
2609 (M049) </soap:Envelope>

```

2610

2611 Lines (M011)-(M042) contain the WS-Security 1.0 header required by the WssSamlV11Token10
2612 assertion (P030).

2613 Lines (M012)-(M014) contain the wsu:Timestamp required by IncludeTimestamp assertion (P023).

2614 Lines (M015)-(M041) contain the SAML 1.1 Assertion required by the WssSamlV11Token10 assertion
2615 (P030).

2616 Note that the additional requirements identified in the policy above are met by the SSL transport
2617 capabilities and do not appear in any form in the SOAP message (M001)-(M049).

2618 **2.3.1.3 (WSS1.0) SAML1.1 Assertion (HK) over SSL**

2619 Initiator adds a SAML assertion (hk) to the SOAP Security Header. Because the TransportBinding
2620 requires a Client Certificate AND the SAML token is in an EndorsingSupportingTokens element, the
2621 Initiator may be considered to be authorized by the issuer of the hk SAML assertion to bind message
2622 content to the Subject of the assertion. If the Client Certificate matches the certificate identified in the hk
2623 assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor,
2624 who would be the Subject of the hk assertion, to the content of the message.

2625 In this scenario, the IssuingAuthority is the issuer(signer) of the hk SAML Assertion. The Requestor is the
2626 Subject of the Assertion and the Initiator is authorized by the Authority to bind the Assertion to the
2627 message using the ClientCertificate identified in the SAML Assertion, which should also be used to sign
2628 the timestamp of the message with a separate Signature included in the WS-Security header.

2629

```

2630 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2631 (P002) wsu:Id="Wss10SamlHkV11Tran_policy">>
2632 (P003) <sp:TransportBinding>
2633 (P004) <wsp:Policy>
2634 (P005) <sp:TransportToken>
2635 (P006) <wsp:Policy>
2636 (P007) <sp:HttpsToken>

```

```

2637 (P008) <wsp:Policy>
2638 (P009) <sp:RequireClientCertificate>
2639 (P010) </wsp:Policy>
2640 (P011) </sp:HttpsToken>
2641 (P012) </wsp:Policy>
2642 (P013) </sp:TransportToken>
2643 (P014) <sp:AlgorithmSuite>
2644 (P015) <wsp:Policy>
2645 (P016) <sp:Basic256 />
2646 (P017) </wsp:Policy>
2647 (P018) </sp:AlgorithmSuite>
2648 (P019) <sp:Layout>
2649 (P020) <wsp:Policy>
2650 (P021) <sp:Strict />
2651 (P022) </wsp:Policy>
2652 (P023) </sp:Layout>
2653 (P024) <sp:IncludeTimestamp />
2654 (P025) </wsp:Policy>
2655 (P026) </sp:TransportBinding>
2656 (P027) <sp:SignedEndorsingSupportingTokens>
2657 (P028) <wsp:Policy>
2658 (P029) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2659 open.org/ws-sx/ws-
2660 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2661 (P030) <wsp:Policy>
2662 (P031) <sp:WssSamlV11Token10/>
2663 (P032) </wsp:Policy>
2664 (P033) </sp:SamlToken>
2665 (P034) </wsp:Policy>
2666 (P035) </sp:SignedEndorsingSupportingTokens>
2667 (P036) <wsp:Policy>

```

2668

2669 Section 2.3.2.3 contains an example message that is similar to one that could be used for the policy
2670 above, except the Signed Endorsing Supporting Token there is a SAML Version 2.0 token, and a SAML
2671 Version 1.1 token would be required here.

2672 **2.3.1.4 (WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional** 2673 **Encrypt**

2674 This scenario is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #3](#)].
2675 In this case, the SAML token is included as part of the message signature and sent only to the Recipient.
2676 The message security is provided using X.509 certificates with both requestor and service having
2677 exchanged these credentials via some out of band mechanism, which provides the basis of trust of each
2678 others' certificates.

2679 In this scenario the SAML Authority is the Initiator who uses the message signature to both provide the
2680 integrity of the message and to establish the Initiator as the SAML Authority based on the X509 certificate
2681 used to sign the message and the SignedSupportingTokens SAML Assertion. Effectively, the SAML
2682 Assertion is being "issued" as part of the message construction process. The Requestor is the Subject of
2683 the SAML Assertion. The Initiator knows that the Recipient is expecting it to be the SAML Authority by the
2684 fact that the policy specifies that the message requires a SignedSupportingTokens SamlToken.

2685

```

2686 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2687 (P002) wsu:Id="Wss10SamlSvV11Asymm_endpoint_policy">
2688 (P003) <wsp:ExactlyOne>
2689 (P004) <wsp:All>
2690 (P005) <sp:AsymmetricBinding>
2691 (P006) <wsp:Policy>
2692 (P007) <sp:InitiatorToken>
2693 (P008) <wsp:Policy>

```

```

2694 (P009) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2695 open.org/ws-sx/ws-
2696 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2697 (P010) <wsp:Policy>
2698 (P011) <sp:WssX509V3Token10/>
2699 (P012) </wsp:Policy>
2700 (P013) </sp:X509Token>
2701 (P014) </wsp:Policy>
2702 (P015) </sp:InitiatorToken>
2703 (P016) <sp:RecipientToken>
2704 (P017) <wsp:Policy>
2705 (P018) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2706 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
2707 (P019) <wsp:Policy>
2708 (P020) <sp:WssX509V3Token10/>
2709 (P021) </wsp:Policy>
2710 (P022) </sp:X509Token>
2711 (P023) </wsp:Policy>
2712 (P024) </sp:RecipientToken>
2713 (P025) <sp:AlgorithmSuite>
2714 (P026) <wsp:Policy>
2715 (P027) <sp:Basic256/>
2716 (P028) </wsp:Policy>
2717 (P029) </sp:AlgorithmSuite>
2718 (P030) <sp:Layout>
2719 (P031) <wsp:Policy>
2720 (P032) <sp:Strict/>
2721 (P033) </wsp:Policy>
2722 (P034) </sp:Layout>
2723 (P035) <sp:IncludeTimestamp/>
2724 (P036) <sp:OnlySignEntireHeadersAndBody/>
2725 (P037) </wsp:Policy>
2726 (P038) </sp:AsymmetricBinding>
2727 (P039) <sp:Wss10>
2728 (P040) <wsp:Policy>
2729 (P041) <sp:MustSupportRefKeyIdentifier/>
2730 (P042) </wsp:Policy>
2731 (P043) </sp:Wss10>
2732 (P044) <sp:SignedSupportingTokens>
2733 (P045) <wsp:Policy>
2734 (P046) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2735 open.org/ws-sx/ws-
2736 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2737 (P047) <wsp:Policy>
2738 (P048) <sp:WssSamlV11Token10/>
2739 (P049) </wsp:Policy>
2740 (P050) </sp:SamlToken>
2741 (P051) </wsp:Policy>
2742 (P052) </sp:SignedSupportingTokens>
2743 (P053) </wsp:All>
2744 (P054) </wsp:ExactlyOne>
2745 (P055) </wsp:Policy>
2746 (P056)
2747 (P057) <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_input_policy">
2748 (P058) <wsp:ExactlyOne>
2749 (P059) <wsp:All>
2750 (P060) <sp:SignedParts>
2751 (P061) <sp:Body/>
2752 (P062) </sp:SignedParts>
2753 (P063) <sp:EncryptedParts wsp:Optional="true">
2754 (P064) <sp:Body/>
2755 (P065) </sp:EncryptedParts>
2756 (P066) </wsp:All>
2757 (P067) </wsp:ExactlyOne>

```

```

2758      (P068)    </wsp:Policy>
2759
2760      (P069)    <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_output_policy">
2761      (P070)      <wsp:ExactlyOne>
2762      (P071)        <wsp:All>
2763      (P072)          <sp:SignedParts>
2764      (P073)            <sp:Body/>
2765      (P074)          </sp:SignedParts>
2766      (P075)          <sp:EncryptedParts>
2767      (P076)            <sp:Body/>
2768      (P077)          </sp:EncryptedParts>
2769      (P078)        </wsp:All>
2770      (P079)      </wsp:ExactlyOne>
2771      (P080)    </wsp:Policy>

```

2772 Line (P004) is a `wsp:All` element whose scope carries down to line (P053), which means all 3 of the
2773 contained assertions: (P005)-(P038) `AsymmetricBinding`, (P039)-(P043) `WSS10`, and (P044)-(P052)
2774 `SignedSupportingTokens` must be complied with by message exchanges to a Web Service covered by
2775 this policy.

2776 Lines (P005)-(P038) contain an `AsymmetricBinding` assertion which indicates that the Initiator's token
2777 must be used for the message signature and that the recipient's token must be used for message
2778 encryption.

2779 Lines (P007)-(P015) contain the `InitiatorToken` assertion. Within the `InitiatorToken` assertion, lines
2780 (P009)-(P013) indicate that the Initiator's token must be an `X509Token` that must always be included as
2781 part of the request message (as opposed to an external reference). Line (P011) indicates that the `X509`
2782 token must be an `X.509 V3` signature-verification certificate and used in the manner described in
2783 [\[WSS10-X509-PROFILE\]](#).

2784 Lines (P016)-(P023) contain the `RecipientToken` assertion. Again, this an `X.509 V3` certificate (P020) that
2785 must be used as described in [\[WSS10-X509-PROFILE\]](#), however the token itself, must never be included
2786 in messages in either direction (P018).

2787 Instead (momentarily skipping slightly ahead), policy lines (P039)-(P043), the `WSS10` assertion, indicate
2788 that the Recipient's token must be referenced by a `KeyIdentifier` element, which is described in the `WS-`
2789 `Security 1.0` core specification [\[WSS10-SOAPMSG\]](#).

2790 Lines (P025)-(P029) contain the `AlgorithmSuite` assertion, which specifies the `sp:Basic256` set of
2791 cryptographic components. The relevant values for this example within the `sp:Basic256` components are
2792 the asymmetric signing algorithm, `ds:rsa-sha1`, and the digest algorithm, `ds:sha1`, where "ds:" =
2793 "http://www.w3.org/2000/09/xmlsig#" [\[XML-DSIG\]](#).

2794 Lines (P030)-(P034) contain the `sp:Layout` assertion, which is set to `sp:Strict` (P032), which governs the
2795 required relative layout of various `Timestamp`, `Signature` and `Token` elements in the messages.

2796 Line (P035) `IncludeTimestamp` means a `Timestamp` element must be included in the `WS-Security` header
2797 block and signed by the message signature for messages in both directions.

2798 Line (P036) `OnlySignEntireHeaderAndBody` indicates that the message signature must explicitly cover
2799 the `SOAP:Body` element (not children), and if there are any signed elements in the `SOAP:Header` they
2800 must be direct child elements of the `SOAP:Header`. However, the one exception where the latter condition
2801 is not applied is that if the child of the `SOAP:Header` is a `WS-Security` header (i.e. a `wsse:Security`
2802 element), then individual direct child only elements of that `Security` element may also be signed.

2803 Lines (P044)-(P052) contain a `SignedSupportingTokens` assertion, which contains only one token
2804 (P046)-(P050), a `SamlToken`, which must always be sent to the Recipient (P046) and it must be a
2805 `SAML 1.1 Assertion` token. Because the `SamlToken` is in a "SignedSupportingTokens" element, it is
2806 implicitly a `SAML sender-vouches ConfirmationMethod` token as described in section 2.3 above.

2807 Lines (P057)-(P068) contain a policy that applies only to input messages from the Initiator to the
2808 Recipient.

2809 Lines (P060)-(P062) specify that the `SOAP:Body` element of the input message must be signed by the
2810 Initiator's message signature.

2811 Lines (P063)-(P065) specify that the SOAP:Body element of the input message may optionally be
2812 encrypted (signified by wsp:Optional="true") using the Recipient's encryption token (in this case (P020), it
2813 is an X.509 certificate).

2814 Lines (P069)-(P080) contain a policy that applies only to output messages from the Recipient to the
2815 Initiator.

2816 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be signed by the
2817 Recipient's message signature.

2818 Lines (P075)-(P077) specify that the SOAP:Body element of the output message must be encrypted by
2819 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

2820 Here is an example request:

2821

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
2822 (M002) <S12:Envelope
2823 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
2824 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2825 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2826 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
2827 (M007) 200401-wss-wssecurity-secext-1.0.xsd"
2828 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
2829 (M009) 200401-wss-wssecurity-utility-1.0.xsd"
2830 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#
2831 (M011)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
2832 (M012) <S12:Header>
2833 (M013)   <wsse:Security S12:mustUnderstand="1">
2834 (M014)     <wsu:Timestamp wsu:Id="timestamp">
2835 (M015)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2836 (M016)     </wsu:Timestamp>
2837 (M017)     <saml:Assertion
2838 (M018)       AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2839 (M019)       IssueInstant="2003-04-17T00:46:02Z"
2840 (M020)       Issuer=www.opensaml.org
2841 (M021)       MajorVersion="1"
2842 (M022)       MinorVersion="1">
2843 (M023)       <saml:Conditions
2844 (M024)         NotBefore="2002-06-19T16:53:33.173Z"
2845 (M025)         NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2846 (M026)       <saml:AttributeStatement>
2847 (M027)         <saml:Subject>
2848 (M028)           <saml:NameIdentifier
2849 (M029)             NameQualifier=www.example.com
2850 (M030)             Format="">
2851 (M031)             uid=joe,ou=people,ou=saml-demo,o=example.com
2852 (M032)           </saml:NameIdentifier>
2853 (M033)         <saml:SubjectConfirmation>
2854 (M034)           <saml:ConfirmationMethod>
2855 (M035)             urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2856 (M036)           </saml:ConfirmationMethod>
2857 (M037)         </saml:SubjectConfirmation>
2858 (M038)       </saml:Subject>
2859 (M039)     <saml:Attribute>
2860 (M040)       ...
2861 (M041)     </saml:Attribute>
2862 (M042)     ...
2863 (M043)   </saml:AttributeStatement>
2864 (M044) </saml:Assertion>
2865 (M045) <wsse:SecurityTokenReference wsu:id="STR1">
2866 (M046)   <wsse:KeyIdentifier wsu:id="..."
2867 (M047)     ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
2868 (M048) 2004XXwss-saml-token-profile-1.0#SAMLAssertionID">
2869 (M048)   _a75adf55-01d7-40cc-929f-dbd8372ebdfc
2870 (M049) </wsse:KeyIdentifier>
2871
```



```

2872 (M050) </wsse:SecurityTokenReference>
2873 (M051) <wsse:BinarySecurityToken
2874 (M052)   wsu:Id="attesterCert"
2875 (M053)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2876 (M054) 200401-wss-x509-token-profile-1.0#X509v3"
2877 (M055)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
2878 (M056) 200401-wss-soap-message-security-1.0#Base64Binary">
2879 (M057)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
2880 (M058) </wsse:BinarySecurityToken>
2881 (M059) <ds:Signature>
2882 (M060)   <ds:SignedInfo>
2883 (M061)     <ds:CanonicalizationMethod
2884 (M062)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2885 (M063)     <ds:SignatureMethod
2886 (M064)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2887 (M065)     <ds:Reference URI="#STR1">
2888 (M066)       <ds:Transforms>
2889 (M067)         <ds:Transform
2890 (M068)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
2891 (M069) 200401-wss-soap-message-security-1.0#STR-Transform">
2892 (M070)           <wsse:TransformationParameters>
2893 (M071)             <ds:CanonicalizationMethod
2894 (M072)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2895 (M073)             </wsse:TransformationParameters>
2896 (M074)           </ds:Transform>
2897 (M075)         </ds:Transforms>
2898 (M076)       <ds:DigestMethod
2899 (M077)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2900 (M078)       <ds:DigestValue>...</ds:DigestValue>
2901 (M079)     </ds:Reference>
2902 (M080)     <ds:Reference URI="#MsgBody">
2903 (M081)       <ds:DigestMethod
2904 (M082)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2905 (M083)       <ds:DigestValue>...</ds:DigestValue>
2906 (M084)     </ds:Reference>
2907 (M085)     <ds:Reference URI="#timestamp">
2908 (M086)       <ds:DigestMethod
2909 (M087)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2910 (M088)       <ds:DigestValue>...</ds:DigestValue>
2911 (M089)     </ds:Reference>
2912 (M090)     </ds:SignedInfo>
2913 (M091)     <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
2914 (M092)     <ds:KeyInfo>
2915 (M093)       <wsse:SecurityTokenReference wsu:id="STR2">
2916 (M094)         <wsse:Reference wsu:id="..." URI="#attesterCert" />
2917 (M095)       </wsse:SecurityTokenReference>
2918 (M096)     </ds:KeyInfo>
2919 (M097)     </ds:Signature>
2920 (M098)   </wsse:Security>
2921 (M099) </S12:Header>
2922 (M100) <S12:Body wsu:Id="MsgBody">
2923 (M101)   <ReportRequest>
2924 (M102)     <TickerSymbol>SUNW</TickerSymbol>
2925 (M103)   </ReportRequest>
2926 (M104) </S12:Body>
2927 (M105) </S12:Envelope>

```

2928

2929 Note: For the instructional purposes of this document, in order to be compliant with WS-
2930 SecurityPolicy, this copy of the original message had to be modified from the original. In particular,
2931 the modifications that are applied above are the inclusion of a wsu:Id attribute in the Timestamp
2932 element start tag (M014) and the addition of a ds:Reference element and URI attribute identifying
2933 that Timestamp element in the message signature (M085)-(M089). (The original message in

2934 [WSS10-SAML11-INTEROP scenario #3] is not compliant with WS-SecurityPolicy because the
 2935 Timestamp that it contains is not covered by the message signature in that document.)

2936 Lines (M002)-(M0105) contain the SOAP:Envelope, i.e. the whole SOAP message.

2937 Lines (M012)-(M099) contain the SOAP:Header, which is the header portion of the SOAP message.

2938 Lines (M0100)-(M0104) contain the SOAP:Body, which is just some dummy content for test purposes.

2939 Lines (M013)-(M098) contain the wsse:Security header, which is the primary focus of this example.

2940 Lines (M014)-(M016) contain the wsu:Timestamp, which is required by the policy (P035).

2941 Lines (M017)-(M044) contain the SAML Assertion, which is the sp:SamlToken required by the policy
 2942 sp:SignedSupportingTokens (P046)-(P050).

2943 Lines (M021)-(M022) identify the SAML Assertion as being Version 1.1 as required by the policy (P048).

2944 Line (M035) identifies the SAML Assertion as having ConfirmationMethod sender-vouches, which is the
 2945 implicit requirement of the sp:SamlToken being in the SignedSupportingTokens as described above in the
 2946 policy section covering (P044)-(P052).

2947 Lines (M045)-(M050) contain a wsse:SecurityTokenReference which contains a wsse:KeyIdentifier, which
 2948 is used to reference the SAML Assertion, as described in [WSS10-SAML11-PROFILE] and required by
 2949 the policy (P041).

2950 Lines (M051)-(M058) contain the Initiator's wsse:BinarySecurityToken, which is required by the policy
 2951 (P007)-(P015), and in particular lines (M053)-(M054) identify the token as an X.509 V3 token as required
 2952 by the policy (P011). Line (M057) contains a truncated portion of the Base64 representation of the actual
 2953 certificate, which is included in the message as required by the policy sp:IncludeToken (P009).

2954 Lines (M059)-(M0102) contain the ds:Signature, which is the sp: message signature as required by
 2955 various parts of the Endpoint Policy (P001)-(P055) and Input Message Policy (P057)-(P068).

2956 Lines (M060)-(M095) contain the ds:SignedInfo element which describes the signing algorithm used
 2957 (M064) and specific elements that are signed (M065)-(M094) as required by the policies, which are
 2958 described in detail immediately following.

2959 Lines (M061)-(M062) contain the ds:CanonicalizationMethod Algorithm, which is specified as xml-exc-
 2960 c14n#, which is the default value required by the policy sp:AlgorithmSuite (P025)-(P029).

2961 Line (M064) identifies the SignatureMethod Algorithm as ds:rsa-sha1, which is the asymmetric key
 2962 signing algorithm required by the policy sp:AlgorithmSuite (P025)-(P029).

2963 Lines (M065)-(M079) identify the SAML Assertion (M017)-(M044) as an element that is signed, which is
 2964 referenced through the URI "#STR1" on line (M065), which references the SecurityTokenReference
 2965 (M045)-(M050), which in turn uses the identifier on line (M048) to reference the actual SAML Assertion by
 2966 its AssertionID attribute on line (M018). The SAML Assertion is required to be signed because it is
 2967 identified in the policy within the SignedSupportingTokens element on line (P048).

2968 Lines (M077)-(M078) contain the digest algorithm, which is specified to be sha1, as required by the policy
 2969 sp:Basic256 assertion (P027) in the sp:AlgorithmSuite.

2970 Lines (M080)-(M084) identify the SOAP:Body (M0100)-(M0104) as an element that is signed, which is
 2971 referenced through the URI "#MsgBody" on line (M080). The SOAP Body is required to be signed by the
 2972 input message policy lines (P060)-(P062).

2973 Lines (M085)-(M089) identify the wsu:Timestamp (M014)-(M016) as an element that is signed, which is
 2974 referenced through the URI "#timestamp" on line (M085). The wsu:Timestamp is required to be signed by
 2975 the policy sp:IncludeTimestamp assertion (P035). Note that the wsu:Timestamp occurs in the
 2976 wsse:Security header prior to the ds:Signature as required by the sp:Strict layout policy (P032).

2977 Finally, lines (M092)-(M096) contain the ds:KeyInfo element that identifies the signing key associated with
 2978 this ds:Signature element. The actual signing key is referenced through the
 2979 wsse:SecurityTokenReference (M093)-(M095), with URI "#attesterCert", which identifies the
 2980 InitiatorToken identified by the policy (P011) and contained in the wsse:BinarySecurityToken element
 2981 (M051)-(M058), which occurs in the wsse:Security header prior to this ds:Signature element, as required
 2982 by the sp:Strict layout policy assertion (P032). (Note: this BinarySecurityToken would need to be included

2983 in the signature, if the sp:AsymmetricBinding assertion in the endpoint policy contained a
2984 sp:ProtectTokens assertion, but it does not, so it does not need to be signed.)

2985 2.3.1.5 (WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt

2986 This example is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #4](#)].

2987 In this example the SAML token provides the key material for message security, hence acts as the
2988 Initiator token, and therefore the Requestor and Initiator may be considered to be the same entity. The
2989 SAML HK Assertion contains a reference to the public key of the signer of the message (the Initiator). The
2990 Initiator knows Recipient's public key, which it may use to encrypt the request, but the Initiator does not
2991 share a direct trust relation with the Recipient except indirectly through the SAML Assertion Issuer. The
2992 Recipient has a trust relation with the Authority that issues the SAML HK Assertion. The indirect trust
2993 relation between the Recipient and the Initiator is established by the fact that the Initiator's public key has
2994 been signed by the Authority in the SAML HK Assertion. On the request the message body is signed
2995 using Initiator's private key referenced in the SAML HK Assertion and it is optionally encrypted using
2996 Recipient's server certificate. On the response, the server signs the message using its private key and
2997 encrypts the message using the key provided within SAML HK Assertion.

2998 HK Note: there is a trust model aspect to the WS-Security holder-of-key examples that
2999 implementors may want to be aware of. The [[SAML20-CORE](#)] specification defines the Subject of
3000 the SAML Assertion such that the "presenter" of the Assertion is the entity that "attests" to the
3001 information in the Assertion. "The attesting entity and the actual Subject **may or may not** be the
3002 same entity." In general, these two choices map to the actors in [Figure 1](#) as follows: the
3003 "attesting" entity may be regarded as the Initiator. The Subject entity, about whom the information
3004 in the Assertion applies, may be regarded as the Requestor. In the case where the Subject and
3005 attestor are one and the same, the Initiator and Requestor may be regarded as one and the
3006 same. In this case the potential exists to use the Assertion for non-repudiation purposes with
3007 respect to messages that the Requestor/Initiator signs. When the Subject and attestor are
3008 separate entities, then holder-of-key is more similar to sender-vouches where the Initiator/attestor
3009 is sending the message on behalf of the Subject. The mechanisms for determining whether the
3010 Subject and attestor may be verified to be the same entity are dependent on the arrangements
3011 among the business entities and the structure of the associated application scenarios.

3012

```
3013 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."  
3014 (P002)   wsu:Id="Wss10SamlHkV11Asymm_endpoint_policy">  
3015 (P003)   <wsp:ExactlyOne>  
3016 (P004)     <wsp>All>  
3017 (P005)       <sp:AsymmetricBinding>  
3018 (P006)         <wsp:Policy>  
3019 (P007)           <sp:InitiatorToken>  
3020 (P008)             <wsp:Policy>  
3021 (P009)               <sp:SamlToken sp:IncludeToken="http://docs.oasis-  
3022 open.org/ws-sx/ws-  
3023 securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3024 (P010)                 <wsp:Policy>  
3025 (P011)                   <wsp:WssSamlV11Token10/>  
3026 (P012)                 </wsp:Policy>  
3027 (P013)               </sp:SamlToken>  
3028 (P014)             </wsp:Policy>  
3029 (P015)           </sp:InitiatorToken>  
3030 (P016)         <sp:RecipientToken>  
3031 (P017)           <wsp:Policy>  
3032 (P018)             <sp:X509Token sp:IncludeToken="http://docs.oasis-  
3033 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
3034 (P019)               <wsp:Policy>  
3035 (P020)                 <sp:WssX509V3Token10/>  
3036 (P021)               </wsp:Policy>  
3037 (P022)             </sp:X509Token>  
3038 (P023)           </wsp:Policy>  
3039 (P024)         </sp:RecipientToken>
```

```

3040 (P025) <sp:AlgorithmSuite>
3041 (P026) <wsp:Policy>
3042 (P027) <sp:Basic256/>
3043 (P028) </wsp:Policy>
3044 (P029) </sp:AlgorithmSuite>
3045 (P030) <sp:Layout>
3046 (P031) <wsp:Policy>
3047 (P032) <sp:Strict/>
3048 (P033) </wsp:Policy>
3049 (P034) </sp:Layout>
3050 (P035) <sp:IncludeTimestamp/>
3051 (P036) <sp:OnlySignEntireHeadersAndBody/>
3052 (P037) </wsp:Policy>
3053 (P038) </sp:AsymmetricBinding>
3054 (P039) <sp:Wss10>
3055 (P040) <wsp:Policy>
3056 (P041) <sp:MustSupportRefKeyIdentifier/>
3057 (P042) </wsp:Policy>
3058 (P043) </sp:Wss10>
3059 (P044) </wsp:All>
3060 (P045) </wsp:ExactlyOne>
3061 (P046) </wsp:Policy>
3062 (P047)
3063 (P048) <wsp:Policy wsu:Id="WSS10SamlHok_input_policy">
3064 (P049) <wsp:ExactlyOne>
3065 (P050) <wsp:All>
3066 (P051) <sp:SignedParts>
3067 (P052) <sp:Body/>
3068 (P053) </sp:SignedParts>
3069 (P054) <wsp:ExactlyOne>
3070 (P055) <wsp:All>
3071 (P056) <sp:EncryptedParts>
3072 (P057) <sp:Body/>
3073 (P058) </sp:EncryptedParts>
3074 (P059) </wsp:All>
3075 (P060) <wsp:All/>
3076 (P061) </wsp:ExactlyOne>
3077 (P062) </wsp:All>
3078 (P063) </wsp:ExactlyOne>
3079 (P064) </wsp:Policy>
3080 (P065)
3081 (P066) <wsp:Policy wsu:Id="WSS10SamlHok_output_policy">
3082 (P067) <wsp:ExactlyOne>
3083 (P068) <wsp:All>
3084 (P069) <sp:SignedParts>
3085 (P070) <sp:Body/>
3086 (P071) </sp:SignedParts>
3087 (P072) <sp:EncryptedParts>
3088 (P073) <sp:Body/>
3089 (P074) </sp:EncryptedParts>
3090 (P075) </wsp:All>
3091 (P076) </wsp:ExactlyOne>
3092 (P077) </wsp:Policy>

```

3093

3094 Lines (P001)-(P046) contain the endpoint policy, and lines (P048)-(P064) and (P066)-(P077) contain the
3095 message input and message output policies, respectively.

3096 Lines (P005)-(P038) contain the AsymmetricBinding assertion, which requires separate security tokens
3097 for the Initiator and Recipient.

3098 Lines (P007)-(P015) contain the InitiatorToken, which is defined to be a SamlToken (P009)-(P013), which
3099 must always be sent to the Recipient (P009). The SamlToken is further specified by the
3100 WssSamIV11Token10 assertion (P011) that requires the token to be a SAML 1.1 Assertion and the token
3101 must be used in accordance with the WS-Security [[WSS10-SAML11-PROFILE](#)]. Furthermore, as

3102 described in section 2.3 above, because the SamlToken assertion appears within an InitiatorToken
3103 assertion, the SAML Assertion must use the holder-of-key ConfirmationMethod, whereby for the indicated
3104 WS-Security profile, the SAML Assertion contains a reference to the Initiator's X.509 signing certificate or
3105 equivalent.

3106 Lines (P016)-(P024) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that
3107 must be used as described in [WSS10-SAML11-PROFILE], however the token itself, must never be
3108 included in messages in either direction (P018) (i.e not only will the Recipient not send the token, but the
3109 Initiator must not send the actual token, even when using it for encryption).

3110 Lines (P025)-(P029) AlgorithmSuite and (P030)-(P034) Layout are the same as described above in
3111 section 2.3.1.4.

3112 Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header
3113 block and signed by the message signature for messages in both directions.

3114 Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover
3115 the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they
3116 must be direct child elements of the SOAP:Header. However, the one exception where the latter condition
3117 is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security
3118 element), then individual direct child only elements of that Security element may also be signed.

3119

3120 Lines (P039)-(P043) contain the WSS10 assertion, which indicates that the Recipient's token must be
3121 referenced by a KeyIdentifier element (P041), the usage of which is described in the WS-Security 1.0
3122 core specification [[WSS10-SOAPMSG](#)].

3123 Lines (P048)-(P064) contain the message input policy that applies only to messages from the Initiator to
3124 the Recipient.

3125 Lines (P051)-(P053) specify that the SOAP:Body element of the input message must be signed by the
3126 Initiator's message signature.

3127 Lines (P054)-(P061) specify that the SOAP:Body element of the input message may optionally (signified
3128 by the empty policy alternative on line (P060)) be encrypted using the Recipient's encryption token (in this
3129 case (P020), it is an X.509 certificate).

3130 Note: Because the input policy above (P048-P064) has the EncryptedParts assertion
3131 (P056-P058) contained in an <ExactlyOne> element (P054-P061), which also contains an empty
3132 policy element (P060), either a message with an encrypted Body element or an unencrypted
3133 Body element will be accepted.

3134 Lines (P066)-(P077) contain a message output policy that applies only to messages from the Recipient to
3135 the Initiator.

3136 Lines (P069)-(P071) specify that the SOAP:Body element of the output message must be signed by the
3137 Recipient's message signature.

3138 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be encrypted by
3139 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

3140

3141 The following example request is taken from the [[WSS10-SAML11-INTEROP](#)] document scenario #4:

3142

```
3143 (M001) <?xml version="1.0" encoding="utf-8" ?>
3144 (M002) <S12:Envelope
3145 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
3146 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
3147 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
3148 (M006)   xmlns:wsse=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
3149 (M007)   wss-wssecurity-secext-1.0.xsd"
3150 (M008)   xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
3151 (M009)   wssecurity-utility-1.0.xsd"
3152 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#">
3153 (M011)   <S12:Header>
```

```

3154 (M012) <wsse:Security S12:mustUnderstand="1">
3155 (M013)   <wsu:Timestamp wsu:Id="timestamp">
3156 (M014)     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
3157 (M015)   </wsu:Timestamp>
3158 (M016)   <saml:Assertion
3159 (M017)     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3160 (M018)     IssueInstant="2003-04-17T00:46:02Z"
3161 (M019)     Issuer=www.opensaml.org
3162 (M020)     MajorVersion="1"
3163 (M021)     MinorVersion="1"
3164 (M022)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
3165 (M023)   <saml:Conditions
3166 (M024)     NotBefore="2002-06-19T16:53:33.173Z"
3167 (M025)     NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
3168 (M026)   <saml:AttributeStatement>
3169 (M027)     <saml:Subject>
3170 (M028)       <saml:NameIdentifier
3171 (M029)         NameQualifier=www.example.com
3172 (M030)         Format="">
3173 (M031)         uid=joe,ou=people,ou=saml-demo,o=example.com
3174 (M032)       </saml:NameIdentifier>
3175 (M033)     <saml:SubjectConfirmation>
3176 (M034)       <saml:ConfirmationMethod>
3177 (M035)         urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
3178 (M036)       </saml:ConfirmationMethod>
3179 (M037)     <ds:KeyInfo>
3180 (M038)       <ds:KeyValue>...</ds:KeyValue>
3181 (M039)     </ds:KeyInfo>
3182 (M040)   </saml:SubjectConfirmation>
3183 (M041) </saml:Subject>
3184 (M042)   <saml:Attribute
3185 (M043)     AttributeName="MemberLevel"
3186 (M044)     AttributeNamespace=
3187 (M045)       "http://www.oasis-open.org/Catalyst2002/attributes">
3188 (M046)     <saml:AttributeValue>gold</saml:AttributeValue>
3189 (M047)   </saml:Attribute>
3190 (M048)   <saml:Attribute
3191 (M049)     AttributeName="E-mail"
3192 (M050)     AttributeNamespace="http://www.oasis-
3193 (M051)       open.org/Catalyst2002/attributes">
3194 (M052)     <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
3195 (M053)   </saml:Attribute>
3196 (M054) </saml:AttributeStatement>
3197 (M055) <ds:Signature>...</ds:Signature>
3198 (M056) </saml:Assertion>
3199 (M057) <ds:Signature>
3200 (M058) <ds:SignedInfo>
3201 (M059)   <ds:CanonicalizationMethod
3202 (M060)     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
3203 (M061)   <ds:SignatureMethod
3204 (M062)     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3205 (M063)   <ds:Reference URI="#MsgBody">
3206 (M064)     <ds:DigestMethod
3207 (M065)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3208 (M066)     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
3209 (M067)   </ds:Reference>
3210 (M068)   <ds:Reference URI="#timestamp">
3211 (M069)     <ds:DigestMethod
3212 (M070)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3213 (M071)     <ds:DigestValue>...</ds:DigestValue>
3214 (M072)   </ds:Reference>
3215 (M073) </ds:SignedInfo>
3216 (M074) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
3217 (M075) <ds:KeyInfo>

```

```

3218 (M076) <wsse:SecurityTokenReference wsu:id="STR1">
3219 (M077) <wsse:KeyIdentifier wsu:id="..."
3220 (M078) Value="http://docs.oasis-open.org/wss/2004/XX/oasis-
3221 (M079) 2004XX-wss-saml-token-profile-1.0#SAMLAssertionID">
3222 (M080) _a75adf55-01d7-40cc-929f-dbd8372ebdfc
3223 (M081) </wsse:KeyIdentifier>
3224 (M082) </wsse:SecurityTokenReference>
3225 (M083) </ds:KeyInfo>
3226 (M084) </ds:Signature>
3227 (M085) </wsse:Security>
3228 (M086) </S12:Header>
3229 (M087) <S12:Body wsu:Id="MsgBody">
3230 (M088) <ReportRequest>
3231 (M089) <TickerSymbol>SUNW</TickerSymbol>
3232 (M090) </ReportRequest>
3233 (M091) </S12:Body>
3234 (M092) </S12:Envelope>

```

3235
3236 Note: for the instructional purposes of this document, it was necessary to make changes to the
3237 original sample request to meet the requirements of WS-SecurityPolicy. The changes to the
3238 message above include:

- 3239 • Line (M013): added wsu:Id="timestamp" attribute to sp:Timestamp element.
- 3240 • Lines (M068)-(M072): added a ds:Reference element to include the Timestamp in the
3241 message signature required by the policy sp:IncludeTimestamp assertion (P035).

3242 Lines (M002)-(M092) contain the SOAP Envelope, i.e. the whole SOAP message.

3243 Lines (M011)-(M086) contain the SOAP Header.

3244 Lines (M087)-(M091) contain the unencrypted SOAP Body, which is allowed to be unencrypted based on
3245 line (P060) of the input message policy, which is the alternative choice to encrypting based on lines
3246 (P055)-(P059).

3247 Lines (M012)-(M080) contain the wsse:Security header entry, which is the only header entry in this SOAP
3248 Header.

3249 Lines (M013)-(M015) contain the wsu:Timestamp which is required by the endpoint policy (P035).

3250 Lines (M016)-(M056) contain the SAML Assertion, which is required in the policy by the SamlToken
3251 (P009)-(P013) contained in the InitiatorToken. The SAML Assertion is version 1.1 (M020)-(M021) as
3252 required by the sp:WssSamlV11Token10 assertion (P011). The SAML Assertion is of type that uses the
3253 holder-of-key saml:ConfirmationMethod [[SAML11-CORE](#)] (M034)-(M036) as required by the fact that the
3254 SamlToken is contained in the InitiatorToken assertion of the policy (P009)-(P013), as explained in
3255 section 2.3 above.

3256 Line (M055) contains the ds:Signature of the Issuer of the SAML Assertion. While the details of this
3257 signature are not shown, it is this signature that the Recipient must ultimately trust in order to trust the rest
3258 of the message.

3259 Lines (M057)-(M084) contain the message signature in a ds:Signature element.

3260 Lines (M058)-(M073) contain the ds:SignedInfo element, which identifies the elements to be signed.

3261 Lines (M063)-(M067) contains a ds:Reference to the SOAP:Body, which is signed in the same manner as
3262 example section 2.3.1.4 above.

3263 Lines (M068)-(M072) contains a ds:Reference to the URI "timestamp", which again is signed in the same
3264 manner as the wsu:Timestamp in section 2.3.1.4 above.

3265 Lines (M075)-(M083) contain the ds:KeyInfo element, which identifies the key that should be used to
3266 verify this ds:Signature element. Lines (M076)-(M082) contain a wsse:SecurityTokenReference, which
3267 contains a wsse:KeyIdentifier that identifies the signing key as being contained in a token of
3268 wsse:ValueType "...wss-saml-token-profile-1.0#SAMLAssertionID", which means a SAML V1.1 Assertion
3269 [[WSS10-SAML11-PROFILE](#)]. Line (M080) contains the saml:AssertionID of the SAML Assertion that
3270 contains the signing key. This SAML Assertion is on lines (M016)-(M056) with the correct
3271 saml:AssertionID on line (M017). Note that the fact that the referenced token (the SAML Assertion) occurs
3272 in the wsse:Security header before this ds:KeyInfo element that uses it in compliance with the sp:Strict
3273 layout policy (P032) and the wsse:KeyIdentifier is an acceptable token reference mechanism as specified
3274 in the policy sp:Wss10 assertion containing a sp:MustSupportRefKeyIdentifier assertion (P041).

3275

3276 2.3.2 WSS 1.1 SAML Token Scenarios

3277 This section contains SamlToken examples that use WS-Security 1.1 SOAP Message Security [WSS11]
3278 and the WS-Security 1.1 SAML Profile [WSS11-SAML1120-PROFILE].

3279 2.3.2.1 (WSS1.1) SAML 2.0 Bearer

3280 This example is based on the Liberty Alliance Identity Web Services Framework (ID-WSF 2.0) Security
3281 Mechanism for the SAML 2.0 Profile for WSS 1.1 [WSS11-LIBERTY-SAML20-PROFILE], which itself is
3282 based on the [WSS11-SAML1120-PROFILE].

3283 In this example, an AsymmetricBinding is used for message protection provided by the Initiator, however,
3284 Recipient trust for the content is based only on the SAML bearer token provided by the Requestor.

3285

```
3286 (P001) <wsp:Policy>
3287 (P002)   <sp:AsymmetricBinding>
3288 (P003)     <wsp:Policy>
3289 (P004)       <sp:InitiatorToken>
3290 (P005)         <wsp:Policy>
3291 (P006)           <sp:X509Token sp:IncludeToken="http://docs.oasis-
3292 open.org/ws-sx/ws-
3293 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3294 (P007)             <wsp:Policy>
3295 (P008)               <sp:WssX509V3Token10/>
3296 (P009)             </wsp:Policy>
3297 (P010)           </sp:X509Token>
3298 (P011)         </wsp:Policy>
3299 (P012)       </sp:InitiatorToken>
3300 (P013)     <sp:RecipientToken>
3301 (P014)       <wsp:Policy>
3302 (P015)         <sp:X509Token sp:IncludeToken="http://docs.oasis-
3303 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
3304 (P016)           <wsp:Policy>
3305 (P017)             <sp:WssX509V3Token10/>
3306 (P018)           </wsp:Policy>
3307 (P019)         </sp:X509Token>
3308 (P020)       </wsp:Policy>
3309 (P021)     </sp:RecipientToken>
3310 (P022)   <sp:AlgorithmSuite>
3311 (P023)     <wsp:Policy>
3312 (P024)       <sp:Basic256/>
3313 (P025)     </wsp:Policy>
3314 (P026)   </sp:AlgorithmSuite>
3315 (P027) <sp:Layout>
3316 (P028)   <wsp:Policy>
3317 (P029)     <sp:Strict/>
3318 (P030)   </wsp:Policy>
3319 (P031) </sp:Layout>
3320 (P032) <sp:IncludeTimestamp/>
3321 (P033) <sp:OnlySignEntireHeadersAndBody/>
3322 (P034) </wsp:Policy>
3323 (P035) </sp:AsymmetricBinding>
3324 (P036) <sp:Wss11>
3325 (P037)   <wsp:Policy>
3326 (P038)     <sp:MustSupportRefKeyIdentifier/>
3327 (P039)     <sp:MustSupportRefIssuerSerial/>
3328 (P040)     <sp:MustSupportRefThumbprint/>
3329 (P041)     <sp:MustSupportRefEncryptedKey/>
3330 (P042)   </wsp:Policy>
3331 (P043) </sp:Wss11>
3332 (P044) <sp:SupportingTokens>
3333 (P045)   <wsp:Policy>
```



```

3334 (P046) <sp:SamIToken sp:IncludeToken="http://docs.oasis-open.org/ws-
3335 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3336 (P047) <wsp:Policy>
3337 (P048) <sp:WssSamIv20Token11/>
3338 (P049) </wsp:Policy>
3339 (P050) </sp:SamIToken>
3340 (P051) </wsp:Policy>
3341 (P052) </sp:SupportingTokens>
3342 (P053) </wsp:Policy>

```

3343 Lines (P001)-(P045) contain the endpoint policy, which contains 3 assertions: an sp:AsymmetricBinding
3344 assertion, an sp:Wss11 assertion, and an sp:SupportingTokens assertion.

3345 Lines (P002)-(P035) contain the sp:AsymmetricBinding assertion, which is identical to the same assertion
3346 that is described in section 2.1.3. Please refer to section 2.1.3 for details.

3347 Lines (P036)-(P043) contain an sp:Wss11 assertion, which contains a set of assertions that specify the
3348 token referencing techniques that are required to be supported (P038)-(P041).

3349 Lines (P044)-(P052) contain an sp:SupportingTokens assertion, which contains an sp:SamIToken
3350 (P046)-(P050), which specifies that it must always be sent to the Recipient. The sp:SamIToken contains
3351 an sp:WssSamIv20Token11 assertion (P048), which means that the SamIToken provided must be a
3352 SAML Version 2.0 Assertion that is submitted in compliance with the [WSS11-SAML1120-PROFILE].

3353 The fact that the sp:SamIToken is contained in an sp:SupportingTokens element indicates to the Initiator
3354 that the SAML Assertion must use the saml2:SubjectConfirmation@Method "bearer" as described above
3355 in introductory section 2.3.

3356 The following is an example request compliant with the above policy:

```

3357
3358 (M001) <?xml version="1.0" encoding="UTF-8"?>
3359 (M002) <s:Envelope xmlns:s=".../soap/envelope/" xmlns:sec="..."
3360 (M003) xmlns:wss="..." xmlns:wsu="..." xmlns:wsa="...addressing"
3361 (M004) xmlns:sb="...liberty:sb" xmlns:pp="...liberty.id-sis-pp"
3362 (M005) xmlns:ds="...xmldsig#" xmlns:xenc="...xmlenc#">
3363 (M006) <s:Header>
3364 (M007) <!-- see Liberty SOAP Binding Spec for reqd, optional hdrs -->
3365 (M008) <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>
3366 (M009) <wsa:To wsu:Id="to">...</wsa:To>
3367 (M010) <wsa:Action wsu:Id="action">...</wsa:Action>
3368 (M011) <wsse:Security mustUnderstand="1">
3369 (M012) <wsu:Timestamp wsu:Id="ts">
3370 (M013) <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
3371 (M014) </wsu:Timestamp>
3372 (M015) <!-- this is the bearer token -->
3373 (M016) <saml2:Assertion xmlns:saml2="...SAML:2.0:assertion"
3374 (M017) Version="2.0" ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
3375 (M018) IssueInstant="2005-04-01T16:58:33.173Z">
3376 (M019) <saml2:Issuer>http://authority.example.com/</saml2:Issuer>
3377 (M020) <!-- signature by the issuer over the assertion -->
3378 (M021) <ds:Signature>...</ds:Signature>
3379 (M022) <saml2:Subject>
3380 (M023) <saml2:EncryptedID>
3381 (M024) <xenc:EncryptedData
3382 (M025) >U2XTCNvRX7B1lNK182nmY00TEk==</xenc:EncryptedData>
3383 (M026) <xenc:EncryptedKey>...</xenc:EncryptedKey>
3384 (M027) </saml2:EncryptedID>
3385 (M028) <saml2:SubjectConfirmation
3386 (M029) Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3387 (M030) </saml2:Subject>
3388 (M031) <!-- audience restriction on assertion can limit scope of
3389 (M032) which entity should consume the info in the assertion. -->
3390 (M033) <saml2:Conditions NotBefore="2005-04-01T16:57:20Z"
3391 (M034) NotOnOrAfter="2005-04-01T21:42:4 3Z">
3392 (M035) <saml2:AudienceRestrictionCondition>

```

```

3393 (M036) <saml2:Audience>http://wsp.example.com</saml2:Audience>
3394 (M037) </saml2:AudienceRestrictionCondition>
3395 (M038) </saml2:Conditions>
3396 (M039) <!-- The AuthnStatement carries info that describes authN
3397 (M040) event of the Subject to an Authentication Authority -->
3398 (M041) <saml2:AuthnStatement AuthnInstant="2005-04-01T16:57:30.000Z"
3399 (M042) SessionIndex="6345789">
3400 (M043) <saml2:AuthnContext>
3401 (M044) <saml2:AuthnContextClassRef
3402 (M045) >...:SAML:2.0:ac:classes:PasswordProtectedTransport
3403 (M046) </saml2:AuthnContextClassRef>
3404 (M047) </saml2:AuthnContext>
3405 (M048) </saml2:AuthnStatement>
3406 (M049) <!-- This AttributeStatement carries an EncryptedAttribute.
3407 (M050) Once this element decrypted with supplied key an <Attribute>
3408 (M051) element bearing endpoint reference can be found, specifying
3409 (M052) resources which invoker may access. Details on element can be
3410 (M053) found in discovery service specification. -->
3411 (M054) <saml2:AttributeStatement>
3412 (M055) <saml2:EncryptedAttribute>
3413 (M056) <xenc:EncryptedData
3414 (M057) Type="http://www.w3.org/2001/04/xmlenc#Element" >
3415 (M058) mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzQxdgcX8
3416 (M059) ... hg6nZ5c0I6L6Gn9A=HCQY
3417 (M060) </xenc:EncryptedData>
3418 (M061) <xenc:EncryptedKey> ... </xenc:EncryptedKey>
3419 (M062) </saml2:EncryptedAttribute>
3420 (M063) </saml2:AttributeStatement>
3421 (M064) </saml2:Assertion>
3422 (M065) <!-- This SecurityTokenReference is used to reference the SAML
3423 (M066) Assertion from a ds:Reference -->
3424 (M067) <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsp="..."
3425 (M068) xmlns:wssell="..." wsp:Id="str1" wssell:TokenType=
3426 (M069) ".../wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
3427 (M070) <wsse:KeyIdentifier ValueType=
3428 (M071) ".../wss/oasis-wss-saml-token-profile-1.1#SAMLID"
3429 (M072) >sxJu9g/vvLG9sAN9bKp/8q0NKU=</wsse:KeyIdentifier>
3430 (M073) </wsse:SecurityTokenReference>
3431 (M074) <ds:Signature>
3432 (M075) <ds:SignedInfo>
3433 (M076) <!-- in general include a ds:Reference for each wsa:header
3434 (M077) added according to SOAP binding, plus timestamp, plus ref to
3435 (M078) assertion to avoid token substitution attacks, plus Body -->
3436 (M079) <ds:Reference URI="#to">...</ds:Reference>
3437 (M080) <ds:Reference URI="#action">...</ds:Reference>
3438 (M081) <ds:Reference URI="#mid">...</ds:Reference>
3439 (M082) <ds:Reference URI="#ts">...</ds:Reference>
3440 (M083) <ds:Reference URI="#Str1">
3441 (M084) <ds:Transform Algorithm="...#STR-Transform">
3442 (M085) <wsse:TransformationParameters>
3443 (M086) <ds:CanonicalizationMethod Algorithm=
3444 (M087) "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
3445 (M088) </wsse:TransformationParameters>
3446 (M089) </ds:Transform>
3447 (M090) </ds:Reference>
3448 (M091) <ds:Reference URI="#MsgBody">...</ds:Reference>
3449 (M092) </ds:SignedInfo>
3450 (M093) ...
3451 (M094) </ds:Signature>
3452 (M095) </wsse:Security>
3453 (M096) </s:Header>
3454 (M097) <s:Body wsp:Id="MsgBody">
3455 (M098) <pp:Modify>
3456 (M099) <!-- this is an ID-SIS-PP Modify message -->

```

```
3457 (M0100)      </pp:Modify>
3458 (M0101)      </s:Body>
3459 (M0102)      </s:Envelope>
```

3460

3461 The sample message above was taken and cosmetically edited from the [\[WSS11-LIBERTY-SAML20-](#)
3462 [PROFILE\]](#).

3463 Lines (M002)-(M0102) contain the SOAP:Envelope, which contains the SOAP:Header (M006)-(M096)
3464 and the SOAP:Body (M097)-(M0101).

3465 The SOAP Header contains some WS-Addressing (wsa:) parameters (M008)-(M010) (not discussed
3466 here) and a wsse:Security header.

3467 The wsse:Security header (M011)-(M095) contains a wsu:Timestamp (M012)-(M014), a saml2:Assertion
3468 (M016)-(M064), a wsse:SecurityTokenReference (M067)-(M073), and a ds:Signature (M074)-(M094).

3469 The wsu:Timestamp (M012)-(M014) is required by the policy sp:IncludeTimestamp assertion (P032).

3470 The saml2:Assertion (M016)-(M064) is required by the policy sp:WssSamlV20Token11 (P048). The
3471 saml2:Assertion is Version 2.0 (M017) and it is signed by the IssuingAuthority (M021). This is the
3472 ds:Signature (M021) of the IssuingAuthority, identified in the saml2:Issuer element (M019), that the
3473 Recipient trusts with respect to recognizing the Requestor and determining whether the request should be
3474 granted. In addition, this saml2:Assertion uses the "bearer" saml2:SubjectConfirmation@Method, as
3475 required by the policy sp:SamlToken in the sp:SupportingTokens element described above
3476 (P044)-(P052).

3477 Note: the saml2:Assertion contains a saml2:EncryptedID (M023)-(M027), which contains the
3478 identity of the Requestor and a saml2:EncryptedAttribute (M062), which contains information
3479 about the Requestor. It is presumed that prior to issuing this request, that the Requestor
3480 contacted the IssuingAuthority (directly or indirectly) and was granted permission to access the
3481 web service that is now the target of this request. As such, the IssuingAuthority has knowledge of
3482 this web service and presumably the public certificate associated with that service and has used
3483 the public key contained in that certificate to encrypt the aforementioned portions of the
3484 saml2:Assertion, which can only be decrypted by the RelyingParty who presumably has entrusted
3485 the private key of the service with the Recipient, in order that the Recipient may decrypt the
3486 necessary data.

3487 However, before the Recipient can evaluate these aspects of the request, the requirements of the policy
3488 sp:AsymmetricBinding (P002)-(P035) must be met in terms of proper "presentation" of the request as
3489 described below.

3490 Lines (M074)-(M094) contain the message signature ds:Signature element, which contains a
3491 ds:SignedInfo that contains ds:Reference elements that cover the wsa: headers (M079)-(M081), the
3492 wsu:Timestamp (M082) (required by the policy sp:IncludeTimestamp assertion (P032), the
3493 saml2:Assertion (M083)-(M090), and the SOAP:Body (M091).

3494 The ds:Reference (M083)-(M090) covering the saml2:Assertion warrants further examination.

3495 The first point to note is that to reference the saml2:Assertion the ds:Reference uses an STR-
3496 Transform (M084) to reference a wsse:SecurityTokenReference (M067)-(M073), which is in
3497 compliance with policy sp:Wss11 sp:MustSupportRefKeyIdentifier assertion (P038).

3498 The second point to note about this ds:Reference is that is covering the saml2:Assertion even
3499 though the policy references the sp:SamlToken in an sp:SupportingTokens element and not an
3500 sp:SignedSupportingTokens element. The reason this is the case is that it is the fact that an
3501 sp:SupportingTokens (P044)-(P052) element is used that tells the Initiator that a SAML bearer
3502 Assertion is required. However, this only means that the SamlToken is not "required" to be
3503 signed. It is the Initiator's choice whether to sign it or not, and it is generally good practice to do
3504 so in order to prevent a token substitution attack.

3505

3506

3507 2.3.2.2 (WSS1.1) SAML2.0 Sender Vouches over SSL

3508 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#) Scenario
3509 #2].

3510 Similar to 2.3.1.2 except SAML token is of version 2.0.

3511

3512

```
3513 (P001) <wsp:Policy>  
3514 (P002)   <sp:TransportBinding>  
3515 (P003)   <wsp:Policy>  
3516 (P004)   <sp:TransportToken>  
3517 (P005)   <wsp:Policy>  
3518 (P006)   <sp:HttpsToken>  
3519 (P007)   <wsp:Policy>  
3520 (P008)   <sp:RequireClientCertificate>  
3521 (P009)   </wsp:Policy>  
3522 (P010)   </sp:HttpsToken>  
3523 (P011)   </wsp:Policy>  
3524 (P012)   </sp:TransportToken>  
3525 (P013)   <sp:AlgorithmSuite>  
3526 (P014)   <wsp:Policy>  
3527 (P015)   <sp:Basic256 />  
3528 (P016)   </wsp:Policy>  
3529 (P017)   </sp:AlgorithmSuite>  
3530 (P018)   <sp:Layout>  
3531 (P019)   <wsp:Policy>  
3532 (P020)   <sp:Strict />  
3533 (P021)   </wsp:Policy>  
3534 (P022)   </sp:Layout>  
3535 (P023)   <sp:IncludeTimestamp />  
3536 (P024)   </wsp:Policy>  
3537 (P025)   </sp:TransportBinding>  
3538 (P026)   <sp:SignedSupportingTokens>  
3539 (P027)   <wsp:Policy>  
3540 (P028)   <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
3541         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3542 (P029)   <wsp:Policy>  
3543 (P030)   <sp:WssSamlV20Token11/>  
3544 (P031)   </wsp:Policy>  
3545 (P032)   </sp:SamlToken>  
3546 (P033)   </wsp:Policy>  
3547 (P034)   </sp:SignedSupportingTokens>  
3548 (P035)   </wsp:Policy>
```

3549 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by
3550 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and
3551 sp:SignedSupportingTokens (P026).

3552 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
3553 by a secure transport protocol such as SSL or TLS.

3554 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
3555 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
3556 transport token using the Basic256 algorithm suite (P015).

3557 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of
3558 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section
3559 4.3.1].

3560 The requirements for the sp:Layout assertion (P018)-(P022) should be automatically met (or may be
3561 considered moot) since there are no cryptographic tokens required to be present in the WS-Security
3562 header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout
3563 would need to be considered.)

3564 A timestamp (P023) is required to be included in an acceptable message.

3565

3566 Here is an example request:

3567

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
3568 (M002) <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3569 (M003) xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3570 (M004) xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3571 (M005) xmlns:wsu=".../oasis-200401-wsswssecurity-utility-1.0.xsd"
3572 (M006) xmlns:wssse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3573 (M007) xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3574 (M008) <S11:Header>
3575 (M009) <wsse:Security S11:mustUnderstand="1">
3576 (M010) <wsu:Timestamp>
3577 (M011) <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3578 (M012) </wsu:Timestamp>
3579 (M013) <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3580 (M014) IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3581 (M015) <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3582 (M016) <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3583 (M017) NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3584 (M018) <saml2:Subject>
3585 (M019) <saml2:NameID Format=
3586 (M020) "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3587 (M021) >uid=joe,ou=people,ou=saml demo,o=example.com</saml2:NameID>
3588 (M022) <saml2:SubjectConfirmation Method=
3589 (M023) "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches" />
3590 (M024) </saml2:Subject>
3591 (M025) <saml2:AttributeStatement>
3592 (M026) <saml2:Attribute>...</saml2:Attribute>
3593 (M027) <saml2:Attribute>...</saml2:Attribute>
3594 (M028) </saml2:AttributeStatement>
3595 (M029) </saml2:Assertion>
3596 (M030) </wsse:Security>
3597 (M031) </S11:Header>
3598 (M032) <S11:Body>
3599 (M033) <Ping xmlns="http://xmlsoap.org/Ping">
3600 (M034) <text>EchoString</text>
3601 (M035) </Ping>
3602 (M036) </S11:Body>
3603 (M037) </S11:Envelope>
```

3605 Lines (M002)-(M037) contain the SOAP:Envelope, which contains the SOAP:Header (M003)-(M031) and
3606 the SOAP:Body (M032)-(M036).

3607 Lines (M009)-(M030) contain the wsse:Security header, which, in conjunction with the client
3608 certificate (P008), is the primary basis for trust in this example.

3609 Lines (M010)-(M012) contain the wsu:Timestamp, which meets the sp:IncludeTimestamp assertion policy
3610 requirement (P023).

3611 Lines (M013)-(M029) contain the saml2:Assertion, which is required to be Version 2.0 (M014) and to be
3612 used within the [WSS11-SAML1120-PROFILE], because of the policy sp:SamIToken assertion
3613 (P028)-(P032), which contains the sp:WssSamIV20Token11 (P030).

3614 Lines (M022)-(M023) indicate that the SAML Assertion uses the saml2:Method sender-vouches for the
3615 purposes of saml2:SubjectConfirmation, which is required by the fact that the sp:SamIToken appears in
3616 an sp:SignedSupportingTokens assertion (P026)-(P034) in conjunction with the client certificate required
3617 by the sp:RequireClientCertificate assertion (P008) within the sp:TransportBinding as described in section
3618 2.3 above. In addition, the Recipient should be able to correlate the saml2:Issuer (M015) as being
3619 properly associated with the client certificate received from the HTTPS (SSL) connection.

3620

3621

3622 2.3.2.3 (WSS1.1) SAML2.0 HoK over SSL

3623 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)
3624 Scenario #5].

3625 Similar to 2.3.1.3 except SAML token is of version 2.0.

3626 Initiator adds a SAML Assertion (hk) to the SOAP Security Header. In this policy the sp:TransportBinding
3627 requires a Client Certificate AND the sp:SamlToken is in an sp:SignedEndorsingSupportingTokens
3628 element. A SAML holder-of-key Assertion meets these requirements because it is “virtually signed” by the
3629 message signature as a result of the SSL client certificate authentication procedure as described in
3630 section 2.3. Furthermore, the SAML hk Assertion in this case is a “virtually endorsing”, because the key
3631 identified in the holder-of-key saml2:SubjectConfirmationData is also the client certificate, which is
3632 virtually endorsing its own signature, under the authority of the IssuingAuthority who has signed the
3633 SAML hk Assertion.

3634 As a result, the Initiator may be considered to be authorized by the saml2:Issuer of the hk SAML
3635 Assertion to bind message content to the Subject of the Assertion. If the Client Certificate matches the
3636 certificate identified in the hk Assertion, the Initiator may be regarded as executing SAML hk responsibility
3637 of binding the Requestor, who would be the Subject of the hk Assertion, to the content of the message.

3638 (Note: the same considerations described in section 2.3.1.4 with respect to whether the Subject
3639 of the Assertion and the Subject of the Client Certificate are the same entity determining whether
3640 the Client Certificate is attesting for the Assertion Subject or whether the Client Certificate is
3641 authenticating as the Assertion Subject apply here.)

3642 In this scenario, the IssuingAuthority is the saml2:Issuer(signer) of the hk SAML Assertion. The Requestor
3643 is the Subject of the Assertion and the Initiator is authorized by the IssuingAuthority to bind the Assertion
3644 to the message using the ClientCertificate identified in the SAML Assertion, which may also be
3645 considered to be virtually signing the wsu:Timestamp of the message. Optionally, a separate Signature
3646 may be used to sign the wsu:Timestamp, which the Recipient would also be required to verify was signed
3647 by the client certificate in this example.

3648

```
3649 (P001) <wsp:Policy>  
3650 (P002)   <sp:TransportBinding>  
3651 (P003)   <wsp:Policy>  
3652 (P004)     <sp:TransportToken>  
3653 (P005)       <wsp:Policy>  
3654 (P006)         <sp:HttpsToken>  
3655 (P007)           <wsp:Policy>  
3656 (P008)             <sp:RequireClientCertificate>  
3657 (P009)               </wsp:Policy>  
3658 (P010)                 </sp:HttpsToken>  
3659 (P011)                   </wsp:Policy>  
3660 (P012)                     </sp:TransportToken>  
3661 (P013)                       <sp:AlgorithmSuite>  
3662 (P014)                         <wsp:Policy>  
3663 (P015)                           <sp:Basic256 />  
3664 (P016)                             </wsp:Policy>  
3665 (P017)                               </sp:AlgorithmSuite>  
3666 (P018)                                 <sp:Layout>  
3667 (P019)                                   <wsp:Policy>  
3668 (P020)                                     <sp:Strict />  
3669 (P021)                                       </wsp:Policy>  
3670 (P022)                                         </sp:Layout>  
3671 (P023)                                           <sp:IncludeTimestamp />  
3672 (P024)                                             </wsp:Policy>  
3673 (P025)                                               </sp:TransportBinding>  
3674 (P026)                                                 <sp:SignedEndorsingSupportingTokens>  
3675 (P027)                                                   <wsp:Policy>  
3676 (P028)                                                     <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
3677 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3678 (P029)                                                       <wsp:Policy>
```

```

3679      (P030)          <sp:WssSamlV20Token11/>
3680      (P031)          </wsp:Policy>
3681      (P032)          </sp:SamlToken>
3682      (P033)          </wsp:Policy>
3683      (P034)          </sp:SignedEndorsingSupportingTokens>
3684      (P035)          </wsp:Policy>

```

3685

3686 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by
3687 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and
3688 sp:EndorsingSupportingTokens (P026).

3689 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected
3690 by a secure transport protocol such as SSL or TLS.

3691 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by
3692 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the
3693 transport token using the Basic256 algorithm suite (P015).

3694 In addition, because this is SAML holder-of-key, a client certificate is required (P008) as the basis of trust
3695 for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section 4.3.1]. In
3696 the holder-of-key case, there will be an additional certificate required in the trust chain, which is the
3697 certificate used to sign the SAML hk Assertion, which will be contained or referenced in the Assertion.

3698 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)
3699 since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a
3700 signature element was included to cover the wsse:Timestamp, then the layout would need to be
3701 considered.)

3702 A timestamp (P023) is required to be included in an acceptable message.

3703 Lines (P026)-(P034) contain an sp:SignedEndorsingSupportingTokens element, which means that the
3704 contained supporting token (the SAML hk Assertion) references a key that will be “signing” (endorsing)
3705 the message signature. The token itself may also be considered to be “signed”, because it is contained in
3706 the message sent over the SSL link. In the case of sp:TransportBinding, there may be no actual
3707 “message signature”, however, when a client certificate is used, the service can be assured that the
3708 connection was set up by the client and that the SSL link guarantees the integrity of the data that is sent
3709 on the link by the client, while it is on the link and when it is received from the link by using the key
3710 referenced by the token. However, it does not guarantee the integrity after the data is received (i.e. after it
3711 is received there is no way to tell whether any changes have been made to it since it has been received).
3712 In any event, within this context a Signed Endorsing Supporting Token can be used to tell the Recipient
3713 that the Issuer of the token is making claims related to the holder of the private key that is referenced by
3714 the token, which in this case would be the private key associated with the client certificate used to set up
3715 the SSL link, as described further below.

3716 Lines (P028)-(P032) contain an sp:SamlToken, which must always be sent to the Recipient.

3717 Line (P030) specifies that the token is of type WssSamlV20Token11, which means that the Endorsing
3718 Supporting Token must be a SAML Version 2.0 token and it must be sent using WS-Security 1.1 using
3719 the [[WSS11-SAML1120-PROFILE](#)]. Note that because the SamlToken is contained in an
3720 sp:EndorsingSupportingTokens element, that it implicitly must be a holder-of-key token as described in
3721 section 2.3 above.

3722 Here is an example request taken from the WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)
3723 Scenario #5] containing only minor cosmetic modifications and corrections.

3724

```

3725      (M001)      <?xml version="1.0" encoding="utf-8" ?>
3726      (M002)      <S11:Envelope
3727      (M003)          xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3728      (M004)          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3729      (M005)          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3730      (M006)          xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3731      (M007)          xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3732      (M008)          xmlns:wssell=".../oasis-wss-wssecurity-secext-1.1.xsd"

```

```

3733 (M009)      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3734 (M010)      xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3735 (M011)      <S11:Header>
3736 (M012)      <wsse:Security S11:mustUnderstand="1">
3737 (M013)      <wsu:Timestamp>
3738 (M014)      <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3739 (M015)      </wsu:Timestamp>
3740 (M016)      <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3741 (M017)      IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3742 (M018)      <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3743 (M019)      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3744 (M020)      <ds:SignedInfo>
3745 (M021)      <ds:CanonicalizationMethod
3746 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3747 (M023)      <ds:SignatureMethod
3748 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3749 (M025)      <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3750 (M026)      <ds:Transforms>
3751 (M027)      <ds:Transform Algorithm=
3752 (M028)      "http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
3753 (M029)      <ds:Transform Algorithm=
3754 (M030)      "http://www.w3.org/2001/10/xml-exc-c14n#">
3755 (M031)      <InclusiveNamespaces
3756 (M032)      PrefixList="#default saml ds xs xsi"
3757 (M033)      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3758 (M034)      </ds:Transform>
3759 (M035)      </ds:Transforms>
3760 (M036)      <ds:DigestMethod Algorithm=
3761 (M037)      "http://www.w3.org/2000/09/xmldsig#sha1" />
3762 (M038)      <ds:DigestValue
3763 (M039)      >Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3764 (M040)      </ds:Reference>
3765 (M041)      </ds:SignedInfo>
3766 (M042)      <ds:SignatureValue>
3767 (M043)      hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3768 (M044)      7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmtp3TD
3769 (M045)      MwuL/cBUj20tBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3770 (M046)      </ds:SignatureValue>
3771 (M047)      <ds:KeyInfo>
3772 (M048)      <ds:X509Data>
3773 (M049)      <ds:X509Certificate>
3774 (M050)      MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxkCzAJBgNVBAYTA1VT
3775 (M051)      ...
3776 (M052)      8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
3777 (M053)      </ds:X509Certificate>
3778 (M054)      </ds:X509Data>
3779 (M055)      </ds:KeyInfo>
3780 (M056)      </ds:Signature>
3781 (M057)      <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3782 (M058)      NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3783 (M059)      <saml2:Subject>
3784 (M060)      <saml2:NameID
3785 (M061)      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3786 (M062)      >uid=joe,ou=people,ou=saml-demo,o=example.com</saml2:NameID>
3787 (M063)      <saml2:SubjectConfirmation
3788 (M064)      Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" />
3789 (M065)      <saml2:SubjectConfirmationData
3790 (M066)      xsi:type="saml2:KeyInfoConfirmationDataType">
3791 (M067)      <ds:KeyInfo>
3792 (M068)      <ds:X509Data>
3793 (M069)      <ds:X509IssuerName>
3794 (M070)      C=ZA, ST=Western Cape, L=Cape Town,
3795 (M071)      O=Thawte Consulting cc,
3796 (M072)      OU=Certification Services Division,

```



```

3797      (M073)          CN=Thawte Server CA/Email=server-certs@thawte.com
3798      (M074)          </ds:X509IssuerName>
3799      (M075)          <X509SerialNumber>12345678</X509SerialNumber>
3800      (M076)          </ds:X509Data>
3801      (M077)          </ds:KeyInfo>
3802      (M078)          </saml2:SubjectConfirmationData>
3803      (M079)          </saml2:Subject>
3804      (M080)          <saml2:AttributeStatement>
3805      (M081)          <saml2:Attribute Name="MemberLevel">
3806      (M082)          <saml2:AttributeValue>gold</saml2:AttributeValue>
3807      (M083)          </saml2:Attribute>
3808      (M084)          <saml2:Attribute Name="E-mail">
3809      (M085)          <saml2:AttributeValue
3810      (M086)          >joe@yahoo.com</saml2:AttributeValue>
3811      (M087)          </saml2:Attribute>
3812      (M088)          </saml2:AttributeStatement>
3813      (M089)          </saml2:Assertion>
3814      (M090)          </wsse:Security>
3815      (M091)          </S11:Header>
3816      (M092)          <S11:Body wsu:Id="MsgBody">
3817      (M093)          <ReportRequest>
3818      (M094)          <TickerSymbol>SUNW</TickerSymbol>
3819      (M095)          </ReportRequest>
3820      (M096)          </S11:Body>
3821      (M097)          </S11:Envelope>

```

3822 Lines (M002)-(M097) contain the SOAP:Envelope, which contains the SOAP:Header (M011)-(M091) and
3823 the SOAP:Body (M092)-(M096).

3824 Lines (M012)-(M090) contain the wsse:Security header, which contains a wsu:Timestamp (M013)-(M015)
3825 and a saml2:Assertion (M016)-(M089).

3826 The wsu:Timestamp (M013)-(M015) may be considered to be virtually signed by the client certificate as
3827 explained above and in section 2.3.

3828 The saml2:Assertion was issued by an IssuingAuthority identified in the saml2:Issuer element (M018).

3829 The saml2:Issuer has used the private key of its enterprise certificate to produce the ds:Signature
3830 element (M019)-(M056) that is an enveloped-signature (M028) that covers its parent XML element, the
3831 saml2:Assertion (M016)-(M089). The ds:KeyInfo (M047)-(M055) within this ds:Signature contains an
3832 actual copy of the Issuer's enterprise certificate, that the Recipient can verify for authenticity to establish
3833 that this message is in conformance with any agreement the RelyingParty may have with the
3834 IssuingAuthority. The details of this verification are outside the scope of this document, however they
3835 involve the structures and systems the RelyingParty has in place recognize and verify certificates and
3836 signatures that are associated with business arrangements with the holders of the private keys
3837 of those certificates.

3838 The saml2:Subject of the saml2:Assertion is contained in lines (M059)-(M079). Within the saml2:Subject
3839 is the saml2:NameID (M060)-(M062), which contains the official name of the Subject of this Assertion and
3840 this entity may be considered to the Requestor associated with this request.

3841 The saml2:SubjectConfirmation (M063)-(M064) has Method "holder-of-key" which implies that there is a
3842 saml2:SubjectConfirmationData element (M065)-(M078) which will contain information that identifies a
3843 signing key that the Initiator will use to bind this saml2:Assertion to a message that is associated with the
3844 Requestor. In this context, the Initiator is acting as "attesting entity" with respect to the Subject as defined
3845 in [SAML20], which means that the Initiator is authorized by the IssuingAuthority to present
3846 saml2:Assertions pertaining to saml2:Subjects to Recipients/RelyingParties. In this example there is a
3847 ds:KeyInfo (M067)-(M077) that identifies a specific certificate (ds:X509IssuerName (M069)-(M074) and
3848 ds:SerialNumber (M075)) that the Initiator must prove possession of the associated private key to verify
3849 this message. In this policy that private key is the one associated with the client certificate that the Initiator
3850 is required to use (P008).

3851 The saml2:AttributeStatement (M080)-(M088) contains some information about the Subject that is
3852 officially being provided in this saml2:Assertion by the IssuingAuthority that may have some significance
3853 to the Relying Party in terms of determining whether access is granted for this request.

3855 **2.3.2.4 (WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign,**
3856 **Encrypt**

3857 Here the message and SAML Assertion are signed using a key derived from the ephemeral key K. The
3858 ephemeral key is encrypted using the Recipient's public key for the request input message and the same
3859 shared ephemeral key, K, is encrypted using the Initiators public key for the response output message.

3860 Alternatively, derived keys can be used for each of signing and encryption operations.

3861 In this scenario the Authority is the Initiator who signs the message with the generated key. In order to
3862 establish trust in the generated key, the Initiator must sign the message signature with a second signature
3863 using an X509 certificate, which is indicated as the EndorsingSupportingToken. This X509 certificate
3864 establishes the Initiator as the SAML Authority.

3865 (Note: there are instructive similarities and differences between this example and example 2.3.1.4, where
3866 the difference is that: here the binding is symmetric and the WSS1.1 is used, whereas in 2.3.1.4 the
3867 binding is asymmetric and WSS1.0 is used. One particular item is that in 2.3.1.4 an
3868 EndorsingSupportingToken is not needed because in 2.3.1.4 the asymmetric binding uses X.509
3869 certificates, which may be inherently trusted as opposed to the ephemeral key, K, used here.)

3870

```
3871 (P001) <wsp:Policy wsu:Id="WSS11SamlWithCertificates_policy">
3872 (P002)   <wsp:ExactlyOne>
3873 (P003)     <wsp>All>
3874 (P004)       <sp:SymmetricBinding>
3875 (P005)         <wsp:Policy>
3876 (P006)           <sp:ProtectionToken>
3877 (P007)             <wsp:Policy>
3878 (P008)               <sp:X509Token sp:IncludeToken=
3879 "http://docs.oasis-open.org/ws-sx/ws-
3880 securitypolicy/200702/IncludeToken/Never">
3881 (P009)                 <wsp:Policy>
3882 (P010)                   <sp:RequireThumbprintReference/>
3883 (P011)                   <sp:RequireDerivedKeys wsp:Optional="true"/>
3884 (P012)                   <sp:WssX509V3Token10/>
3885 (P013)                 </wsp:Policy>
3886 (P014)               </sp:X509Token>
3887 (P015)             </wsp:Policy>
3888 (P016)           </sp:ProtectionToken>
3889 (P017)         <sp:AlgorithmSuite>
3890 (P018)           <wsp:Policy>
3891 (P019)             <sp:Basic256/>
3892 (P020)           </wsp:Policy>
3893 (P021)         </sp:AlgorithmSuite>
3894 (P022)       <sp:Layout>
3895 (P023)         <wsp:Policy>
3896 (P024)           <sp:Strict/>
3897 (P025)         </wsp:Policy>
3898 (P026)       </sp:Layout>
3899 (P027)     <sp:IncludeTimestamp/>
3900 (P028)   <sp:OnlySignEntireHeadersAndBody/>
3901 (P029) </wsp:Policy>
3902 (P030) </sp:SymmetricBinding>
3903 (P031) <sp:SignedSupportingTokens>
3904 (P032)   <wsp:Policy>
3905 (P033)     <sp:SamlToken sp:IncludeToken=
3906 "http://docs.oasis-open.org/ws-sx/ws-
3907 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3908 (P034)       <wsp:Policy>
3909 (P035)         <sp:WssSamlV11Token11/>
3910 (P036)       </wsp:Policy>
3911 (P037)     </sp:SamlToken>
3912 (P038)   </wsp:Policy>
3913 (P039) </sp:SignedSupportingTokens>
```

```

3914 (P040) <sp:EndorsingSupportingTokens>
3915 (P041) <wsp:Policy>
3916 (P042) <sp:X509Token sp:IncludeToken="AlwaysToRecipient ">
3917 (P043) <wsp:Policy>
3918 (P044) <sp:WssX509V3Token11/>
3919 (P045) </wsp:Policy>
3920 (P046) </sp:X509Token>
3921 (P047) </wsp:Policy>
3922 (P048) </sp:EndorsingSupportingTokens>
3923 (P049) <sp:Wss11>
3924 (P050) <wsp:Policy>
3925 (P051) <sp:MustSupportRefKeyIdentifier/>
3926 (P052) <sp:MustSupportRefIssuerSerial/>
3927 (P053) <sp:MustSupportRefThumbprint/>
3928 (P054) <sp:MustSupportRefEncryptedKey/>
3929 (P055) </wsp:Policy>
3930 (P056) </sp:Wss11>
3931 (P057) </wsp:All>
3932 (P058) </wsp:ExactlyOne>
3933 (P059) </wsp:Policy>
3934 (P060)
3935 (P061) <wsp:Policy wsu:Id="SamlForCertificates_input_policy">
3936 (P062) <wsp:ExactlyOne>
3937 (P063) <wsp:All>
3938 (P064) <sp:SignedParts>
3939 (P065) <sp:Body/>
3940 (P066) </sp:SignedParts>
3941 (P067) <sp:EncryptedParts>
3942 (P068) <sp:Body/>
3943 (P069) </sp:EncryptedParts>
3944 (P070) </wsp:All>
3945 (P071) </wsp:ExactlyOne>
3946 (P072) </wsp:Policy>
3947 (P073)
3948 (P074) <wsp:Policy wsu:Id="SamlForCertificate_output_policy">
3949 (P075) <wsp:ExactlyOne>
3950 (P076) <wsp:All>
3951 (P077) <sp:SignedParts>
3952 (P078) <sp:Body/>
3953 (P079) </sp:SignedParts>
3954 (P080) <sp:EncryptedParts>
3955 (P081) <sp:Body/>
3956 (P082) </sp:EncryptedParts>
3957 (P083) </wsp:All>
3958 (P084) </wsp:ExactlyOne>
3959 (P085) </wsp:Policy>

```

3960 Lines (P001)-(P059) contain the policy that requires all the contained assertions to be complied with by
3961 the Initiator. In this case there are 4 assertions: sp:SymmetricBinding (P004)-(P030),
3962 sp:SignedSupportingTokens (P031)-(P039), sp:EndorsingSupportingTokens (P040)-(P048), and
3963 sp:Wss11 (P049)-(P056).

3964 The sp:SymmetricBinding assertion (P004) contains an sp:ProtectionToken (P006)-(P016), which
3965 indicates that both the Initiator and Recipient must use each other's public key respectively associated
3966 with the X509Token (P008)-(P014) associated with the respective message receiver to encrypt the
3967 shared ephemeral symmetric key as explained at the beginning of this section. The messages may either
3968 be encrypted and signed using keys derived from the ephemeral key as indicated by the
3969 sp:RequireDerivedKeys assertion (P011) or the encryption and signing can be done using the ephemeral
3970 key itself without deriving keys, because the RequireDerivedKey assertion is an optional requirement as
3971 indicated by the wsp:Optional attribute on line (P011).

3972 The sp:SignedSupportingTokens assertion (P031) contains an sp:SamlToken assertion (P033)-(P037),
3973 which indicates that a signed SAML Assertion must always be included in the Initiator's request to the
3974 Recipient (AlwaysToRecipient (P033)). This SAML Assertion must be included in the WS-Security header

3975 and referenced and signed as described in the WS-Security 1.1 Profile for SAML [[WSS11-SAML1120-](#)
3976 [PROFILE](#)] as indicated by the sp:WssSamlV11Token11 assertion (P036), which indicates the SAML 1.1
3977 option in that profile (as opposed to the SAML 2.0 option, which would have been indicated by
3978 sp:WssSamlV20Token11).

3979 The sp:EndorsingSupportingToken assertion (P040) is needed because there are no guarantees that the
3980 ephemeral key, K, is still being used by the Initiator with whom the Recipient would have collaborated
3981 originally to establish the ephemeral key as a shared secret. The purpose of the endorsing token is to
3982 sign the signature made by the ephemeral key, using the Initiator's private key, which will explicitly
3983 guarantee the content of this particular Initiator request. The endorsing token in this policy is an
3984 sp:X509Token (P042)-(P046), which, in particular, is an sp:WssX509V3Token11, which must be used in
3985 accordance with the WS-Security 1.1 Profile for X509 Tokens [[WSS11-X509-PROFILE](#)]. (Note: it may be
3986 the case that this X509 certificate is the same X509 certificate referred to in the ProtectionToken
3987 assertion (P012). However, it is important to keep in mind that line (P012) only indicates that the Initiator's
3988 token will be used by the Recipient to protect the ephemeral key for the symmetric binding. Therefore, the
3989 fact that the token is identified in line (P012) as an sp:X509V3Token10 is not directly related to the fact
3990 that the same key may be used for the additional purpose of explicitly signing the request message).

3991 The sp:Wss11 assertion (P049-P056) indicates that WS-Security 1.1 constructs are accepted. (Note also
3992 that eitherWssX509V3Token10 or WssX509V3Token11 may be used with the Wss11 since both WS-
3993 Security 1.0 and WS-Security 1.1 Profiles are supported by WS-Security 1.1)

3994 There are also 2 Policys, one each for the input message and the output message, each of which
3995 contains an assertion indicating the message SOAP Body must be signed (P064)-(P066) for the input
3996 message and (P077)-(P079) for the output message, and each contains an assertion that the message
3997 SOAP Body must be encrypted (P067)-(P069) for the input message and (P080)-(P082) for the output
3998 message.

3999 The following is a sample request that is compliant with this policy.

```
4000 (M001) <?xml version="1.0" encoding="utf-8" ?>
4001 (M002) <S12:Envelope
4002 (M003)   xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
4003 (M004)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4004 (M005)   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4005 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4006 (M007)   wss-wssecurity-secext-1.0.xsd"
4007 (M007)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4008 (M008)   wssecurity-utility-1.0.xsd"
4009 (M008)   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4010 (M009)   xmlns:xenc="..."
4011 (M010)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
4012 (M011) <S12:Header>
4013 (M012) <wsse:Security S12:mustUnderstand="1">
4014 (M013) <wsu:Timestamp wsu:Id="timestamp">
4015 (M014) <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
4016 (M015) </wsu:Timestamp>
4017 (M016) <xenc:EncryptedKey Id="encKey" >
4018 (M017) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
4019 (M018) <ds:DigestMethod Algorithm="...#sha1"/>
4020 (M019) </xenc:EncryptionMethod>
4021 (M020) <ds:KeyInfo >
4022 (M021) <wsse:SecurityTokenReference >
4023 (M022) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
4024 (M022) Value="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
4025 (M023) </wsse:SecurityTokenReference>
4026 (M024) </ds:KeyInfo>
4027 (M025) <xenc:CipherData>
4028 (M026) <xenc:CipherValue>TE...=</xenc:CipherValue>
4029 (M027) </xenc:CipherData>
4030 (M028) </xenc:EncryptedKey>
4031 (M029) <n1:ReferenceList xmlns:n1=".../xmlenc#">
4032 (M030) <n1:DataReference URI="#encBody"/>
4033 (M031) </n1:ReferenceList>
```

```

4034 (M032) <saml:Assertion
4035 (M033)   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
4036 (M034)   IssueInstant="2003-04-17T00:46:02Z"
4037 (M035)   Issuer="www.opensaml.org"
4038 (M036)   MajorVersion="1"
4039 (M037)   MinorVersion="1">
4040 (M038)   <saml:Conditions
4041 (M039)     NotBefore="2002-06-19T16:53:33.173Z"
4042 (M040)     NotOnOrAfter="2002-06-19T17:08:33.173Z" />
4043 (M041)   <saml:AttributeStatement>
4044 (M042)     <saml:Subject>
4045 (M043)       <saml:NameIdentifier
4046 (M044)         NameQualifier="www.example.com"
4047 (M045)         Format="">
4048 (M046)         uid=joe,ou=people,ou=saml-demo,o=example.com
4049 (M047)       </saml:NameIdentifier>
4050 (M048)     <saml:SubjectConfirmation>
4051 (M049)       <saml:ConfirmationMethod>
4052 (M050)         urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
4053 (M051)       </saml:ConfirmationMethod>
4054 (M052)     </saml:SubjectConfirmation>
4055 (M053)   </saml:Subject>
4056 (M054)   <saml:Attribute>
4057 (M055)     ...
4058 (M056)   </saml:Attribute>
4059 (M057)   ...
4060 (M058) </saml:AttributeStatement>
4061 (M059) </saml:Assertion>
4062 (M060) <wsse:SecurityTokenReference wsu:id="STR1">
4063 (M061)   <wsse:KeyIdentifier wsu:id="..."
4064 (M062)     ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
4065 (M063)     profile-1.0#SAMLAssertionID">
4066 (M064)     _a75adf55-01d7-40cc-929f-dbd8372ebdfc
4067 (M064)   </wsse:KeyIdentifier>
4068 (M065) </wsse:SecurityTokenReference>
4069 (M066) <wsse:BinarySecurityToken
4070 (M067)   wsu:Id="attesterCert"
4071 (M068)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4072 (M069)   wss-x509-token-profile-1.0#X509v3"
4073 (M069)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
4074 (M070)   200401-wss-soap-message-security-1.0#Base64Binary">
4075 (M070)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
4076 (M071) </wsse:BinarySecurityToken>
4077 (M072) <ds:Signature wsu:Id="message-signature">
4078 (M073)   <ds:SignedInfo>
4079 (M074)     <ds:CanonicalizationMethod
4080 (M075)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4081 (M076)     <ds:SignatureMethod
4082 (M077)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4083 (M078)     <ds:Reference URI="#STR1">
4084 (M079)       <ds:Transforms>
4085 (M080)         <ds:Transform
4086 (M081)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
4087 (M082)           200401-wss-soap-message-security-1.0#STR-Transform">
4088 (M082)         <wsse:TransformationParameters>
4089 (M083)           <ds:CanonicalizationMethod
4090 (M084)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4091 (M085)         </wsse:TransformationParameters>
4092 (M086)       </ds:Transform>
4093 (M087)     </ds:Transforms>
4094 (M088)     <ds:DigestMethod
4095 (M089)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
4096 (M090)     <ds:DigestValue>...</ds:DigestValue>
4097 (M091)   </ds:Reference>

```

```

4098 (M092) <ds:Reference URI="#MsgBody" >
4099 (M093) <ds:DigestMethod
4100 (M094) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
4101 (M095) <ds:DigestValue>...</ds:DigestValue>
4102 (M096) </ds:Reference>
4103 (M097) <ds:Reference URI="#timestamp" >
4104 (M098) <ds:DigestMethod
4105 (M099) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
4106 (M100) <ds:DigestValue>...</ds:DigestValue>
4107 (M101) </ds:Reference>
4108 (M102) </ds:SignedInfo>
4109 (M103) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
4110 (M104) <ds:KeyInfo>
4111 (M105) <wsse:SecurityTokenReference wsu:id="STR2" >
4112 (M106) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey" />
4113 (M107) </wsse:SecurityTokenReference>
4114 (M108) </ds:KeyInfo>
4115 (M109) </ds:Signature>
4116 (M110) <ds:Signature wsu:Id="endorsing-signature" >
4117 (M111) <ds:SignedInfo>
4118 (M112) ...
4119 (M113) <ds:Reference URI="#message-signature" >
4120 (M114) ...
4121 (M115) </ds:Reference>
4122 (M116) </ds:SignedInfo>
4123 (M117) <ds:SignatureValue>Bu ...=</ds:SignatureValue>
4124 (M118) <ds:KeyInfo>
4125 (M119) <wsse:SecurityTokenReference >
4126 (M120) <wsse:Reference URI="#attesterCert" ValueType="...#X509v3" />
4127 (M121) </wsse:SecurityTokenReference>
4128 (M122) </ds:KeyInfo>
4129 (M123) </ds:Signature>
4130 (M124) </wsse:Security>
4131 (M125) </S12:Header>
4132 (M126) <S12:Body wsu:Id="MsgBody" >
4133 (M127) <xenc:EncryptedData Id="encBody" Type="...#Content"
4134 MimeType="text/xml" Encoding="UTF-8" >
4135 (M128) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc" />
4136 (M129) <ds:KeyInfo >
4137 (M130) <wsse:SecurityTokenReference>
4138 (M131) <wsse:Reference URI="#enckey" ValueType=".../EncryptedKey" />
4139 (M132) </wsse:SecurityTokenReference>
4140 (M133) </ds:KeyInfo>
4141 (M134) <xenc:CipherData>
4142 (M135) <xenc:CipherValue>70...=</xenc:CipherValue>
4143 (M136) </xenc:CipherData>
4144 (M137) </xenc:EncryptedData>
4145 (M138) </S12:Body>
4146 (M139) </S12:Envelope>

```

4147

4148 The message above contains a request compliant with the policy described in (P001)-(P072), which
4149 includes the endpoint policy and the input policy.

4150 Lines (M016)-(M028) contain an xenc:EncryptedKey element that contains an Initiator-generated
4151 symmetric signing and encryption key, K, that can be used to decrypt the xenc:EncryptedData contained
4152 in the SOAP S12:Body (M0126) as indicated by the wsse:SecurityTokenReference (M0130)-(M0132) that
4153 uses a direct reference to the Id of the xenc:EncryptedKey, "encKey" on lines (M0131) and (M016).

4154 The xenc:EncryptedKey contains a dsig KeyInfo (M020)-(M024) reference to the Thumbprint of the
4155 Recipient's public X509 certificate (M022). This complies with the policy (P068) that requires the sp:Body
4156 to of the input message to be encrypted. It also complies with the policy (P013) to protect using
4157 encryption based on X509 token and to refer to it by Thumbprint (P014).

4158 Lines (M029)-(M031) contain an xenc:ReferenceList referring to the xenc:EncryptedData in the S12:Body
4159 (M0127)-(M0137).

4160 Lines (M032)-(M059) contain the saml:Assertion as a SignedSupportingToken as required by the
4161 endpoint policy (P035). A saml:Assertion used as a SignedSupportingToken uses the “sender-vouches”
4162 ConfirmationMethod (M049)-(M051) as described in the introductory section 2.3.

4163 Lines (M060)-(M065) contain a WS-Security wsse:SecurityTokenReference that has a KeyIdentifier of
4164 ValueType “...1.0#SAMLAssertionID”, which indicates a SAML 1.1 Assertion as described in the WS-
4165 Security 1.1 [[SAML1120_TOKEN_PROFILE](#)].

4166 Lines (M066)-(M071) contain a wsse:BinarySecurityToken that contains the Initiator’s X509 certificate for
4167 use as an EndorsingSupportingToken as required by line (P042) of the sp:EndorsingSupportingTokens
4168 assertion (P040)-(P048).

4169 Lines (M072)-(M0109) contain the message signature. The dsig Signature covers the following:

- 4170 ➤ The SAML Assertion using the ds:Reference (M078)-(M091), which uses the WS-Security STR-
4171 Transform technique to refer to the SAML Assertion indirectly through the
4172 wsse:SecurityTokenReference (M060) described above. This signature is required by the
4173 sp:SignedSupportingTokens assertion (P031)-(P039).
- 4174 ➤ The message sp:Body (M0126)-(M0138) using the ds:Reference (M092)-(M096) as required by
4175 the input message policy (P065).
- 4176 ➤ The message wsu:Timestamp (M013)-(M015) using the ds:Reference (M097)-(M0101) as
4177 required by the endpoint policy (P027).

4178 The key used to sign the message signature is referenced in the dsig KeyInfo (M0104)-(M0108), which
4179 contains a SecurityTokenReference with a direct URI reference to the xenc:EncryptedKey, “encKey”,
4180 which contains the Initiator-generated signing and encryption key, K, as described above.

4181 Lines (M0110)-(M0123) contain the endorsing signature. The dsig endorsing Signature covers the
4182 following:

- 4183 ➤ The message Signature (M072)-(M0109) using the ds:Reference (M0113)-(M0115), which is a
4184 direct URI reference to the Id “message-signature” on line (M072).

4185 This signature is required by the policy EndorsingSupportingTokens assertion (P040)-(P048) to be an
4186 X509 certificate (P044). The dsig KeyInfo (M0118)-(M0122) contains a WS-Security
4187 SecurityTokenReference with a direct URI to the Initiator’s X509 certificate on line (M066)-(M071) with Id
4188 = “attesterCert”.

4189 Lines (M0127)-(M0137) contain the xenc:EncryptedData, which contains the SOAP S12:Body message
4190 payload that is encrypted using the encryption key, K, contained in the xenc:EncryptedKey CipherValue
4191 (M026), which can only be decrypted using the Recipient’s X509 certificate referred to by
4192 ThumbprintSHA1 on line (M022).

4193 **2.3.2.5 (WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt**

4194 This scenario is based on WS-SX Interop Scenarios Phase 2 (October 31, 2006) [[WSSX-WSTR-WSSC-](#)
4195 [INTEROP](#)] Scenario 5 (Client and STS: Mutual Certificate WSS1.1 (section 3.5 of interop ref), Client and
4196 Service: Issued SAML 1.1 Token for Certificate WSS1.1 (section 4.3 of interop ref)).

4197 In this scenario, the service specifies that the client must obtain an IssuedToken from a designated
4198 Security Token Service (STS), which must be a SAML 1.1 Assertion. The Assertion contains an
4199 ephemeral key K2 in an EncryptedKey element encrypted using service’s certificate. The client also
4200 obtains the same ephemeral key K2 from the RequestedProofToken returned by the STS. The body of
4201 the message from the client to the service is signed using DKT1(K2), encrypted using DKT2(K2), and
4202 endorsed using DKT3(K2), which are keys the client derives from K2 using the algorithm specified in
4203 section 7 of [[WS-SecureConversation](#)]. The response from the service is also signed using derived keys.
4204 In a simpler alternative, ephemeral key K itself could be used for message protection.

4205 Note: In this scenario, in terms of Figure 1 [[Figure01](#)], the STS (Issuing Authority) is the
4206 Issuer of the SAML 1.1 holder-of-key Assertion that contains the ephemeral symmetric
4207 key K2. The service (combined Recipient/RelyingParty) can trust the client (combined

4208 Initiator/Requestor) that uses the ephemeral symmetric key K2 obtained from the
4209 RequestedProofToken, because the same key gets delivered to the service in the SAML
4210 1.1 holder-of-key Assertion, which is signed by the trusted Authority.

4211 This scenario is a 2-step sequence from a WS-SP perspective. These 2 steps will be described at a high
4212 level first in order to explain the context for the policies and messages that follow.

4213 **In step 1** the following takes place:

- 4214 • the client accesses the service's WS-SP policy (P001 -> P116 below), and determines that it
4215 needs to use an IssuedToken from a Security Token Service (STS), specified in the policy.
4216 This IssuedToken will serve as the basis of trust by the service. Effectively, the service only trusts
4217 the client because the client is able to obtain the IssuedToken from the STS.

4218 To complete step 1, the client will then do the following:

- 4219 • the client will access the STS and process the STS policy (PSTS-001 -> PSTS-098 below)
- 4220 • based on the STS policy, the client will send a request to the STS for an IssuedToken (MSTS-001
4221 -> MSTS-0231 below)
- 4222 • the STS will send a response to the client containing the IssuedToken, which in this example is a
4223 SAML 1.1 holder-of-key Assertion (RSTS-001 -> RSTS-0263 below)

4224 **In step 2** the following takes place:

- 4225 • because the client now has the IssuedToken it is now able fulfill the requirements of the service's
4226 WS-SP policy (P001-P116 below) that it accessed in step 1
- 4227 • based on the service's policy the clients sends a request to the service (M001-M0229 below)
- 4228 • the service will send a response to the client containing the requested resource data (R001-
4229 R0153 below)

4230 As an aid to understanding the security properties of the policies and messages in this example, the
4231 following is a list of identifiers used in the text descriptions to reference the cryptographic keys that are
4232 called for in the policies and used in the messages in this example:

- 4233 ➤ **X509T1**: Client's (Requestor/Initiator) X509 certificate used for authentication to the STS.
- 4234 ➤ **X509T2**: STS' (Issuing Authority) X509 certificate used to encrypt keys sent to STS, used to
4235 authenticate STS to Service.
- 4236 ➤ **X509T3**: Service's (Recipient/RelyingParty/ValidatingAuthority) X509 certificate used to encrypt
4237 keys sent to Service.
- 4238 ➤ **K1**: Client-generated ephemeral symmetric key for crypto communication to the STS.
- 4239 ➤ **K2**: Client-generated ephemeral symmetric key for crypto communication between the Client and
4240 the Service.
- 4241 ➤ **K3**: STS-generated proof key for use by Client to authenticate with Service via saml:Assertion.
- 4242 ➤ **DKT1(K2)**: Client-generated derived key used for signing requests to the Service.
- 4243 ➤ **DKT2(K2)**: Client-generated derived key used for encrypting requests to the Service.
- 4244 ➤ **DKT3(K3)**: Client-generated derived key used for endorsing signed requests to the Service.
- 4245 ➤ **DKT4(K2)**: Service-generated derived key used for encrypting responses to the Client.
- 4246 ➤ **DKT5(K2)**: Service-generated derived key used for signing responses to the Client.

4247

4248 Here is the WS-SP Policy that the Service presents to a Client:

4249

```
4250 (P001) <wsp:Policy wsu:Id="Service5-Policy"  
4251 (P002)  
4252 (P003) xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4253 (P004) xmlns:sp"... "  
4254 (P005)
```



```

4255 (P006)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4256 wssecurity-utility-1.0.xsd"
4257 (P007)      >
4258 (P008)      <wsp:ExactlyOne>
4259 (P009)      <wsp:All>
4260 (P010)      <sp:SymmetricBinding>
4261 (P011)      <wsp:Policy>
4262 (P012)      <sp:ProtectionToken>
4263 (P013)      <wsp:Policy>
4264 (P014)      <sp:X509Token IncludeToken=
4265 (P015)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4266 /Never">
4267 (P016)      <wsp:Policy>
4268 (P017)      <sp:RequireDerivedKeys/>
4269 (P018)      <sp:RequireThumbprintReference/>
4270 (P019)      <sp:WssX509V3Token10/>
4271 (P020)      </wsp:Policy>
4272 (P021)      </sp:X509Token>
4273 (P022)      </wsp:Policy>
4274 (P023)      </sp:ProtectionToken>
4275 (P024)      <sp:AlgorithmSuite>
4276 (P025)      <wsp:Policy>
4277 (P026)      <sp:Basic256/>
4278 (P027)      </wsp:Policy>
4279 (P028)      </sp:AlgorithmSuite>
4280 (P029)      <sp:Layout>
4281 (P030)      <wsp:Policy>
4282 (P031)      <sp:Strict/>
4283 (P032)      </wsp:Policy>
4284 (P033)      </sp:Layout>
4285 (P034)      <sp:IncludeTimestamp/>
4286 (P035)      <sp:OnlySignEntireHeadersAndBody/>
4287 (P036)      </wsp:Policy>
4288 (P037)      </sp:SymmetricBinding>
4289 (P038)      <sp:EndorsingSupportingTokens>
4290 (P039)      <wsp:Policy>
4291 (P040)      <sp:IssuedToken IncludeToken=
4292 (P041)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4293 /AlwaysToRecipient">
4294 (P042)      <sp:Issuer>
4295 (P043)      <a:Address>http://example.com/STS</a:Address>
4296 (P044)      </sp:Issuer>
4297 (P045)      <sp:RequestSecurityTokenTemplate>
4298 (P046)      <t:TokenType
4299 (P047)      >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4300 1.1#SAMLV1.1</t:TokenType>
4301 (P048)      <t:KeyType
4302 (P049)      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:KeyTy
4303 pe>
4304 (P050)      <t:KeySize>256</t:KeySize>
4305 (P051)      <t:CanonicalizationAlgorithm
4306 (P052)      >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorithm>
4307 (P053)      <t:EncryptionAlgorithm
4308 (P054)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4309 (P055)      <t:EncryptWith
4310 (P056)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4311 (P057)      <t:SignWith
4312 (P058)      >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4313 (P059)      </sp:RequestSecurityTokenTemplate>
4314 (P060)      <wsp:Policy>
4315 (P061)      <sp:RequireDerivedKeys/>
4316 (P062)      <sp:RequireInternalReference/>
4317 (P063)      </wsp:Policy>
4318 (P064)      </sp:IssuedToken>

```

```

4319 (P065)         </wsp:Policy>
4320 (P066)         </sp:EndorsingSupportingTokens>
4321 (P067)         <sp:Wss11>
4322 (P068)         <wsp:Policy>
4323 (P069)         <sp:MustSupportRefKeyIdentifier/>
4324 (P070)         <sp:MustSupportRefIssuerSerial/>
4325 (P071)         <sp:MustSupportRefThumbprint/>
4326 (P072)         <sp:MustSupportRefEncryptedKey/>
4327 (P073)         <sp:RequireSignatureConfirmation/>
4328 (P074)         </wsp:Policy>
4329 (P075)         </sp:Wss11>
4330 (P076)         <sp:Trust13>
4331 (P077)         <wsp:Policy>
4332 (P078)         <sp:MustSupportIssuedTokens/>
4333 (P079)         <sp:RequireClientEntropy/>
4334 (P080)         <sp:RequireServerEntropy/>
4335 (P081)         </wsp:Policy>
4336 (P082)         </sp:Trust13>
4337 (P083)         </wsp:All>
4338 (P084)         </wsp:ExactlyOne>
4339 (P085)         </wsp:Policy>
4340
4341 (P086)         <wsp:Policy wsu:Id="InOut-Policy"
4342 (P087)
4343 (P088)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4344 (P089)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4345 (P090)
4346 (P091)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4347 (P092)         wssecurity-utility-1.0.xsd">
4348 (P093)         <wsp:ExactlyOne>
4349 (P094)         <wsp:All>
4350 (P095)         <sp:SignedParts>
4351 (P096)         <sp:Body/>
4352 (P097)         <sp:Header Name="To"
4353 (P098)         Namespace="http://www.w3.org/2005/08/addressing"/>
4354 (P099)         <sp:Header Name="From"
4355 (P100)         Namespace="http://www.w3.org/2005/08/addressing"/>
4356 (P101)         <sp:Header Name="FaultTo"
4357 (P102)         Namespace="http://www.w3.org/2005/08/addressing"/>
4358 (P103)         <sp:Header Name="ReplyTo"
4359 (P104)         Namespace="http://www.w3.org/2005/08/addressing"/>
4360 (P105)         <sp:Header Name="MessageID"
4361 (P106)         Namespace="http://www.w3.org/2005/08/addressing"/>
4362 (P107)         <sp:Header Name="RelatesTo"
4363 (P108)         Namespace="http://www.w3.org/2005/08/addressing"/>
4364 (P109)         <sp:Header Name="Action"
4365 (P110)         Namespace="http://www.w3.org/2005/08/addressing"/>
4366 (P111)         </sp:SignedParts>
4367 (P112)         <sp:EncryptedParts>
4368 (P113)         <sp:Body/>
4369 (P114)         </sp:EncryptedParts>
4370 (P115)         </wsp:All>
4371 (P116)         </wsp:ExactlyOne>
4372 (P117)         </wsp:Policy>

```

4373

4374 When a Client encounters the above Service Policy, it determines that it must obtain an IssuedToken
4375 from the designated Issuer. After obtaining the IssuedToken, the client may send the request in
4376 accordance with the rest of the policy. Details of the Service Policy follow. The STS Policy its details
4377 follow after the Service Policy.

4378 Lines (P001)-(P085) above contain the Service Endpoint wsp:Policy, which contains a wsp:All assertion
4379 that requires all 4 of its contained assertions to be complied with: sp:SymmetricBinding (P010)-(P037),
4380 sp:EndorsingSupportingTokens (P038)-(P066), sp:Wss11 (P067)-(P075), and sp:Trust13 (P076)-(P082).

4381 Lines (P010)-(P037) contain the **sp:SymmetricBinding assertion** that requires that derived keys (DKT1,
4382 DKT2, DKT3) be used to protect the message (P017) and that the ephemeral key (K2) used to derive
4383 these keys be encrypted using the service's X509 certificate (X509T3) as specified by the sp:X509Token
4384 assertion (P014)-(P021), which also indicates that the X509 token, itself should not be sent (P015), but
4385 that a Thumbprint reference (P018) to it be sent. Finally, the sp:SymmetricBinding specifies that the
4386 sp:Basic256 sp:AlgorithmSuite be used (P024)-(P028), that sp:Strict sp:Layout be used (P029)-(P033),
4387 that an sp:Timestamp be included (P034), and that only the complete message Body and Headers be
4388 signed (P035), where the Headers part means that only direct child elements of the WS-Security SOAP
4389 header element be signed.

4390 Lines (P038)-(P066) contain the **sp:EndorsingSupportingTokens assertion** that an sp:IssuedToken
4391 (P040)-(P064) be used to sign the message signature and that the IssuedToken must be included with
4392 the request (P040)-(P041). Lines (P042)-(P044) specify the address of the SecurityTokenService (STS)
4393 from which the IssuedToken must be obtained. Lines (P045)-(P059) contain an
4394 sp:RequestSecurityTokenTemplate (P046)-(P058) which contains explicit WS-Trust elements that the
4395 client should directly copy to a t:SecondaryParameters element to include with the WS-Trust
4396 t:RequestSecurityToken to the STS to obtain the sp:IssuedToken. Of particular interest here is that the
4397 t:TokenType (P046)-(P047) requested is a SAML 1.1 Assertion, which will be used to contain the
4398 ephemeral symmetric key (K2) (P048)-(P049). K2 will be used for communication between the Client and
4399 the Service. K2 will be encrypted by the STS using the Service's X509 certificate (X509T3). The Client is
4400 also informed by the IssuedToken assertion that the IssuedToken may only be referenced internally
4401 within the message (P062) and that the ephemeral key (K2) associated with the IssuedToken be used by
4402 the Client to derive the keys (DKT1(K2), DKT2(K2), DKT3(K2)) used in the Client's request to the Service.

4403 Lines (P067)-(P075) contain the **sp:Wss11 assertion** that indicates that WS-Security 1.1 will be used,
4404 which includes Wss11-only features such as Thumbprint (P073), EncryptedKey (P074), and
4405 SignatureConfirmation (P075).

4406 Lines (P076)-(P082) contain the **sp:Trust13 assertion** that indicates the Client should expect to use WS-
4407 Trust 1.3 ([WSTRUST](#)) to obtain the IssuedToken from the STS.

4408 Lines (P086)-(P0116) contain the **operation input and output policies** that the client should use to
4409 determine what parts of the messages are to be signed (P094)-(P0110) and encrypted (P0111)-(P0113).

4410

4411 Here is the WS-SP Policy that the STS presents to a Client:

```
4412 (PSTS-001) <wsp:Policy wsu:Id="STS5-Policy"
4413 (PSTS-002)
4414 (PSTS-003)   xmlns:wsp="http://www.w3.org/ns/ws-policy"
4415 (PSTS-004)   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4416 (PSTS-005)
4417 (PSTS-006)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4418 wss-wssecurity-utility-1.0.xsd"
4419 (PSTS-007)   >
4420 (PSTS-008)   <wsp:ExactlyOne>
4421 (PSTS-009)     <wsp>All>
4422 (PSTS-010)       <sp:SymmetricBinding>
4423 (PSTS-011)         <wsp:Policy>
4424 (PSTS-012)           <sp:ProtectionToken>
4425 (PSTS-013)             <wsp:Policy>
4426 (PSTS-014)               <sp:X509Token IncludeToken=
4427 (PSTS-015)                 "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4428 ken/Never">
4429 (PSTS-016)                   <wsp:Policy>
4430 (PSTS-017)                     <sp:RequireThumbprintReference/>
4431 (PSTS-018)                     <sp:WssX509V3Token10/>
4432 (PSTS-019)                   </wsp:Policy>
4433 (PSTS-020)                 </sp:X509Token>
4434 (PSTS-021)               </wsp:Policy>
4435 (PSTS-022)             </sp:ProtectionToken>
4436 (PSTS-023)           <sp:AlgorithmSuite>
4437 (PSTS-024)             <wsp:Policy>
4438 (PSTS-025)               <sp:Basic256/>
```

```

4439 (PSTS-026)         </wsp:Policy>
4440 (PSTS-027)         </sp:AlgorithmSuite>
4441 (PSTS-028)         <sp:Layout>
4442 (PSTS-029)         <wsp:Policy>
4443 (PSTS-030)         <sp:Strict/>
4444 (PSTS-031)         </wsp:Policy>
4445 (PSTS-032)         </sp:Layout>
4446 (PSTS-033)         <sp:IncludeTimestamp/>
4447 (PSTS-034)         <sp:OnlySignEntireHeadersAndBody/>
4448 (PSTS-035)         </wsp:Policy>
4449 (PSTS-036)         </sp:SymmetricBinding>
4450 (PSTS-037)         <sp:EndorsingSupportingTokens>
4451 (PSTS-038)         <wsp:Policy>
4452 (PSTS-039)         <sp:X509Token IncludeToken=
4453 (PSTS-040)         "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4454 (PSTS-040)         ken/AlwaysToRecipient">
4455 (PSTS-041)         <wsp:Policy>
4456 (PSTS-042)         <sp:RequireThumbprintReference/>
4457 (PSTS-043)         <sp:WssX509V3Token10/>
4458 (PSTS-044)         </wsp:Policy>
4459 (PSTS-045)         </sp:X509Token>
4460 (PSTS-046)         </wsp:Policy>
4461 (PSTS-047)         </sp:EndorsingSupportingTokens>
4462 (PSTS-048)         <sp:Wss11>
4463 (PSTS-049)         <wsp:Policy>
4464 (PSTS-050)         <sp:MustSupportRefKeyIdentifier/>
4465 (PSTS-051)         <sp:MustSupportRefIssuerSerial/>
4466 (PSTS-052)         <sp:MustSupportRefThumbprint/>
4467 (PSTS-053)         <sp:MustSupportRefEncryptedKey/>
4468 (PSTS-054)         <sp:RequireSignatureConfirmation/>
4469 (PSTS-055)         </wsp:Policy>
4470 (PSTS-056)         </sp:Wss11>
4471 (PSTS-057)         <sp:Trust13>
4472 (PSTS-058)         <wsp:Policy>
4473 (PSTS-059)         <sp:MustSupportIssuedTokens/>
4474 (PSTS-060)         <sp:RequireClientEntropy/>
4475 (PSTS-061)         <sp:RequireServerEntropy/>
4476 (PSTS-062)         </wsp:Policy>
4477 (PSTS-063)         </sp:Trust13>
4478 (PSTS-064)         </wsp:All>
4479 (PSTS-065)         </wsp:ExactlyOne>
4480 (PSTS-066)         </wsp:Policy>
4481
4482 (PSTS-067)         <wsp:Policy wsu:Id="InOut-Policy"
4483 (PSTS-068)
4484 (PSTS-069)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4485 (PSTS-070)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4486 (PSTS-071)
4487 (PSTS-072)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4488 (PSTS-072)         wss-wssecurity-utility-1.0.xsd"
4489 (PSTS-073)         >
4490 (PSTS-074)         <wsp:ExactlyOne>
4491 (PSTS-075)         <wsp:All>
4492 (PSTS-076)         <sp:SignedParts>
4493 (PSTS-077)         <sp:Body/>
4494 (PSTS-078)         <sp:Header Name="To"
4495 (PSTS-079)         Namespace="http://www.w3.org/2005/08/addressing"/>
4496 (PSTS-080)         <sp:Header Name="From"
4497 (PSTS-081)         Namespace="http://www.w3.org/2005/08/addressing"/>
4498 (PSTS-082)         <sp:Header Name="FaultTo"
4499 (PSTS-083)         Namespace="http://www.w3.org/2005/08/addressing"/>
4500 (PSTS-084)         <sp:Header Name="ReplyTo"
4501 (PSTS-085)         Namespace="http://www.w3.org/2005/08/addressing"/>
4502 (PSTS-086)         <sp:Header Name="MessageID"

```

```

4503 (PSTS-087)      Namespace="http://www.w3.org/2005/08/addressing"/>
4504 (PSTS-088)      <sp:Header Name="RelatesTo"
4505 (PSTS-089)      Namespace="http://www.w3.org/2005/08/addressing"/>
4506 (PSTS-090)      <sp:Header Name="Action"
4507 (PSTS-091)      Namespace="http://www.w3.org/2005/08/addressing"/>
4508 (PSTS-092)      </sp:SignedParts>
4509 (PSTS-093)      <sp:EncryptedParts>
4510 (PSTS-094)      <sp:Body/>
4511 (PSTS-095)      </sp:EncryptedParts>
4512 (PSTS-096)      </wsp:All>
4513 (PSTS-097)      </wsp:ExactlyOne>
4514 (PSTS-098)      </wsp:Policy>

```

4515 Above is the STS Policy that the Client will encounter when obtaining the IssuedToken required by the
4516 Service Policy. This policy is quite similar in detail to the Service Policy and therefore only the differences
4517 that are noteworthy will be discussed in the details below.

4518 Similar to the Service Policy, the STS endpoint Policy contains 4 assertions to be complied with:
4519 sp:SymmetricBinding (PSTS-010)-(PSTS-036), sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047),
4520 sp:Wss11 (PSTS-048)-(PSTS-056), and sp:Trust13 (PSTS-057)-(PSTS-063).

4521 Lines (PSTS-010)-(PSTS-036) contain the **sp:SymmetricBinding assertion**, where the main difference
4522 from the previous policy is that here the sp:ProtectionToken is an sp:X509Token that will be used to
4523 encrypt the ephemeral client-generated key (K1) that will be used for Client-STS communication. Derived
4524 keys will not be required in this communication.

4525 Lines (PSTS-037)-(PSTS-047) contain the **sp:EndorsingSupportingTokens assertion**, which in this
4526 case contains an sp:X509Token assertion (PSTS-039)-(PSTS-045) that requires the Client to include
4527 (PSTS-040) its X509 certificate, which must be used to sign the message signature. The STS uses this
4528 mechanism to authenticate the Client.

4529 The **sp:Wss11 assertion** (PSTS-048)-(PSTS-056), **sp:Trust13 assertion** (PSTS-057)-(PSTS-063) and
4530 the **operation input and output policies** (PSTS-067)-(PSTS-098) are the same as those described for
4531 the Service policy above (P067)-(P0116).

4532

4533 Below are included sample messages that comply with the above policies. The messages are presented
4534 in the same order that they would be used in a real scenario and therefore the Client-STS request
4535 (MSTS-001)-(MSTS-0231) and response (RSTS-001)-(RSTS-0263) are presented first, which are then
4536 followed by the Client-Service request (M001-M229) and response (R001)-(R153).

4537 Here is an example Client request to the STS:

```

4538 (MSTS-001)      <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
4539 (MSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
4540 (MSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
4541 (MSTS-004)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4542 (MSTS-005)      wssecurity-secext-1.0.xsd"
4543 (MSTS-005)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4544 (MSTS-005)      wssecurity-utility-1.0.xsd" >
4545 (MSTS-006)      <s:Header>
4546 (MSTS-007)      <a:Action s:mustUnderstand="1" u:Id="_3">
4547 (MSTS-008)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
4548 (MSTS-009)      </a:Action>
4549 (MSTS-010)      <a:MessageID u:Id="_4">
4550 (MSTS-011)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4551 (MSTS-012)      </a:MessageID>
4552 (MSTS-013)      <a:ReplyTo u:Id="_5">
4553 (MSTS-014)      <a:Address>
4554 (MSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4555 (MSTS-016)      </a:Address>
4556 (MSTS-017)      </a:ReplyTo>
4557 (MSTS-018)      <a:To s:mustUnderstand="1" u:Id="_6">
4558 (MSTS-019)      http://server.example.com/STS/Scenarios5-6
4559 (MSTS-020)      </a:To>
4560 (MSTS-021)      <o:Security s:mustUnderstand="1">

```

```

4561 (MSTS-022) <u:Timestamp
4562 (MSTS-023)   u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4563 (MSTS-024)   <u:Created>2005-10-25T00:47:36.144Z</u:Created>
4564 (MSTS-025)   <u:Expires>2005-10-25T00:52:36.144Z</u:Expires>
4565 (MSTS-026)   </u:Timestamp>
4566 (MSTS-027)   <e:EncryptedKey
4567 (MSTS-028)     Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-9">
4568 (MSTS-029)     <e:EncryptionMethod Algorithm=
4569 (MSTS-030)       "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
4570 (MSTS-031)     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
4571 (MSTS-032)       <o:SecurityTokenReference>
4572 (MSTS-033)         <o:KeyIdentifier ValueType=
4573 (MSTS-034)         "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4574 (MSTS-035)         1.1.xsd#ThumbprintSHA1">
4575 (MSTS-036)           W+rgYBmLmVEG//scD7Vo8Kq5G7I=
4576 (MSTS-037)         </o:KeyIdentifier>
4577 (MSTS-038)       </o:SecurityTokenReference>
4578 (MSTS-039)     </KeyInfo>
4579 (MSTS-040)     <e:CipherData>
4580 (MSTS-041)       <e:CipherValue>
4581 (MSTS-042)         <!--base64 encoded cipher-->
4582 (MSTS-043)       </e:CipherValue>
4583 (MSTS-044)     </e:CipherData>
4584 (MSTS-045)     <e:ReferenceList>
4585 (MSTS-046)       <e:DataReference URI="#_2"/>
4586 (MSTS-047)     </e:ReferenceList>
4587 (MSTS-048)   </e:EncryptedKey>
4588 (MSTS-049)   <o:BinarySecurityToken
4589 (MSTS-050)     u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"
4590 (MSTS-051)     ValueType=
4591 (MSTS-052)     "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4592 (MSTS-053)     profile-1.0#X509v3"
4593 (MSTS-054)     EncodingType=
4594 (MSTS-055)     "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
4595 (MSTS-056)     message-security-1.0#Base64Binary">
4596 (MSTS-057)   MIIDDDCCafSgAwIBAgIQM6YEF7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4w
4597 (MSTS-058)   DAYDVQKDAVPQVNJUzEeMBwGAlUEAwVT0FTSVMgSW50ZXJvcCBUZXN0IENBMB4XDTA1
4598 (MSTS-059)   MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVowQjEOMAwGAlUECgwFT0FTSVMxIDAeBgNV
4599 (MSTS-060)   BAsMF09BU01TIEludGVyY3AgVGVzdCBDZXJ0MQ4wDAYDVQQDDAVBbG1jZTCBnzANBQkq
4600 (MSTS-061)   hkiG9w0BAQEFAAOBjQAwYkCgYEAoqi99By1VYo0aHrkKcNT4DkIgpL/SgahbeKdGhrb
4601 (MSTS-062)   u3K2XG7arfd9tqIBIKMfrx4Gp90NJa85AV1yiNsEyvq+mUnMpNcKnLXL0jkTmMcqDYbb
4602 (MSTS-063)   kehJlXPnaWLzve+mW0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz+EsZCAwEA
4603 (MSTS-064)   AaOBkzCBkDAJBgnVHRMEAjaAMDMGAlUdHwQsMCowKKImhiRodHRWoI8vaW50ZXJvcC5i
4604 (MSTS-065)   YnRlc3QubmV0L2Nybc9jYS5jcmwwDgYDVR0PAQH/BAQDAgSwMB0GAlUdDgQWBQK410T
4605 (MSTS-066)   UHZlQV3V2QtllNDm+PoxiDafBgNVHSMEGDAWgBTAnSj8wes1or3WqqqgHbPnWkkPdZAN
4606 (MSTS-067)   BgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW+6yrLXyU1P2xJbEkohXHI5OWwKWleOb9h1kh
4607 (MSTS-068)   WntUalfcFOJAgUyH30TTPhldzx1+vK2LPzhoUFKYHE1IyQvokBN2JjF064BquKCKnZhl
4608 (MSTS-069)   dLRPxBghfktDxQgdf5rCK/wh3xVsZCNTfuMnmlAM6lOAg8QduDah3WFZpEA0s2nwQaCNQ
4609 (MSTS-070)   TNmjJC8tav1CBR6+E5FAMwPXP7pJxn9Fw9OXRYqbRA4v2y7YpbGkG2GI9UvOHw6SGvff4
4610 (MSTS-071)   FRStHMMO35YbpikGsLix3vAsXWwi4rwfVOYzQK00FPNi9RMCUdSH06m9uLWckiCxjos0
4611 (MSTS-072)   FQODZE9l4ATGy9s9hNVwryOJTW==
4612 (MSTS-073)   </o:BinarySecurityToken>
4613 (MSTS-074)   <Signature Id="_0" xmlns="http://www.w3.org/2000/09/xmldsig#">
4614 (MSTS-075)     <SignedInfo>
4615 (MSTS-076)       <CanonicalizationMethod Algorithm=
4616 (MSTS-077)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4617 (MSTS-078)       <SignatureMethod Algorithm=
4618 (MSTS-079)         "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
4619 (MSTS-080)       <Reference URI="#_1">
4620 (MSTS-081)         <Transforms>
4621 (MSTS-082)           <Transform Algorithm=
4622 (MSTS-083)             "http://www.w3.org/2001/10/xml-exc-c14n#" />
4623 (MSTS-084)         </Transforms>
4624 (MSTS-085)       </Reference>

```

```

4625 (MSTS-083) <DigestMethod Algorithm=
4626 (MSTS-084) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4627 (MSTS-085) <DigestValue>VX1fCPwCzVsSc1hZf0BSbCgW2hM=</DigestValue>
4628 (MSTS-086) </Reference>
4629 (MSTS-087) <Reference URI="#_3">
4630 (MSTS-088) <Transforms>
4631 (MSTS-089) <Transform Algorithm=
4632 (MSTS-090) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4633 (MSTS-091) </Transforms>
4634 (MSTS-092) <DigestMethod Algorithm=
4635 (MSTS-093) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4636 (MSTS-094) <DigestValue>FwiFAUuqNDo9SDkk5A28Mg7Pa8Q=</DigestValue>
4637 (MSTS-095) </Reference>
4638 (MSTS-096) <Reference URI="#_4">
4639 (MSTS-097) <Transforms>
4640 (MSTS-098) <Transform Algorithm=
4641 (MSTS-099) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4642 (MSTS-100) </Transforms>
4643 (MSTS-101) <DigestMethod Algorithm=
4644 (MSTS-102) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4645 (MSTS-103) <DigestValue>oM59PsOTpMrDdOcwXYQzjvU10xw=</DigestValue>
4646 (MSTS-104) </Reference>
4647 (MSTS-105) <Reference URI="#_5">
4648 (MSTS-106) <Transforms>
4649 (MSTS-107) <Transform Algorithm=
4650 (MSTS-108) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4651 (MSTS-109) </Transforms>
4652 (MSTS-110) <DigestMethod Algorithm=
4653 (MSTS-111) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4654 (MSTS-112) <DigestValue>KIK3vklFN1QmMdQkplq2azfzrzg=</DigestValue>
4655 (MSTS-113) </Reference>
4656 (MSTS-114) <Reference URI="#_6">
4657 (MSTS-115) <Transforms>
4658 (MSTS-116) <Transform Algorithm=
4659 (MSTS-117) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4660 (MSTS-118) </Transforms>
4661 (MSTS-119) <DigestMethod Algorithm=
4662 (MSTS-120) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4663 (MSTS-121) <DigestValue>RJEe3hrcyCD6PzFJo6fyut6biVg=</DigestValue>
4664 (MSTS-122) </Reference>
4665 (MSTS-123) <Reference
4666 (MSTS-124) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4667 (MSTS-125) <Transforms>
4668 (MSTS-126) <Transform Algorithm=
4669 (MSTS-127) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4670 (MSTS-128) </Transforms>
4671 (MSTS-129) <DigestMethod Algorithm=
4672 (MSTS-130) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4673 (MSTS-131) <DigestValue>zQdN5XpejfqXn0Wko0m5lZYiasE=</DigestValue>
4674 (MSTS-132) </Reference>
4675 (MSTS-133) </SignedInfo>
4676 (MSTS-134) <SignatureValue
4677 (MSTS-135) >iHGJ+Xv2VZTjM1Rc7AQJrwLY/aM=</SignatureValue>
4678 (MSTS-136) <KeyInfo>
4679 (MSTS-137) <o:SecurityTokenReference>
4680 (MSTS-138) <o:Reference
4681 (MSTS-139) ValueType=
4682 (MSTS-140) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4683 1.1.xsd#EncryptedKey"
4684 (MSTS-141) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-9"/>
4685 (MSTS-142) </o:SecurityTokenReference>
4686 (MSTS-143) </KeyInfo>
4687 (MSTS-144) </Signature>
4688 (MSTS-145) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

```

```

4689 (MSTS-0146) <SignedInfo>
4690 (MSTS-0147)   <CanonicalizationMethod Algorithm=
4691 (MSTS-0148)     "http://www.w3.org/2001/10/xml-exc-c14n#" />
4692 (MSTS-0149)   <SignatureMethod Algorithm=
4693 (MSTS-0150)     "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4694 (MSTS-0151)   <Reference URI="#_0">
4695 (MSTS-0152)     <Transforms>
4696 (MSTS-0153)       <Transform Algorithm=
4697 (MSTS-0154)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4698 (MSTS-0155)     </Transforms>
4699 (MSTS-0156)     <DigestMethod Algorithm=
4700 (MSTS-0157)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4701 (MSTS-0158)     <DigestValue>UZKtShk8q6iu9WR5uQZp04iAitg=</DigestValue>
4702 (MSTS-0159)     </Reference>
4703 (MSTS-0160)   </SignedInfo>
4704 (MSTS-0161)   <SignatureValue>
4705 (MSTS-0162) Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWlcmG3xGLZYcc8tbmCtZFuQz9tnW35Lmst6vI
4706 (MSTS-0163) RefuPA7ewRLYORAOjff92SxMbeVTLrIxQbIQNw0bs4SBSLfAo14=
4707 (MSTS-0164)   </SignatureValue>
4708 (MSTS-0165)   <KeyInfo>
4709 (MSTS-0166)     <o:SecurityTokenReference>
4710 (MSTS-0167)       <o:Reference
4711 (MSTS-0168)         ValueType=
4712 (MSTS-0169)       "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4713 (MSTS-0170)         profile-1.0#X509v3"
4714 (MSTS-0171)         URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-6" />
4715 (MSTS-0172)       </o:SecurityTokenReference>
4716 (MSTS-0173)     </KeyInfo>
4717 (MSTS-0174)   </Signature>
4718 (MSTS-0175) </o:Security>
4719 (MSTS-0176) </s:Header>
4720 (MSTS-0177) <s:Body u:Id="_1">
4721 (MSTS-0178)   <e:EncryptedData
4722 (MSTS-0179)     Id="_2"
4723 (MSTS-0180)     Type="http://www.w3.org/2001/04/xmlenc#Content">
4724 (MSTS-0181)     <e:EncryptionMethod Algorithm=
4725 (MSTS-0182)       "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
4726 (MSTS-0183)     <e:CipherData>
4727 (MSTS-0184)       <e:CipherValue>
4728 (MSTS-0185)         <!-- base64 encoded octets with encrypted RST request-->
4729 (MSTS-0186)         <!-- Unencrypted form: -->
4730 (MSTS-0187)         <!--
4731 (MSTS-0188)     <t:RequestSecurityToken>
4732 (MSTS-0189)       <t:RequestType>
4733 (MSTS-0190)         http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
4734 (MSTS-0191)       </t:RequestType>
4735 (MSTS-0192)     <wsp:AppliesTo
4736 (MSTS-0193)       xmlns:wsp="http://www.w3.org/ns/ws-policy">
4737 (MSTS-0194)         <a:EndpointReference
4738 (MSTS-0195)           xmlns:a="http://www.w3.org/2005/08/addressing">
4739 (MSTS-0196)             <a:Address
4740 (MSTS-0197)               >http://server.example.com/Scenarios5</a:Address>
4741 (MSTS-0198)           </a:EndpointReference>
4742 (MSTS-0199)         </wsp:AppliesTo>
4743 (MSTS-0200)     <t:Entropy>
4744 (MSTS-0201)       <t:BinarySecret
4745 (MSTS-0202)         u:Id="uuid-4acf589c-0076-4a83-8b66-5f29341514b7-3"
4746 (MSTS-0203)         Type=
4747 (MSTS-0204)           "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
4748 (MSTS-0205)       >Uv38QLxDQM9gLoDZ6OwYDiFk094nmwu3Wmay7EdKmhw=</t:BinarySecret>
4749 (MSTS-0206)     </t:Entropy>
4750 (MSTS-0207)     <t:KeyType>
4751 (MSTS-0208)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey
4752 (MSTS-0209)     </t:KeyType>

```



```

4753 (MSTS-0209) <t:KeySize>256</t:KeySize>
4754 (MSTS-0210) <t:ComputedKeyAlgorithm>
4755 (MSTS-0211) http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
4756 (MSTS-0212) </t:ComputedKeyAlgorithm>
4757 (MSTS-0213) <t:SecondaryParameters>
4758 (MSTS-0214) <t:TokenType>
4759 (MSTS-0215) >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4760 1.1#SAMLV1.1</t:TokenType>
4761 (MSTS-0216) <t:KeyType>
4762 (MSTS-0217) >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:Ke
4763 yType>
4764 (MSTS-0218) <t:KeySize>256</t:KeySize>
4765 (MSTS-0219) <t:CanonicalizationAlgorithm>
4766 (MSTS-0220) >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorith
4767 m>
4768 (MSTS-0221) <t:EncryptionAlgorithm>
4769 >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4770 (MSTS-0222) <t:EncryptWith>
4771 >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4772 (MSTS-0223) <t:SignWith>
4773 >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4774 (MSTS-0224) </t:SecondaryParameters>
4775 (MSTS-0225) </t:RequestSecurityToken>
4776 (MSTS-0226) -->
4777 (MSTS-0227) </e:CipherValue>
4778 (MSTS-0228) </e:CipherData>
4779 (MSTS-0229) </e:EncryptedData>
4780 (MSTS-0230) </s:Body>
4781 (MSTS-0231) </s:Envelope>

```

4782 The message above is a request by the Client to the STS in compliance with the STS policy (PSTS-001)-
4783 (PSTS-098). Salient features of this message appropriate to the compliance are described below.

4784 Lines (MSTS-027)-(MSTS-047) contain the **e:EncryptedKey element**, which contains the encrypted
4785 client-generated ephemeral key (K1) (MSTS-039)-(MSTS-043). K1 is encrypted using the STS certificate
4786 (X509T2) which is specified by its Thumbprint identifier in the WS-Security o:SecurityTokenReference
4787 (MSTS-032)-(MSTS-037). The e:ReferenceList in the e:EncryptedKey (MSTS-044)-(MSTS-046) indicates
4788 that the encryption key K1 is used to directly encrypt the message Body which is referenced by the
4789 e:DataReference URI="#_2" (MSTS-045).

4790 Lines (MSTS-048)-(MSTS-071) contain a **WS-Security o:BinarySecurityToken**, which contains the
4791 Client's public X509 certificate (X509T1) that is required for Client authentication to the STS by the
4792 sp:EndorsingToken in the STS policy (PSTS-037)-(PSTS-047).

4793 Lines (MSTS-072)-(MSTS-0144) contain the **WS-SP message signature** in an XML Digital Signature
4794 (dsig) element (namespace = ...xmldsig (MSTS-072)). The elements covered by the dsig Signature are
4795 identified in the dsig Reference elements:

- 4796 ➤ dsig Reference (MSTS-078) covers the s:Body (Id="_1" (MSTS-0176))
- 4797 ➤ dsig Reference (MSTS-087) covers the a:Action (Id="_3" (MSTS-007))
- 4798 ➤ dsig Reference (MSTS-096) covers the a:MessageID (Id="_4" (MSTS-010))
- 4799 ➤ dsig Reference (MSTS-0105) covers the a:ReplyTo (Id="_5" (MSTS-013))
- 4800 ➤ dsig Reference (MSTS-0114) covers the a:To (Id="_6" (MSTS-018))
- 4801 ➤ dsig Reference (MSTS-0123) covers the u:Timestamp (Id="uuid ... 3849-10" (MSTS-023))

4802 all as required by the STS Input and Output Policy sp:SignedParts (PSTS-076)-(PSTS-092), which covers
4803 the first 5 elements above (note: if an element is not present that is identified in the policy (such as
4804 FaultTo or RelatesTo) it obviously is not required to be signed, but if present must be signed). The
4805 u:Timestamp is required to be signed by its presence in the STS endpoint policy (PSTS-033).

4806 Lines (MSTS-0136)-(MSTS-0143) of the **WS-SP message signature contain the dsig KeyInfo**, which
4807 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing
4808 validation key (i.e. the key which can be used to verify this dsig:Signature), which in this case is contained

4809 in the e:EncryptedKey element (Id="uuid ... 3849-9" (MSTS-027)) that was described above. Note: at this
4810 point to verify the Signature, the STS will decrypt the e:EncryptedKey using the private key from the STS
4811 certificate, X509T2, which will produce the client-generated ephemeral signing and encryption key, K1,
4812 which, in turn, may be used to validate the message signature. Note also: at this point the STS only
4813 knows whether the data covered by the message signature is valid or not, but the STS does not yet know
4814 the identity of the entity that actually sent the data. That is covered next:

4815 Lines (MSTS-0145)-(MSTS-0173) contain the **WS-SP endorsing signature**, also in an XML Digital
4816 Signature element. The endorsing signature only covers one element, the message signature element,
4817 which is identified by the dsig Reference element:

4818 ➤ dsig Reference (MSTS-0151) covers the dsig Signature (Id="_0" (MSTS-072))
4819 as required by the STS endpoint policy sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047).

4820 Lines (MSTS-0165)-(MSTS-0174) of the **WS-SP endorsing signature contain the dsig KeyInfo**, which
4821 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing
4822 validation key, which in this case is in the o:BinarySecurityToken element (Id="uuid ... 3849-6"
4823 (MSTS-048)) that was also described above. Note: now the STS finally has the credentials necessary to
4824 authenticate the Client. The STS will generally have an identity database of trusted clients and their
4825 associated X509 certificates. If a client successfully produces a signature that can be validated by the
4826 associated certificate in the STS database, which would have to match the certificate in the
4827 o:BinarySecurityToken element, then the client is presumed to be authenticated to the level of security
4828 provided by this mechanism (strong single factor authentication).

4829 Lines (MSTS-0176)-(MSTS-0230) contain the **SOAP message s:Body element**, which contains its
4830 payload in an e:EncryptedData element (MSTS-0177)-(MSTS-0229). This e:EncryptedData element was
4831 identified above by its Id="_2" in the e:ReferenceList (MSTS-044) of the e:EncryptedKey that was
4832 processed by the STS above to obtain the client ephemeral key (K1), with which this e:EncryptedData
4833 can be decrypted. Generally, when the e:EncryptedKey is processed, this e:EncryptedData would have
4834 been decrypted at that time and made available for processing in the event of successful Client
4835 authentication as described above. Because the contents of this payload are relevant to the rest of this
4836 example, the contents of the payload will be briefly described.

4837 Lines (MSTS-0187)-(MSTS-0225) contain the **decrypted contents of the SOAP message s:Body**
4838 **element**, which contain a WS-Trust t:RequestSecurityToken element. The t:RequestType (MSTS-0189)
4839 is "...Issue", which means this is a request to issue a security token, which is the main service of the STS.
4840 The WS-Policy wsp:AppliesTo element (MSTS-0191)-(MSTS-0198) contains the a:Address of the service,
4841 to which the Client is requesting access, a service which the STS is presumably responsible for
4842 authenticating and equipping validated clients to enable their access to. In this case the service is
4843 identified by the URL <http://server.example.com/Scenarios5>, which was part of the WS-SX Interop
4844 [[WSSX-WSTR-WSSC-INTEROP](#)].

4845 Lines (MSTS-0199)-(MSTS-0205) contain the Client entropy required by the Service policy (P079), which
4846 is entropy data provided by the Client to aid in production of the ephemeral key, K2, that will be used for
4847 Client-Service communication. Lines (MSTS-0206)-(MSTS-0212) contain additional parameters used in
4848 the WS-Trust interface to the STS that will not be described here.

4849 Lines (MSTS-0213)-(MSTS-0224) contain the WS-Trust t:SecondaryParameters, which contains the
4850 contents of the Service policy's RequestSecurityTokenTemplate (P045)-(P059), which the Service
4851 requires that the Client pass to the STS as described above. The t:SecondaryParameters contains the
4852 information the Service has requested about the SAML 1.1 IssuedToken that the STS is being requested
4853 to create and deliver to the Client in order the enable the Client to access the Service successfully.

4854 Assuming everything above has executed successfully, the STS will then issue the SAML 1.1
4855 IssuedToken and return it to the Client in a WS-Trust t:RequestSecurityTokenResponse that is described
4856 next.

4857

4858 Here is an example STS response to the above Client request:

```
4859 (RSTS-001) <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope  
4860 (RSTS-002)     xmlns:a=http://www.w3.org/2005/08/addressing  
4861 (RSTS-003)     xmlns:e=http://www.w3.org/2001/04/xmlenc#
```

```

4862 (RSTS-004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
4863 secext-1.1.xsd"
4864 (RSTS-005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4865 wssecurity-secext-1.0.xsd"
4866 (RSTS-006)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4867 wssecurity-utility-1.0.xsd" >
4868 (RSTS-007)      <s:Header>
4869 (RSTS-008)      <a:Action s:mustUnderstand="1" u:Id="_4">
4870 (RSTS-009)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
4871 (RSTS-010)      </a:Action>
4872 (RSTS-011)      <a:RelatesTo u:Id="_5">
4873 (RSTS-012)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4874 (RSTS-013)      </a:RelatesTo>
4875 (RSTS-014)      <a:To s:mustUnderstand="1" u:Id="_6">
4876 (RSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4877 (RSTS-016)      </a:To>
4878 (RSTS-017)      <o:Security s:mustUnderstand="1">
4879 (RSTS-018)      <u:Timestamp
4880 (RSTS-019)      u:Id="uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4881 (RSTS-020)      <u:Created>2005-10-25T00:47:38.718Z</u:Created>
4882 (RSTS-021)      <u:Expires>2005-10-25T00:52:38.718Z</u:Expires>
4883 (RSTS-022)      </u:Timestamp>
4884 (RSTS-023)      <e:ReferenceList>
4885 (RSTS-024)      <e:DataReference URI="#_3"/>
4886 (RSTS-025)      </e:ReferenceList>
4887 (RSTS-026)      <k:SignatureConfirmation u:Id="_0"
4888 (RSTS-027)      Value="iHGJ+xV2VZTjM1Rc7AQJrwLY/aM="/>
4889 (RSTS-028)      <k:SignatureConfirmation u:Id="_1"
4890 (RSTS-029)      Value=
4891 (RSTS-030)      "Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbnCtZFuQz9tnW35
4892 (RSTS-031)      Lmst6vRefuPA7ewRLYORAOjf92SxMbeVtLrIxQbIQNw0bs4SBSLfAo14="/>
4893 (RSTS-032)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
4894 (RSTS-033)      <SignedInfo>
4895 (RSTS-034)      <CanonicalizationMethod Algorithm=
4896 (RSTS-035)      "http://www.w3.org/2001/10/xml-exc-c14n#" />
4897 (RSTS-036)      <SignatureMethod Algorithm=
4898 (RSTS-037)      "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
4899 (RSTS-038)      <Reference URI="#_2">
4900 (RSTS-039)      <Transforms>
4901 (RSTS-040)      <Transform Algorithm=
4902 (RSTS-041)      "http://www.w3.org/2001/10/xml-exc-c14n#" />
4903 (RSTS-042)      </Transforms>
4904 (RSTS-043)      <DigestMethod Algorithm=
4905 (RSTS-044)      "http://www.w3.org/2000/09/xmldsig#sha1" />
4906 (RSTS-045)      <DigestValue>kKx5bpLLlyucgXQ6exv/PbjsFlA=</DigestValue>
4907 (RSTS-046)      </Reference>
4908 (RSTS-047)      <Reference URI="#_4">
4909 (RSTS-048)      <Transforms>
4910 (RSTS-049)      <Transform Algorithm=
4911 (RSTS-050)      "http://www.w3.org/2001/10/xml-exc-c14n#" />
4912 (RSTS-051)      </Transforms>
4913 (RSTS-052)      <DigestMethod Algorithm=
4914 (RSTS-053)      "http://www.w3.org/2000/09/xmldsig#sha1" />
4915 (RSTS-054)      <DigestValue>LB+VGn4fP2z45jg0Mdzyo8yTAWQ=</DigestValue>
4916 (RSTS-055)      </Reference>
4917 (RSTS-056)      <Reference URI="#_5">
4918 (RSTS-057)      <Transforms>
4919 (RSTS-058)      <Transform Algorithm=
4920 (RSTS-059)      "http://www.w3.org/2001/10/xml-exc-c14n#" />
4921 (RSTS-060)      </Transforms>
4922 (RSTS-061)      <DigestMethod Algorithm=
4923 (RSTS-062)      "http://www.w3.org/2000/09/xmldsig#sha1" />
4924 (RSTS-063)      <DigestValue>izHLxm6V4Lc3Pss9Y6VRv3I5RPw=</DigestValue>
4925 (RSTS-064)      </Reference>

```

```

4926 (RSTS-065) <Reference URI="#_6">
4927 (RSTS-066)   <Transforms>
4928 (RSTS-067)     <Transform Algorithm=
4929 (RSTS-068)       "http://www.w3.org/2001/10/xml-exc-c14n#" />
4930 (RSTS-069)     </Transforms>
4931 (RSTS-070)     <DigestMethod Algorithm=
4932 (RSTS-071)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4933 (RSTS-072)     <DigestValue>6LS4X08vC/GMGay2vwmD8fL7J2U=</DigestValue>
4934 (RSTS-073)     </Reference>
4935 (RSTS-074)     <Reference
4936 (RSTS-075)       URI="#uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4937 (RSTS-076)     <Transforms>
4938 (RSTS-077)       <Transform Algorithm=
4939 (RSTS-078)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4940 (RSTS-079)     </Transforms>
4941 (RSTS-080)     <DigestMethod Algorithm=
4942 (RSTS-081)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4943 (RSTS-082)     <DigestValue>uXGSpCBfbT1fLNBLdGMgy6DGDio=</DigestValue>
4944 (RSTS-083)     </Reference>
4945 (RSTS-084)     <Reference URI="#_0">
4946 (RSTS-085)     <Transforms>
4947 (RSTS-086)       <Transform Algorithm=
4948 (RSTS-087)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4949 (RSTS-088)     </Transforms>
4950 (RSTS-089)     <DigestMethod Algorithm=
4951 (RSTS-090)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4952 (RSTS-091)     <DigestValue>z86w+GrzqRZF56ciuz6ogzVXAUA=</DigestValue>
4953 (RSTS-092)     </Reference>
4954 (RSTS-093)     <Reference URI="#_1">
4955 (RSTS-094)     <Transforms>
4956 (RSTS-095)       <Transform Algorithm=
4957 (RSTS-096)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4958 (RSTS-097)     </Transforms>
4959 (RSTS-098)     <DigestMethod Algorithm=
4960 (RSTS-099)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4961 (RSTS-100)     <DigestValue>Z9Tfd20a5aGngsyOPEvQuE0urvQ=</DigestValue>
4962 (RSTS-101)     </Reference>
4963 (RSTS-102)     </SignedInfo>
4964 (RSTS-103)     <SignatureValue>Q7HhboPUaZyXqUKgG7NCYlhMTXI=</SignatureValue>
4965 (RSTS-104)     <KeyInfo>
4966 (RSTS-105)       <o:SecurityTokenReference
4967 (RSTS-106)         k:TokenType="http://docs.oasis-open.org/wss/
4968 (RSTS-107)           oasis-wss-soap-message-security-1.1#EncryptedKey"
4969 (RSTS-108)           xmlns:k="http://docs.oasis-open.org/wss/
4970 (RSTS-109)             oasis-wss-wssecurity-secext-1.1.xsd">
4971 (RSTS-110)       <o:KeyIdentifier ValueType=
4972 (RSTS-111)         "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4973 (RSTS-112)         1.1.xsd#EncryptedKeySHA1">
4974 (RSTS-113)           CixQW5yEb3mw6XYqD8Ysvrf8cwI=
4975 (RSTS-114)         </o:KeyIdentifier>
4976 (RSTS-115)       </o:SecurityTokenReference>
4977 (RSTS-116)     </KeyInfo>
4978 (RSTS-117)     </Signature>
4979 (RSTS-118)     </o:Security>
4980 (RSTS-119)     </s:Header>
4981 (RSTS-120)     <s:Body u:Id="_2">
4982 (RSTS-121)       <e:EncryptedData Id="_3"
4983 (RSTS-122)         Type="http://www.w3.org/2001/04/xmlenc#Content">
4984 (RSTS-123)       <e:EncryptionMethod Algorithm=
4985 (RSTS-124)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
4986 (RSTS-125)     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
4987 (RSTS-126)       <o:SecurityTokenReference
4988 (RSTS-127)         k:TokenType="http://docs.oasis-open.org/wss/
4989 (RSTS-128)           oasis-wss-soap-message-security-1.1#EncryptedKey"

```

```

4990 (RSTS-0127)          xmlns:k="http://docs.oasis-open.org/wss/
4991 (RSTS-0128)          oasis-wss-wssecurity-secext-1.1.xsd">
4992 (RSTS-0129)          <o:KeyIdentifier ValueType=
4993 (RSTS-0130)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4994 (RSTS-0131)          1.1.xsd#EncryptedKeySHA1">
4995 (RSTS-0132)          CixQW5yEb3mw6XYqD8Ysvrf8cwI=
4996 (RSTS-0133)          </o:KeyIdentifier>
4997 (RSTS-0134)          </o:SecurityTokenReference>
4998 (RSTS-0135)          </KeyInfo>
4999 (RSTS-0136)          <e:CipherData>
5000 (RSTS-0137)          <e:CipherValue>
5001 (RSTS-0138)          <!--base64 encoded octets of encrypted RSTR-->
5002 (RSTS-0139)          <!--
5003 (RSTS-0140)          <t:RequestSecurityTokenResponseCollection>
5004 (RSTS-0141)          <t:RequestSecurityTokenResponse>
5005 (RSTS-0142)          <t:TokenType>
5006 (RSTS-0143)          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5007 (RSTS-0144)          1.1#SAMLV1.1
5008 (RSTS-0145)          </t:TokenType>
5009 (RSTS-0146)          <t:KeySize>256</t:KeySize>
5010 (RSTS-0147)          <t:RequestedAttachedReference>
5011 (RSTS-0148)          <o:SecurityTokenReference>
5012 (RSTS-0149)          <o:KeyIdentifier ValueType=
5013 (RSTS-0150)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5014 (RSTS-0151)          1.0#SAMLAssertionID">
5015 (RSTS-0152)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5016 (RSTS-0153)          </o:KeyIdentifier>
5017 (RSTS-0154)          </o:SecurityTokenReference>
5018 (RSTS-0155)          </t:RequestedAttachedReference>
5019 (RSTS-0156)          <t:RequestedUnattachedReference>
5020 (RSTS-0157)          <o:SecurityTokenReference>
5021 (RSTS-0158)          <o:KeyIdentifier ValueType=
5022 (RSTS-0159)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5023 (RSTS-0160)          1.0#SAMLAssertionID">
5024 (RSTS-0161)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5025 (RSTS-0162)          </o:KeyIdentifier>
5026 (RSTS-0163)          </o:SecurityTokenReference>
5027 (RSTS-0164)          </t:RequestedUnattachedReference>
5028 (RSTS-0165)          <t:Lifetime>
5029 (RSTS-0166)          <u:Created>2005-10-24T20:19:26.526Z</u:Created>
5030 (RSTS-0167)          <u:Expires>2005-10-25T06:24:26.526Z</u:Expires>
5031 (RSTS-0168)          </t:Lifetime>
5032 (RSTS-0169)          <t:RequestedSecurityToken>
5033 (RSTS-0170)          <saml:Assertion MajorVersion="1" MinorVersion="1"
5034 (RSTS-0171)          AssertionID="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16"
5035 (RSTS-0172)          Issuer="Test STS" IssueInstant="2005-10-24T20:24:26.526Z"
5036 (RSTS-0173)          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
5037 (RSTS-0174)          <saml:Conditions NotBefore="2005-10-24T20:19:26.526Z"
5038 (RSTS-0175)          NotOnOrAfter="2005-10-25T06:24:26.526Z">
5039 (RSTS-0176)          <saml:AudienceRestrictionCondition>
5040 (RSTS-0177)          <saml:Audience
5041 (RSTS-0178)          >http://server.example.com/Scenarios5</saml:Audience>
5042 (RSTS-0179)          </saml:AudienceRestrictionCondition>
5043 (RSTS-0180)          </saml:Conditions>
5044 (RSTS-0181)          <saml:Advice>
5045 (RSTS-0182)          </saml:Advice>
5046 (RSTS-0183)          <saml:AttributeStatement>
5047 (RSTS-0184)          <saml:Subject>
5048 (RSTS-0185)          <saml:SubjectConfirmation>
5049 (RSTS-0186)          <saml:ConfirmationMethod>
5050 (RSTS-0187)          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
5051 (RSTS-0188)          </saml:ConfirmationMethod>
5052 (RSTS-0189)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5053 (RSTS-0190)          <e:EncryptedKey

```

```

5054 (RSTS-0187)          xmlns:e="http://www.w3.org/2001/04/xmlenc#">
5055 (RSTS-0188)          <e:EncryptionMethod Algorithm=
5056 (RSTS-0189)          "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
5057 (RSTS-0190)          </e:EncryptionMethod>
5058 (RSTS-0191)          <KeyInfo>
5059 (RSTS-0192)          <o:SecurityTokenReference xmlns:o=
5060 (RSTS-0193)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5061 (RSTS-0194)          secext-1.0.xsd">
5062 (RSTS-0195)          <o:KeyIdentifier ValueType=
5063 (RSTS-0196)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5064 (RSTS-0197)          1.1.xsd#ThumbprintSHA1">
5065 (RSTS-0198)          NQM0IBvuplAtETQvk+6gn8C13wE=
5066 (RSTS-0199)          </o:KeyIdentifier>
5067 (RSTS-0200)          </o:SecurityTokenReference>
5068 (RSTS-0201)          </KeyInfo>
5069 (RSTS-0202)          <e:CipherData>
5070 (RSTS-0203)          <e:CipherValue>
5071 (RSTS-0204)          EEcYjwNoYcJ+20xTYE5e/fixl5K0gzgrfaYAxkDFv/VXiuKfl084h8PmogTfM+azcgAf
5072 (RSTS-0205)          mArVQvOyKWXRb5vmXYfVHLlhZTbXacy+nowSUNnEjp37VDbI3RJ5k6tBHF+owONM/P6G
5073 (RSTS-0206)          PNZ9ZqJi1lGDgWJkFsJzNZXNbbMgwuFu3cA=</e:CipherValue>
5074 (RSTS-0207)          </e:CipherData>
5075 (RSTS-0208)          </e:EncryptedKey>
5076 (RSTS-0209)          </KeyInfo>
5077 (RSTS-0210)          </saml:SubjectConfirmation>
5078 (RSTS-0211)          </saml:Subject>
5079 (RSTS-0212)          </saml:AttributeStatement>
5080 (RSTS-0213)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5081 (RSTS-0214)          <SignedInfo>
5082 (RSTS-0215)          <CanonicalizationMethod Algorithm=
5083 (RSTS-0216)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5084 (RSTS-0217)          </CanonicalizationMethod>
5085 (RSTS-0218)          <SignatureMethod Algorithm=
5086 (RSTS-0219)          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
5087 (RSTS-0220)          </SignatureMethod>
5088 (RSTS-0221)          <Reference
5089 (RSTS-0222)          URI="#uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16">
5090 (RSTS-0223)          <Transforms>
5091 (RSTS-0224)          <Transform Algorithm=
5092 (RSTS-0225)          "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
5093 (RSTS-0226)          </Transform>
5094 (RSTS-0227)          <Transform Algorithm=
5095 (RSTS-0228)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5096 (RSTS-0229)          </Transform>
5097 (RSTS-0230)          </Transforms>
5098 (RSTS-0231)          <DigestMethod Algorithm=
5099 (RSTS-0232)          "http://www.w3.org/2000/09/xmldsig#sha1">
5100 (RSTS-0233)          </DigestMethod>
5101 (RSTS-0234)          <DigestValue
5102 (RSTS-0235)          >7nHBrFPsm+LEFAoV4NoQPoEl5Lk=</DigestValue>
5103 (RSTS-0236)          </Reference>
5104 (RSTS-0237)          </SignedInfo>
5105 (RSTS-0238)          <SignatureValue
5106 (RSTS-0239)          >TugV4pTIwCH87bLD4jjiMgVGtkbRb1tRlHXJArL34A/YfA4AnGBLXB4pJdUsUxMUTbQ
5107 (RSTS-0240)          14PoGgEsdLNg8C77peARELGP1/Tqw7T3u5zBYHxCHCiv2FWBBfeOmwJmgoaBf8XZJ4Al
5108 (RSTS-0241)          yqPq61P61jrQjzJafpHuYpAZnZQSVsiJaBPQ=</SignatureValue>
5109 (RSTS-0242)          <KeyInfo>
5110 (RSTS-0243)          <o:SecurityTokenReference xmlns:o=
5111 (RSTS-0244)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5112 (RSTS-0245)          secext-1.0.xsd">
5113 (RSTS-0246)          <o:KeyIdentifier ValueType=
5114 (RSTS-0247)          http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5115 (RSTS-0248)          1.1.xsd#ThumbprintSHA1
5116 (RSTS-0249)          >W+rqYBmLmVEG//scD7Vo8Kq5G7I=</o:KeyIdentifier>
5117 (RSTS-0250)          </o:SecurityTokenReference>

```

```

5118 (RSTS-0247)         </KeyInfo>
5119 (RSTS-0248)         </Signature>
5120 (RSTS-0249)         </saml:Assertion>
5121 (RSTS-0250)         </t:RequestedSecurityToken>
5122 (RSTS-0251)         <t:RequestedProofToken>
5123 (RSTS-0252)         <t:BinarySecret
5124 (RSTS-0253)         u:Id="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-14"
5125 (RSTS-0254)         >zT8LWAUwUrIVKA/rkCr0kx1EmKAehcB6TGWJuAgucBM=</t:BinarySecret>
5126 (RSTS-0255)         </t:RequestedProofToken>
5127 (RSTS-0256)         </t:RequestSecurityTokenResponse>
5128 (RSTS-0257)         </t:RequestSecurityTokenResponseCollection>
5129 (RSTS-0258)         -->
5130 (RSTS-0259)         </e:CipherValue>
5131 (RSTS-0260)         </e:CipherData>
5132 (RSTS-0261)         </e:EncryptedData>
5133 (RSTS-0262)         </s:Body>
5134 (RSTS-0263)         </s:Envelope>

```

5135 From here on the description will be less detailed except where new concepts that have not been covered
5136 previously in this example. For instance, tracing the dsig References to their associated elements and
5137 policy requirements will not be detailed since it follows the same patterns that have just been described in
5138 the message above.

5139 The message above is a response from the STS to the Client that contains the requested security tokens
5140 that the Client needs to access the Service.

5141 Lines (RSTS-023)-(RSTS-025) contain a **standalone e:ReferenceList** that has an e:DataReference
5142 (Id="_3") that points to the e:EncryptedData element in the SOAP s:Body (RSTS-0119). This
5143 e:DataReference will be used later in the processing of this message.

5144 Lines (RSTS-026)-(RSTS-031) contain **2 WS-Security 1.1 k:SignatureConfirmation elements** that
5145 indicate to the Client that the STS has processed the data in the Client request and in particular, has
5146 processed the information covered by the 2 dsig Signatures in that request, that are identified by their
5147 respective dsig SignatureValue elements (see lines (MSTS-0134)-(MSTS-0135) and (MSTS-0161)-
5148 (MSTS-0163) in the Client request to the STS above to compare SignatureValues).

5149 Lines (RSTS-032)-(RSTS-0115) contain the **WS-SP message signature** for this message.

5150 Lines (RSTS-0103)-(RSTS-0114) contain the **WS-SP message signature dsig KeyInfo element**, which
5151 contains a WS-Security 1.1 o:SecurityTokenReference, which contains an o:KeyIdentifier of ValueType
5152 "... EncryptedKeySHA1". This is a WS-Security 1.1 construct [[WS_SECURITY_11](#)] used to reference a
5153 security token that is not included in the message, which in this case is the Client-generated ephemeral
5154 key, K1, that the Client used to prepare the request and delivered to the STS in the e:EncryptedKey
5155 element in that request (MSTS-027) that was described in detail above. The contents of the
5156 o:KeyIdentifier (RSTS-0111) consist of the SHA1 of K1. This should be sufficient information to let the
5157 Client know that the STS used its X509 certificate, X509T2, to decrypt K1 from the e:EncryptedKey in the
5158 Client request, which will enable the Client to trust the contents of this STS response.

5159 Lines (RSTS-0119)-(RSTS-0261) contain the **e:EncryptedData**, which is also provided in decrypted form
5160 for instructive purposes. As mentioned above, this element is referred to by the standalone
5161 e:ReferenceList element in the WS-Security header (RSTS-023) and as a result can be not tied to any
5162 particular encryption key as described in [[WS_SECURITY_11](#)], and since referenced will be processed by
5163 WS-Security.

5164 Lines (RSTS-0123)-(RSTS-0134) contain the **e:EncryptedData dsig KeyInfo**, which refers to the same
5165 ephemeral key, K1, as described above for the message signature for this message.

5166 Lines (RSTS-0140)-(RSTS-0256) contain the **decrypted WS-Trust t:RequestSecurityTokenResponse**
5167 from the STS. Briefly, lines (RSTS-0145)-(RSTS-0160) contain WS-Security o:SecurityTokenReference
5168 elements that refer to the SAML 1.1 Assertion that is provided below. These are convenience elements
5169 for the Client to use in subsequent message preparation where the SAML Assertion is used.

5170 Lines (RSTS-0165)-(RSTS-0250) contain **the actual t:RequestedSecurityToken** that has been the main
5171 object of discussion up until this point, which is the SAML 1.1 Assertion, saml:Assertion (RSTS-0166)-
5172 (RSTS-0249), that is the sp:IssuedToken provided by the STS for the Client to use to access the Service.

5173 Lines (RSTS-0170)-(RSTS-0176) of the saml:Assertion contain the **saml:Conditions** that specify that this
5174 token is intended only for the Service, identified by the URL in the **saml:Audience** element (RSTS-0174).

5175 Lines (RSTS-0181)-(RSTS-0208) contain the **all-important saml:SubjectConfirmation element**, which
5176 contains the **STS-generated encrypted ephemeral key, K3**, that will be used by the Client and Service
5177 to communicate securely. There are **2 copies of this key, K3**, in this t:RequestSecurityTokenResponse.
5178 This copy is for the Service. The Client's copy is described below. In any event, the
5179 saml:SubjectConfirmation element contains a dsig KeyInfo (RSTS-0185)-(RSTS-0207), which contains an
5180 e:EncryptedKey element (RSTS-0186)-(RSTS-0206), which, in turn, contains a second dsig KeyInfo,
5181 which contains a WS-Security o:SecurityTokenReference element that contains an o:KeyIdentifier (RSTS-
5182 0194)-(RSTS-0197) that identifies the key, X509T3, the Service's public X509 certificate, that can be
5183 used by the Service to decrypt the ultimate object here, which is the ephemeral key, K3, contained in the
5184 e:CipherData (RSTS-0200)-(RSTS-0205). The Service's public X509 certificate is identified by its WS-
5185 Security 1.1 ThumbprintSHA1 (RSTS-0196). Therefore, when the Service receives this saml:Assertion, it
5186 has the ability to obtain the ephemeral key, K3, contained in the saml:SubjectConfirmation, with which it
5187 can securely communicate with the Client, based on the assurances provided by the STS.

5188 Lines (RSTS-0211)-(RSTS-0248) contain the **saml:Assertion dsig Signature** that is contained in and
5189 covers the saml:Assertion via the saml:AssertionID, the value of which appears on lines (RSTS-0220) and
5190 (RSTS-0167).

5191 Lines (RSTS-0240)-(RSTS-0247) contain the **saml:Assertion dsig Signature KeyInfo element**, which
5192 contains the ThumbprintSHA1 of the **STS public key, X509T2**, which is the same certificate that the
5193 Client referenced in the e:EncryptedKey when it made initial contact with the STS above (MSTS-035).
5194 This completes the initial discussion of the characteristics of the IssuedToken saml:Assertion that will be
5195 used in the remainder of this example.

5196 Lines (RSTS-0251)-(RSTS-0255) of the t:RequestSecurityTokenResponse contains the
5197 **t:RequestedProofToken**, which contains the **Client copy of the STS-generated ephemeral key, K3**,
5198 which will be used in the Client-Service communication to authenticate the client to the Service.

5199 At this point we have completed the setup portion of this example that has enabled secure trusted
5200 communication between the Client and Service as governed by an STS. The exact strength of the
5201 security protecting this example would be the subject of an official security analysis that is beyond the
5202 scope of this document, however, the intent has been to provide sufficient detail that all parties concerned
5203 with such an analysis would have sufficient context to understand and evaluate such an analysis.

5204

5205 Here is an example of a Client request to the Service using the tokens from the STS response:

5206

```
5207 (M001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5208 (M002)   xmlns:a=http://www.w3.org/2005/08/addressing
5209 (M003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5210 (M004)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5211 wssecurity-secext-1.0.xsd"
5212 (M005)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5213 secureconversation/200512"
5214 (M006)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5215 wssecurity-utility-1.0.xsd" >
5216 (M007)   <s:Header>
5217 (M008)     <a:Action s:mustUnderstand="1" u:Id="_5">
5218 (M009)       http://example.org/Ping
5219 (M010)     </a:Action>
5220 (M011)     <a:MessageID u:Id="_6">
5221 (M012)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5222 (M013)     </a:MessageID>
5223 (M014)     <a:ReplyTo u:Id="_7">
5224 (M015)       <a:Address>
5225 (M016)         http://www.w3.org/2005/08/addressing/anonymous
5226 (M017)       </a:Address>
5227 (M018)     </a:ReplyTo>
5228 (M019)     <a:To s:mustUnderstand="1" u:Id="_8">
5229 (M020)       http://server.example.com/Scenarios5
```



```

5230 (M021) </a:To>
5231 (M022) <o:Security s:mustUnderstand="1">
5232 (M023) <u:Timestamp
5233 (M024) <u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5234 (M025) <u:Created>2005-10-25T00:47:38.222Z</u:Created>
5235 (M026) <u:Expires>2005-10-25T00:52:38.222Z</u:Expires>
5236 (M027) </u:Timestamp>
5237 (M028) <e:EncryptedKey
5238 (M029) <Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-4">
5239 (M030) <e:EncryptionMethod Algorithm=
5240 (M031) http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5241 (M032) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5242 (M033) <o:SecurityTokenReference>
5243 (M034) <o:KeyIdentifier ValueType=
5244 (M035) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5245 1.1.xsd#ThumbprintSHA1">
5246 (M036) NQM0IBvuplAtETQvk+6gn8C13wE=
5247 (M037) </o:KeyIdentifier>
5248 (M038) </o:SecurityTokenReference>
5249 (M039) </KeyInfo>
5250 (M040) <e:CipherData>
5251 (M041) <e:CipherValue>
5252 (M042) <!-- base64 encoded octets of encrypted key K2 -->
5253 (M043) </e:CipherValue>
5254 (M044) </e:CipherData>
5255 (M045) </e:EncryptedKey>
5256 (M046) <sc:DerivedKeyToken u:Id="_0" >
5257 (M047) <o:SecurityTokenReference>
5258 (M048) <o:Reference
5259 (M049) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5260 (M050) </o:SecurityTokenReference>
5261 (M051) <sc:Offset>0</sc:Offset>
5262 (M052) <sc:Length>24</sc:Length>
5263 (M053) <sc:Nonce>7hI6U160HavfYgppquHWuQ==</sc:Nonce>
5264 (M054) </sc:DerivedKeyToken>
5265 (M055) <sc:DerivedKeyToken u:Id="_2">
5266 (M056) <o:SecurityTokenReference>
5267 (M057) <o:Reference
5268 (M058) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5269 (M059) </o:SecurityTokenReference>
5270 (M060) <sc:Nonce>OEu+WEEUxPFRQK7SCFAnEQ==</sc:Nonce>
5271 (M061) </sc:DerivedKeyToken>
5272 (M062) <e:ReferenceList>
5273 (M063) <e:DataReference URI="#_4"/>
5274 (M064) </e:ReferenceList>
5275 (M065) <!-- encrypted SAML assertion -->
5276 (M066) <e:EncryptedData Id="_3"
5277 (M067) Type="http://www.w3.org/2001/04/xmlenc#Element">
5278 (M068) <e:EncryptionMethod Algorithm=
5279 (M069) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5280 (M070) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5281 (M071) <!-- encrypted Key K2 -->
5282 (M072) <e:EncryptedKey>
5283 (M073) <e:EncryptionMethod Algorithm=
5284 (M074) "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5285 (M075) <KeyInfo>
5286 (M076) <o:SecurityTokenReference>
5287 (M077) <o:KeyIdentifier ValueType=
5288 (M078) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5289 1.1.xsd#ThumbprintSHA1">
5290 (M079) NQM0IBvuplAtETQvk+6gn8C13wE=
5291 (M080) </o:KeyIdentifier>
5292 (M081) </o:SecurityTokenReference>
5293 (M082) </KeyInfo>

```

```

5294 (M083) <e:CipherData>
5295 (M084) <e:CipherValue>
5296 (M085)
5297 cb7+JW2idPNSarK9quqCe9PQwmW2hoUghuyKRe+I9zOts6HaMcg73LqCWuK/jtdpvNl6
5298 (M086) GT/ZDYfcAJ7NLyMGxSiwi4DULtOShqS60TYBIKgUKiA+zXNl2koVsy7amcUhPMIT6/fo
5299 (M087) hH+6MZDA4t6jomcyhlCiW8d9IAzSWFkfg2k=
5300 (M088) </e:CipherValue>
5301 (M089) </e:CipherData>
5302 (M090) </e:EncryptedKey>
5303 (M091) </KeyInfo>
5304 (M092) <e:CipherData>
5305 (M093) <e:CipherValue>
5306 (M094) <!-- base64 encoded octets from SAML assertion encrypted
5307 (M095) with the encrypted key K2 above -->
5308 (M096) <!-- SAML assertion element is identical as received in
5309 (M097) RSTR , unencrypted form is omitted for brevity -->
5310 (M098) <!--.....-->
5311 (M099) </e:CipherValue>
5312 (M0100) </e:CipherData>
5313 (M0101) </e:EncryptedData>
5314 (M0102)
5315 (M0103) <sc:DerivedKeyToken u:Id="_9">
5316 (M0104) <o:SecurityTokenReference>
5317 (M0105) <o:KeyIdentifier ValueType=
5318 (M0106) "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5319 1.0#SAMLAssertionID">
5320 (M0107) <b>uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16</b>
5321 (M0108) </o:KeyIdentifier>
5322 (M0109) </o:SecurityTokenReference>
5323 (M0110) <sc:Offset>0</sc:Offset>
5324 (M0111) <sc:Length>24</sc:Length>
5325 (M0112) <sc:Nonce>pgnS/VDSzJn6SFz+Vy23JA==</sc:Nonce>
5326 (M0113) </sc:DerivedKeyToken>
5327 (M0114) <Signature Id="_1" xmlns="http://www.w3.org/2000/09/xmldsig#">
5328 (M0115) <SignedInfo>
5329 (M0116) <CanonicalizationMethod Algorithm=
5330 (M0117) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5331 (M0118) <SignatureMethod Algorithm=
5332 (M0119) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5333 (M0120) <Reference URI="#_3">
5334 (M0121) <Transforms>
5335 (M0122) <Transform Algorithm=
5336 (M0123) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5337 (M0124) </Transforms>
5338 (M0125) <DigestMethod Algorithm=
5339 (M0126) "http://www.w3.org/2000/09/xmldsig#sha1" />
5340 (M0127) <DigestValue>eQdQVGRkVI1YfKJBw7vOYCOeLQw=</DigestValue>
5341 (M0128) </Reference>
5342 (M0129) <Reference URI="#_5">
5343 (M0130) <Transforms>
5344 (M0131) <Transform Algorithm=
5345 (M0132) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5346 (M0133) </Transforms>
5347 (M0134) <DigestMethod Algorithm=
5348 (M0135) "http://www.w3.org/2000/09/xmldsig#sha1" />
5349 (M0136) <DigestValue>xxyKpp5RZ2TebKca2IG0afIgcxk=</DigestValue>
5350 (M0137) </Reference>
5351 (M0138) <Reference URI="#_6">
5352 (M0139) <Transforms>
5353 (M0140) <Transform Algorithm=
5354 (M0141) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5355 (M0142) </Transforms>
5356 (M0143) <DigestMethod Algorithm=
5357 (M0144) "http://www.w3.org/2000/09/xmldsig#sha1" />

```

```

5358 (M0145) <DigestValue>WyGDDyYbL/hQZJfE3Yx2aK3RkK8=</DigestValue>
5359 (M0146) </Reference>
5360 (M0147) <Reference URI="#_7">
5361 (M0148) <Transforms>
5362 (M0149) <Transform Algorithm=
5363 (M0150) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5364 (M0151) </Transforms>
5365 (M0152) <DigestMethod Algorithm=
5366 (M0153) "http://www.w3.org/2000/09/xmldsig#sha1" />
5367 (M0154) <DigestValue>AEOH0t2KYR8mivgqUGDrgMtxgEQ=</DigestValue>
5368 (M0155) </Reference>
5369 (M0156) <Reference URI="#_8">
5370 (M0157) <Transforms>
5371 (M0158) <Transform Algorithm=
5372 (M0159) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5373 (M0160) </Transforms>
5374 (M0161) <DigestMethod Algorithm=
5375 (M0162) "http://www.w3.org/2000/09/xmldsig#sha1" />
5376 (M0163) <DigestValue>y8n6Dxd3DbD6TR6d6H/oVWsV4yE=</DigestValue>
5377 (M0164) </Reference>
5378 (M0165) <Reference
5379 (M0166) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5380 (M0167) <Transforms>
5381 (M0168) <Transform Algorithm=
5382 (M0169) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5383 (M0170) </Transforms>
5384 (M0171) <DigestMethod Algorithm=
5385 (M0172) "http://www.w3.org/2000/09/xmldsig#sha1" />
5386 (M0173) <DigestValue>/Cc+bGkkeQ6jlvXZx8PGgmF6MjI=</DigestValue>
5387 (M0174) </Reference>
5388 (M0175) </SignedInfo>
5389 (M0176) <!--base64 encoded signature value -->
5390 (M0177) <SignatureValue>EyKUHUuffPUPE/ZjaFrMJJ5KLKY=</SignatureValue>
5391 (M0178) <KeyInfo>
5392 (M0179) <o:SecurityTokenReference>
5393 (M0180) <o:Reference URI="#_0" />
5394 (M0181) </o:SecurityTokenReference>
5395 (M0182) </KeyInfo>
5396 (M0183) </Signature>
5397 (M0184) <!-- signature over the primary signature above
5398 (M0185) using the key derived from the proof-key, K3,
5399 (M0186) associated with SAML assertion -->
5400 (M0187) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5401 (M0188) <SignedInfo>
5402 (M0189) <CanonicalizationMethod Algorithm=
5403 (M0190) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5404 (M0191) <SignatureMethod Algorithm=
5405 (M0192) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5406 (M0193) <Reference URI="#_1">
5407 (M0194) <Transforms>
5408 (M0195) <Transform Algorithm=
5409 (M0196) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5410 (M0197) </Transforms>
5411 (M0198) <DigestMethod Algorithm=
5412 (M0199) "http://www.w3.org/2000/09/xmldsig#sha1" />
5413 (M0200) <DigestValue>TMSmLlgeUn8cxyb6OYe5Q2nUuxY=</DigestValue>
5414 (M0201) </Reference>
5415 (M0202) </SignedInfo>
5416 (M0203) <SignatureValue>Fh4NyOpAi+NqVFiHBgHwyvzah9I=</SignatureValue>
5417 (M0204) <KeyInfo>
5418 (M0205) <o:SecurityTokenReference>
5419 (M0206) <o:Reference URI="#_9" />
5420 (M0207) </o:SecurityTokenReference>
5421 (M0208) </KeyInfo>

```

```

5422 (M0209)          </Signature>
5423 (M0210)          </o:Security>
5424 (M0211)          </s:Header>
5425 (M0212)          <s:Body u:Id="_3">
5426 (M0213)          <e:EncryptedData Id="_4"
5427 (M0214)              Type="http://www.w3.org/2001/04/xmlenc#Content">
5428 (M0215)          <e:EncryptionMethod Algorithm=
5429 (M0216)              "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5430 (M0217)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5431 (M0218)          <o:SecurityTokenReference >
5432 (M0219)              <o:Reference URI="#_2"/>
5433 (M0220)          </o:SecurityTokenReference>
5434 (M0221)          </KeyInfo>
5435 (M0222)          <e:CipherData>
5436 (M0223)              <e:CipherValue>
5437 (M0224)                  <!-- base64 encoded octets of encrypted body content-->
5438 (M0225)              </e:CipherValue>
5439 (M0226)          </e:CipherData>
5440 (M0227)          </e:EncryptedData>
5441 (M0228)          </s:Body>
5442 (M0229)          </s:Envelope>

```

5443 The message above is a request from the Client to the Service using the tokens provided by the STS
5444 above.

5445 Lines (M022)-(M0210) contain the **WS-Security o:Security header** for this request.

5446 Lines (M028)-(M045) contain an **e:EncryptedKey** that contains the **ephemeral key, K2**, for the Client-
5447 Service communication. The service must use its X509T3 (M034)-(M037) to decrypt this Client-generated
5448 ephemeral key, K2.

5449 Lines (M046)-(M054) contain a **WS-SecureConversation sc:DerivedKeyToken**, that contains the
5450 information required to **generate the signing key, DKT1(K2)** [**WS_SECURE_CONV**], including an
5451 sc:Nonce (M053), and a WS-Security SecurityTokenReference that contains a direct reference (M049) to
5452 the e:EncryptedKey, K2 (M042). This derived key, DKT1(K2), is used to sign the message in the
5453 message signature element (M0114)-(M0183).

5454 Lines (M055)-(M054) provide similar **sc:DerivedKeyToken** constructs the **generate the encrypting key**
5455 **DKT2(K2)**. This derived key, DKT2(K2), is used to encrypt the message body, resulting in the
5456 EncryptedData element in the s:Body, on lines (M0213)-(M0227).

5457 Lines (M062)-(M064) provide an **e:ReferenceList to reference the e:EncryptedData in the s:Body**
5458 (M0213), which contains the Client request for the Service. The s:Body payload data in this Client request
5459 is not of interest to the security properties of this example and will not be shown in decrypted form.

5460 Lines (M066)-(M0101) contain an **e:EncryptedData element that contains the saml:Assertion**
5461 described in the STS response above. This e:EncryptedData contains a dsig KeyInfo, which, in turn,
5462 contains an e:EncryptedKey element (M072)-(M090), which contains the Client-generated ephemeral
5463 key, K2, which was used by the Client to directly encrypt the saml:Assertion. The encryption key, K2, may
5464 be decrypted, again using the Service public X509 certificate, again identified by its ThumbprintSHA1,
5465 (M077)-(M080).

5466 Lines (M0103)-(M0113) contain a **third sc:DerivedKeyToken** that contains the information necessary for
5467 the Service to **generate the endorsing signing key, DKT3(K2)**.

5468 Lines (M0114)-(M0183) contains the **message signature**. It is **signed using** the key identified in the dsig
5469 KeyInfo (M0178)-(M0182), which contains a reference to the **derived signing key, DKT1(K2)**, which may
5470 be constructed by the service using the sc:DerivedKeyToken (M046) described above.

5471 Lines (M0187)-(M0209) contain the **message endorsing signature**. It is **signed using the client proof**
5472 **key, K3**, that was generated by the STS. Note that the Service should compare this key, K3, with the one
5473 in the saml:SubjectConfirmation in the decrypted saml:Assertion to verify that the Client is using the same
5474 proof key, K3, that is contained in the saml:Assertion that authenticates this request.

5475 Lines (M0213)-(M0227) contain the **SOAP s:Body message payload, e:EncryptedData**, which may be
5476 decrypted using the sc:DerivedKeyToken (M055) to generate the decryption key DKT2(K2).

5477

5478 Here is an example response from the Service to the Client:

5479

```
(R001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5481 (R002)   xmlns:a=http://www.w3.org/2005/08/addressing
5482 (R003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5483 (R004)   xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5484 1.1.xsd"
5485 (R005)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5486 wssecurity-secext-1.0.xsd"
5487 (R006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5488 secureconversation/200512"
5489 (R007)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5490 wssecurity-utility-1.0.xsd">
5491 (R008)   <s:Header>
5492 (R009)     <a:Action s:mustUnderstand="1" u:Id="_6">
5493 (R010)       http://example.org/PingResponse
5494 (R011)     </a:Action>
5495 (R012)     <a:RelatesTo u:Id="_7">
5496 (R013)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5497 (R014)     </a:RelatesTo>
5498 (R015)     <a:To s:mustUnderstand="1" u:Id="_8">
5499 (R016)       http://www.w3.org/2005/08/addressing/anonymous
5500 (R017)     </a:To>
5501 (R018)     <o:Security s:mustUnderstand="1">
5502 (R019)       <u:Timestamp u:Id="uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5503 (R020)         <u:Created>2005-10-25T00:47:38.921Z</u:Created>
5504 (R021)         <u:Expires>2005-10-25T00:52:38.921Z</u:Expires>
5505 (R022)       </u:Timestamp>
5506 (R023)       <sc:DerivedKeyToken u:Id="_0" >
5507 (R024)         <o:SecurityTokenReference
5508 (R025)           k:TokenType="http://docs.oasis-open.org/wss/
5509 (R026)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5510 (R027)           xmlns:k="http://docs.oasis-open.org/wss/
5511 (R028)             oasis-wss-wssecurity-secext-1.1.xsd">
5512 (R029)             <o:KeyIdentifier ValueType=
5513 (R030)               "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5514 1.1.xsd#EncryptedKeySHA1"
5515 (R031)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5516 (R032)           </o:SecurityTokenReference>
5517 (R033)           <sc:Offset>0</sc:Offset>
5518 (R034)           <sc:Length>24</sc:Length>
5519 (R035)           <sc:Nonce>KFjylGb73BubLul0ZGgx+w==</sc:Nonce>
5520 (R036)         </sc:DerivedKeyToken>
5521 (R037)       <sc:DerivedKeyToken u:Id="_3">
5522 (R038)         <o:SecurityTokenReference
5523 (R039)           k:TokenType="http://docs.oasis-open.org/wss/
5524 (R040)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5525 (R041)           xmlns:k="http://docs.oasis-open.org/wss/
5526 (R042)             oasis-wss-wssecurity-secext-1.1.xsd">
5527 (R043)             <o:KeyIdentifier ValueType=
5528 (R044)               "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5529 1.1.xsd#EncryptedKeySHA1"
5530 (R045)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5531 (R046)           </o:SecurityTokenReference>
5532 (R047)           <sc:Nonce>omyh+Eg6XIa8q3V5IkHiXg==</sc:Nonce>
5533 (R048)         </sc:DerivedKeyToken>
5534 (R049)       <e:ReferenceList>
5535 (R050)         <e:DataReference URI="#_5"/>
5536 (R051)       </e:ReferenceList>
5537 (R052)       <k:SignatureConfirmation u:Id="_1"
5538 (R053)         Value="EyKUHUuffPUPE/ZjaFrMJJ5KLKY="/>
5539 (R054)       <k:SignatureConfirmation u:Id="_2"
```

```

5540 (R055) Value="Fh4NyOpAi+NqVFihBgHWyvz9I=" />
5541 (R056) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5542 (R057)   <SignedInfo>
5543 (R058)     <CanonicalizationMethod Algorithm=
5544 (R059)       "http://www.w3.org/2001/10/xml-exc-c14n#" />
5545 (R060)     <SignatureMethod Algorithm=
5546 (R061)       "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5547 (R062)     <Reference URI="#_4">
5548 (R063)       <Transforms>
5549 (R064)         <Transform Algorithm=
5550 (R065)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5551 (R066)         </Transforms>
5552 (R067)         <DigestMethod Algorithm=
5553 (R068)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5554 (R069)         <DigestValue>y/oItF5TcTOFan7SavhZTTTv48M=</DigestValue>
5555 (R070)       </Reference>
5556 (R071)     <Reference URI="#_6">
5557 (R072)       <Transforms>
5558 (R073)         <Transform Algorithm=
5559 (R074)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5560 (R075)         </Transforms>
5561 (R076)         <DigestMethod Algorithm=
5562 (R077)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5563 (R078)         <DigestValue>X4UIaLWnaAWTriw4UJ/SFDgm090=</DigestValue>
5564 (R079)       </Reference>
5565 (R080)     <Reference URI="#_7">
5566 (R081)       <Transforms>
5567 (R082)         <Transform Algorithm=
5568 (R083)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5569 (R084)         </Transforms>
5570 (R085)         <DigestMethod Algorithm=
5571 (R086)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5572 (R087)         <DigestValue>vqy8/D4CDCaIlnnd4wllQjyp+qM=</DigestValue>
5573 (R088)       </Reference>
5574 (R089)     <Reference URI="#_8">
5575 (R090)       <Transforms>
5576 (R091)         <Transform Algorithm=
5577 (R092)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5578 (R093)         </Transforms>
5579 (R094)         <DigestMethod Algorithm=
5580 (R095)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5581 (R096)         <DigestValue>H1lvLAr5g8pbZ6jfZ+2WNYiNjiM=</DigestValue>
5582 (R097)       </Reference>
5583 (R098)     <Reference
5584 (R099)       URI="#uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5585 (R100)       <Transforms>
5586 (R101)         <Transform Algorithm=
5587 (R102)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5588 (R103)         </Transforms>
5589 (R104)         <DigestMethod Algorithm=
5590 (R105)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5591 (R106)         <DigestValue>dr0g6hycoc884i+BD8FYCJGbbbE=</DigestValue>
5592 (R107)       </Reference>
5593 (R108)     <Reference URI="#_1">
5594 (R109)       <Transforms>
5595 (R110)         <Transform Algorithm=
5596 (R111)           "http://www.w3.org/2001/10/xml-exc-c14n#" />
5597 (R112)         </Transforms>
5598 (R113)         <DigestMethod Algorithm=
5599 (R114)           "http://www.w3.org/2000/09/xmldsig#sha1" />
5600 (R115)         <DigestValue>Rv3N7VNfAqpn0khr3F/qQZmE/l4=</DigestValue>
5601 (R116)       </Reference>
5602 (R117)     <Reference URI="#_2">
5603 (R118)       <Transforms>

```

```

5604 (R0119)         <Transform Algorithm=
5605 (R0120)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
5606 (R0121)         </Transforms>
5607 (R0122)         <DigestMethod Algorithm=
5608 (R0123)         "http://www.w3.org/2000/09/xmldsig#sha1" />
5609 (R0124)         <DigestValue>X2pxEnYPM8cMLrbhNqPgs8xk+a4=</DigestValue>
5610 (R0125)         </Reference>
5611 (R0126)         </SignedInfo>
5612 (R0127)         <SignatureValue>I2jQuDTWWQiNJy/ziyg8AFYO/z4=</SignatureValue>
5613 (R0128)         <KeyInfo>
5614 (R0129)         <o:SecurityTokenReference>
5615 (R0130)         <o:Reference URI="#_0"/>
5616 (R0131)         </o:SecurityTokenReference>
5617 (R0132)         </KeyInfo>
5618 (R0133)         </Signature>
5619 (R0134)         </o:Security>
5620 (R0135)         </s:Header>
5621 (R0136)         <s:Body u:Id="_4">
5622 (R0137)         <e:EncryptedData Id="_5"
5623 (R0138)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5624 (R0139)         <e:EncryptionMethod Algorithm=
5625 (R0140)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
5626 (R0141)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5627 (R0142)         <o:SecurityTokenReference>
5628 (R0143)         <o:Reference URI="#_3"/>
5629 (R0144)         </o:SecurityTokenReference>
5630 (R0145)         </KeyInfo>
5631 (R0146)         <e:CipherData>
5632 (R0147)         <e:CipherValue>
5633 (R0148)         <!--base64 encoded octets of encrypted body content-->
5634 (R0149)         </e:CipherValue>
5635 (R0150)         </e:CipherData>
5636 (R0151)         </e:EncryptedData>
5637 (R0152)         </s:Body>
5638 (R0153)         </s:Envelope>

```

5639 The message above is a response from the Service to the Client using the tokens based on the
5640 saml:Assertion IssuedToken from the STS.

5641 Lines (R018)-(R0134) contain the WS-Security o:Security header in the SOAP s:Header.

5642 Lines (R023)-(R036) contain an **sc:DerivedKeyToken** that may be used to **construct the derived**
5643 **signing key DKT4(K2)**, which uses the Client-generated ephemeral key, K2, that the Service received in
5644 the Client request (M028)-(M045) above, and is now used in the response to the client, similar to the way
5645 the STS used K1 to respond to the Client, except that in this case the Service will use derived keys
5646 DKT4(K2) and DKT5(K2) in addition to K2 in the response. This sc:DerivedKeyToken contains a WS-
5647 Security o:SecurityTokenReference (R024)-(R032) that uses the WS-Security 1.1 mechanism
5648 EncryptedKeySHA1 to identify the Client-generated ephemeral key, K2, as the key to use to derive
5649 DKT4(K2) along with the sc:Nonce provided (R035) (and the label as described in
5650 [\[WS_SECURE_CONV\]](#)).

5651 Lines (R037)-(R048) contain a **second sc:DerivedKeyToken** that may be used by the Client to
5652 **construct the derived encryption key, DKT5(K2)**. The same EncryptedKeySHA1 mechanism is used
5653 by the Client to construct DKT5(K2) as described for DKT4(K2).

5654 Lines (R049)-(R051) contain an **e:ReferenceList** containing an e:DataReference to the EncryptedData
5655 element in the s:Body (R0136).

5656 Lines (R052)-(R055) contain 2 WS-Security 1.1 k:SignatureConfirmation elements confirming the dsig
5657 Signatures that were in the client request above (M0203) on the endorsing signature and (M0177) on the
5658 message signature of the Client request. The dsig SignatureValues compare respectively to assure the
5659 Client that the data from the Client request was processed and is what is being referred to in this
5660 response.

5661 Lines (R056)-(R0134) contain the message signature, which is signed as referenced in the dsig KeyInfo
5662 (R0128)-(R0132) by DKT4(K2), which may be constructed as described above using the
5663 sc:DerivedKeyToken element (R023)-(R036).

5664 Lines (R0137)-(R0151) contain the Service response payload to the Client which may be decrypted using
5665 DKT5(K2) using the sc:DerivedKeyToken element (R037)-(R048).

5666 2.4 Secure Conversation Scenarios

5667 2.4.1 (WSS 1.0) Secure Conversation bootstrapped by Mutual 5668 Authentication with X.509 Certificates

5669 This scenario corresponds to the situation where both parties have an X.509 certificate (and public/private
5670 key pair). Because of the volume of messages from each source, the Requestor/Initiator uses WS-
5671 SecureConversation to establish a new session key and uses the session key for integrity and/or
5672 confidentiality protection. This improves performance, by using less expensive symmetric key operations
5673 and improves security by reducing the exposure of the long term secret.

5674 The recipient models this scenario by using the symmetric binding that includes a
5675 SecureConversationToken assertion to describe the token type accepted by this endpoint. This token
5676 assertion further contains a bootstrap policy to indicate the security binding that is used by requestors that
5677 want a security context token issued by this service. The bootstrap policy affects the Request Security
5678 Token Request (RST) and Request Security Token Request Response (RSTR) messages sent between
5679 the Initiator and the Recipient to establish the security context. It is modeled by use of an asymmetric
5680 binding assertion for this scenario because both parties mutually authenticate each other using their
5681 X.509 certificates.

5682 The message level policies cover a different scope of the web service definition than the security binding
5683 level policy and so appear as separate policies and are attached at Message Policy Subject. These are
5684 shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint
5685 subject (WSS10SecureConversation_policy) and two (WSS10SecureConversation_input_policy,
5686 WSS10SecureConversation_output_policy) with message subjects to achieve this use case.

5687

```
5688 (P001) <wsp:Policy wsu:Id="WSS10SecureConversation_policy">  
5689 (P002)   <wsp:ExactlyOne>  
5690 (P003)     <wsp>All>  
5691 (P004)       <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-  
5692 securitypolicy/200702">  
5693 (P005)         <wsp:Policy>  
5694 (P006)           <sp:ProtectionToken>  
5695 (P007)             <wsp:Policy>  
5696 (P008)               <sp:SecureConversationToken  
5697 sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken  
5698 /AlwaysToRecipient">  
5699 (P009)                 <wsp:Policy>  
5700 (P010)                   <sp:RequireDerivedKeys/>  
5701 (P011)                   <sp:BootstrapPolicy>  
5702 (P012)                   <wsp:Policy>  
5703 (P013)                     <wsp:ExactlyOne>  
5704 (P014)                       <wsp>All>  
5705 (P015)                         <sp:AsymmetricBinding  
5706 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
5707 (P016)                           <wsp:Policy>  
5708 (P017)                             <sp:InitiatorToken>  
5709 (P018)                               <wsp:Policy>  
5710 (P019)                                 <sp:X509Token  
5711 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/  
5712 AlwaysToRecipient">  
5713 (P020)                                   <wsp:Policy>  
5714 (P021)                                     <sp:WssX509V3Token10/>  
5715 (P022)                                       </wsp:Policy>
```



```

5716 (P023) </sp:X509Token>
5717 (P024) </wsp:Policy>
5718 (P025) </sp:InitiatorToken>
5719 (P026) <sp:RecipientToken>
5720 (P027) <wsp:Policy>
5721 (P028) <sp:X509Token
5722 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
5723 AlwaysToInitiator">
5724 (P029) <wsp:Policy>
5725 (P030) <sp:WssX509V3Token10/>
5726 (P031) </wsp:Policy>
5727 (P032) </sp:X509Token>
5728 (P033) </wsp:Policy>
5729 (P034) </sp:RecipientToken>
5730 (P035) <sp:AlgorithmSuite>
5731 (P036) <wsp:Policy>
5732 (P037) <sp:TripleDesRsa15/>
5733 (P038) </wsp:Policy>
5734 (P039) </sp:AlgorithmSuite>
5735 (P040) <sp:Layout>
5736 (P041) <wsp:Policy>
5737 (P042) <sp:Strict/>
5738 (P043) </wsp:Policy>
5739 (P044) </sp:Layout>
5740 (P045) <sp:IncludeTimestamp/>
5741 (P046) <sp:OnlySignEntireHeadersAndBody/>
5742 (P047) </wsp:Policy>
5743 (P048) </sp:AsymmetricBinding>
5744 (P049) <sp:Wss10>
5745 (P050) <wsp:Policy>
5746 (P051) <sp:MustSupportRefKeyIdentifier/>
5747 (P052) </wsp:Policy>
5748 (P053) </sp:Wss10>
5749 (P054) <sp:SignedParts>
5750 (P055) <sp:Body/>
5751 (P056) <sp:Header Name="Action"
5752 Namespace="http://www.w3.org/2005/08/addressing" />
5753 (P057) </sp:SignedParts>
5754 (P058) <sp:EncryptedParts>
5755 (P059) <sp:Body/>
5756 (P060) </sp:EncryptedParts>
5757 (P061) </wsp:All>
5758 (P062) </wsp:ExactlyOne>
5759 (P063) </wsp:Policy>
5760 (P064) </sp:BootstrapPolicy>
5761 (P065) </wsp:Policy>
5762 (P066) </sp:SecureConversationToken>
5763 (P067) </wsp:Policy>
5764 (P068) </sp:ProtectionToken>
5765 (P069) <sp:AlgorithmSuite>
5766 (P070) <wsp:Policy>
5767 (P071) <sp:Basic256/>
5768 (P072) </wsp:Policy>
5769 (P073) </sp:AlgorithmSuite>
5770 (P074) <sp:Layout>
5771 (P075) <wsp:Policy>
5772 (P076) <sp:Strict/>
5773 (P077) </wsp:Policy>
5774 (P078) </sp:Layout>
5775 (P079) <sp:IncludeTimestamp/>
5776 (P080) <sp:OnlySignEntireHeadersAndBody/>
5777 (P081) </wsp:Policy>
5778 (P082) </sp:SymmetricBinding>
5779 (P083) <sp:Trust13>

```

```

5780 (P084) <wsp:Policy>
5781 (P085) <sp:RequireClientEntropy/>
5782 (P086) <sp:RequireServerEntropy/>
5783 (P087) </wsp:Policy>
5784 (P088) </sp:Trust13>
5785 (P089)
5786 (P090) </wsp:All>
5787 (P091) </wsp:ExactlyOne>
5788 (P092) </wsp:Policy>
5789
5790 (P093) <wsp:Policy wsu:Id="WSS10SecureConversation_input_policy">
5791 (P094) <wsp:ExactlyOne>
5792 (P095) <wsp:All>
5793 (P096) <sp:SignedParts>
5794 (P097) <sp:Header Name="Action"
5795 <sp:Header Name="To"
5796 <sp:Header Name="MessageID"
5797 <sp:Header Name="RelatesTo"
5798 <sp:Header Name="To"
5799 <sp:Header Name="MessageID"
5800 <sp:Header Name="RelatesTo"
5801 <sp:Header Name="To"
5802 <sp:Header Name="MessageID"
5803 <sp:Header Name="RelatesTo"
5804 <sp:Header Name="To"
5805 <sp:Header Name="MessageID"
5806 <sp:Header Name="RelatesTo"
5807 <sp:Header Name="To"
5808 <sp:Header Name="MessageID"
5809 <sp:Header Name="RelatesTo"
5810 <sp:Header Name="To"
5811 <sp:Header Name="MessageID"
5812 <sp:Header Name="RelatesTo"
5813 <sp:Header Name="To"
5814 <sp:Header Name="MessageID"
5815 <sp:Header Name="RelatesTo"
5816 <sp:Header Name="To"
5817 <sp:Header Name="MessageID"
5818 <sp:Header Name="RelatesTo"
5819 <sp:Header Name="To"
5820 <sp:Header Name="MessageID"
5821 <sp:Header Name="RelatesTo"
5822 <sp:Header Name="To"

```

5823 Lines (P004) – (P082) indicate that the service uses the symmetric security binding to protect messages,
5824 using a Security Context Token (Lines (P008) – (P066)) to sign messages in both directions. The actual
5825 basis for the signatures should be keys derived from that Security Context Token as stated by the
5826 RequireDerivedKey assertion in Line (P010). Messages must include a Timestamp element (Line (P079))
5827 and the signature value must be calculated over the entire body and header elements (Line (P080)).

5828 Lines (P083) – (P088) contain the Trust13 assertion which defines the requirements for the WS-Trust
5829 related message exchange as part of the SC bootstrap. Line (P085) indicates that the Initiator (Client) has
5830 to provide entropy to be used as key material for the requested proof token. Line (P086) requires the
5831 same from the recipient (Server) which results in computed key from both client and server entropy
5832 values.

5833 Lines (P011) – (P065) contain the bootstrap policy for the initial creation of the security context between
5834 the communication parties before it is being used. It contains an AsymmetricBinding assertion (Lines
5835 (P015) – (P048)) which indicates that the initiator's (WS-Security 1.0 X.509 Certificate) token (Lines
5836 (P017) – (P025)) used by the recipient to verify the signature in the RST must be included in the message
5837 (P019). The exchange of entropy and key material in the body of the RST and RSTR messages is
5838 protected by the EncryptedParts assertion of the bootstrap policy in lines (P058) – (P061).

5839 According to the MustSupportKeyRefIdentifier assertions in Line (P051), an X.509 Key Identifier must be
5840 used to identify the token. Lines (P054) – (P057) require that the body and the WS-Addressing Action
5841 header is signed on each message (RST, RSTR) within the bootstrap process.

5842 An example of an RST message sent from the initiator to the recipient according to the bootstrap policy
5843 defined by this policy is as follows:

```
5844 (M001) <?xml version="1.0" encoding="utf-8" ?>
5845 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
5846         xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="...">
5847 (M003)   <soap:Header>
5848 (M004)     <wsa:Action wsu:Id="action">
5849 (M005)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
5850 (M006)     </wsa:Action>
5851 (M007)     <wsse:Security>
5852 (M008)       <wsu:Timestamp wsu:Id="timestamp">
5853 (M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5854 (M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5855 (M011)       </wsu:Timestamp>
5856 (M012)       <wsse:BinarySecurityToken wsu:Id="clientToken"
5857             ValueType="...#X509v3" EncodingType="...#Base64Binary">
5858 (M013)         MIICZDCCAc2gAwIBAgIRALSOLzt7...
5859 (M014)       </wsse:BinarySecurityToken>
5860 (M015)       <ds:Signature xmlns:ds="...">
5861 (M016)         <ds:SignedInfo>
5862 (M017)           <ds:CanonicalizationMethod
5863             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5864 (M018)           <ds:SignatureMethod
5865             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
5866 (M019)           <ds:Reference URI="#action">
5867 (M020)             <ds:Transforms>
5868 (M021)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
5869                 exc-c14n#" />
5870 (M022)             </ds:Transforms>
5871 (M023)             <ds:DigestMethod
5872                 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
5873 (M024)             <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
5874 (M025)           </ds:Reference>
5875 (M026)           <ds:Reference URI="#timestamp">...</ds:Reference>
5876 (M027)           <ds:Reference URI="#body">...</ds:Reference>
5877 (M028)         </ds:SignedInfo>
5878 (M029)         <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
5879 (M030)       <ds:KeyInfo>
5880 (M031)         <wsse:SecurityTokenReference>
5881 (M032)           <wsse:Reference URI="#clientToken"
5882             ValueType="...#X509v3" />
5883 (M033)         </wsse:SecurityTokenReference>
5884 (M034)       </ds:KeyInfo>
5885 (M035)     </ds:Signature>
5886 (M036)     <xenc:EncryptedKey>
5887 (M037)       <xenc:EncryptionMethod Algorithm="...#rsa-1_5" />
5888 (M038)       <ds:KeyInfo>
5889 (M039)         <wsse:SecurityTokenReference>
5890 (M040)           <wsse:KeyIdentifier
5891             ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
5892 (M042)           </wsse:KeyIdentifier>
5893 (M043)         </wsse:SecurityTokenReference>
5894 (M044)       </ds:KeyInfo>
5895 (M045)       <xenc:CipherData>
5896 (M046)         <xenc:CipherValue>
5897 (M047)           <!-- encrypted key -->
5898 (M048)         </xenc:CipherValue>
5899 (M049)       </xenc:CipherData>
5900 (M050)     <xenc:ReferenceList>
5901 (M051)       <xenc:DataReference URI="#request" />
```

```

5902 (M052) </xenc:ReferenceList>
5903 (M053) </xenc:EncryptedKey>
5904 (M054) </wsse:Security>
5905 (M055) </soap:Header>
5906 (M056) <soap:Body wsu:Id="body">
5907 (M057) <xenc:EncryptedData Id="request" Type="...#Content">
5908 (M058) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
5909 (M059) <xenc:CipherData>
5910 (M060) <xenc:CipherValue>
5911 (M061) <!-- encrypted RST request -->
5912 (M062) <!-- ### Begin unencrypted RST request
5913 (M063) <wst:RequestSecurityToken>
5914 (M064) <wst:TokenType>
5915 (M065) http://docs.oasis-open.org/ws-sx/ws-secureconversation/
5916 200512/sct
5917 (M066) </wst:TokenType>
5918 (M067) <wst:RequestType>
5919 (M068) http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
5920 (M069) </wst:RequestType>
5921 (M070) <wsp:AppliesTo
5922 xmlns:wsp="http://www.w3.org/ns/ws-policy">
5923 (M071) <wsp:EndpointReference>
5924 (M072) <wsp:Address>
5925 (M073) http://acme.com/ping/
5926 (M074) </wsp:Address>
5927 (M075) </wsp:EndpointReference>
5928 (M076) </wsp:AppliesTo>
5929 (M077) <wst:Entropy>
5930 (M078) <wst:BinarySecret>cr9b0cblwCEyY9ehaGK33e41h9s=
5931 (M079) </wst:BinarySecret>
5932 (M080) </wst:Entropy>
5933 (M081) </wst:RequestSecurityToken>
5934 (M082) ### End unencrypted RST request -->
5935 (M083) </xenc:CipherValue>
5936 (M084) </xenc:CipherData>
5937 (M085) </xenc:EncryptedData>
5938 (M086) </soap:Body>
5939 (M087) </soap:Envelope>

```

- 5940 Line (M005) indicates to the recipient that the SCT Binding of WS-Trust is used.
- 5941 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 5942 Lines (M012) – (M014) contain the initiator’s X.509 token as required by the InitiatorToken assertion’s value (“.../AlwaysToRecipient”).
- 5943
- 5944 Lines (M015) – (M035) hold the message signature.
- 5945 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).
- 5946
- 5947
- 5948
- 5949 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 5950
- 5951 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.
- 5952
- 5953 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.
- 5954
- 5955 Lines (M056) – (M086) hold the Security Token Reference pointing to the initiators X.509 certificate.
- 5956 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).
- 5957
- 5958
- 5959

5960 An example of an RSTR message sent from the recipient to the initiator according to the bootstrap policy
 5961 defined by this policy is as follows:

```

5962 (M001) <?xml version="1.0" encoding="UTF-8"?>
5963 (M002) <soap:Envelope xmlns:soap="..." xmlns:wst="..." xmlns:wsc="..."
5964 (M003)   xmlns:xenc="...">
5965 (M004)   <soap:Header>
5966 (M005)     <wsa:Action wsu:Id="action">
5967 (M006)       http://docs.oasis-open.org/ws-sx/ws-
5968 (M007) trust/200512/RSTRC/IssueFinal
5969 (M008)     </wsa:Action>
5970 (M009)     <wsse:Security>
5971 (M010)       <wsu:Timestamp wsu:Id="timestamp">
5972 (M011)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5973 (M012)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5974 (M013)       </wsu:Timestamp>
5975 (M014)       <wsse:BinarySecurityToken wsu:Id="serviceToken"
5976 (M015)         ValueType="...#X509v3" EncodingType="...#Base64Binary">
5977 (M016)         MIIASDPiOASDsaöAgIRALSOLzt7...
5978 (M017)       </wsse:BinarySecurityToken>
5979 (M018)       <ds:Signature xmlns:ds="...">
5980 (M019)         <ds:SignedInfo>
5981 (M020)           <ds:CanonicalizationMethod
5982 (M021)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5983 (M022)           <ds:SignatureMethod
5984 (M023)             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
5985 (M024)           <ds:Reference URI="#action">
5986 (M025)             <ds:Transforms>
5987 (M026)               <ds:Transform
5988 (M027)                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5989 (M028)             </ds:Transforms>
5990 (M029)           <ds:DigestMethod
5991 (M030)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
5992 (M031)           <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
5993 (M032)         </ds:Reference>
5994 (M033)         <ds:Reference URI="#timestamp">...</ds:Reference>
5995 (M034)         <ds:Reference URI="#body">...</ds:Reference>
5996 (M035)       </ds:SignedInfo>
5997 (M036)       <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
5998 (M037)       <ds:KeyInfo>
5999 (M038)         <wsse:SecurityTokenReference>
6000 (M039)           <wsse:Reference URI="#myToken" ValueType="...#X509v3" />
6001 (M040)         </wsse:SecurityTokenReference>
6002 (M041)       </ds:KeyInfo>
6003 (M042)     </ds:Signature>
6004 (M043)     <xenc:EncryptedKey>
6005 (M044)       <xenc:EncryptionMethod Algorithm="...#rsa-1_5" />
6006 (M045)       <ds:KeyInfo>
6007 (M046)         <wsse:SecurityTokenReference>
6008 (M047)           <wsse:KeyIdentifier
6009 (M048)             ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
6010 (M049)           </wsse:KeyIdentifier>
6011 (M050)         </wsse:SecurityTokenReference>
6012 (M051)       </ds:KeyInfo>
6013 (M052)       <xenc:CipherData>
6014 (M053)         <xenc:CipherValue>
6015 (M054)           <!-- encrypted key -->
6016 (M055)         </xenc:CipherValue>
6017 (M056)       </xenc:CipherData>
6018 (M057)       <xenc:ReferenceList>
6019 (M058)         <xenc:DataReference URI="#response" />
6020 (M059)       </xenc:ReferenceList>
6021 (M060)     </xenc:EncryptedKey>
6022 (M061)   </wsse:Security>
6023 (M062) </soap:Header>
  
```

```

6024      (M056)      <soap:Body wsu:Id="body">
6025      (M057)      <xenc:EncryptedData Id="response" Type="...#Content">
6026      (M058)      <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
6027      (M059)      <xenc:CipherData>
6028      (M060)      <xenc:CipherValue>
6029      (M061)      <!-- encrypted RSTR -->
6030      (M062)      <!-- ### Begin unencrypted RSTR
6031      (M063)      <wst:RequestSecurityTokenResponseCollection>
6032      (M064)      <wst:RequestSecurityTokenResponse>
6033      (M065)      <wst:RequestedSecurityToken>
6034      (M066)      <wsc:SecurityContextToken>
6035      (M067)      <wsc:Identifier>uuid:...</wsc:Identifier>
6036      (M068)      </wsc:SecurityContextToken>
6037      (M069)      </wst:RequestedSecurityToken>
6038      (M070)      <wst:RequestedProofToken>
6039      (M071)      <xenc:EncryptedKey Id="ek049ea390c90011dbba4e00304852867e">
6040      (M072)      <xenc:EncryptionMethod
6041      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
6042      (M073)      <ds:KeyInfo>
6043      (M074)      <wsse:SecurityTokenReference>
6044      (M075)      <wsse:KeyIdentifier
6045      ValueType="...#X509SubjectKeyIdentifier"
6046      EncodingType="...#Base64Binary">
6047      (M076)      Wjw2gDCBye6NJAh0lPCyUldvTN8=
6048      (M077)      </wsse:KeyIdentifier>
6049      (M078)      </wsse:SecurityTokenReference>
6050      (M079)      </ds:KeyInfo>
6051      (M080)      <xenc:CipherData>
6052      (M081)      <xenc:CipherValue>hhx9TcaVL6XwBN1...</xenc:CipherValue>
6053      (M082)      </xenc:CipherData>
6054      (M083)      </xenc:EncryptedKey>
6055      (M084)      </wst:RequestedProofToken>
6056      (M085)      <wsp:AppliesTo
6057      xmlns:wsp="http://www.w3.org/ns/ws-policy">
6058      (M086)      <wsp:EndpointReference>
6059      (M087)      <wsp:Address>
6060      (M088)      http://acme.com/ping/
6061      (M089)      </wsp:Address>
6062      (M090)      </wsp:EndpointReference>
6063      (M091)      </wsp:AppliesTo>
6064      (M092)      <wst:Entropy>
6065      (M093)      <wst:BinarySecret>asdhjwkjqwe123498SDFALasd=
6066      (M094)      </wst:BinarySecret>
6067      (M095)      </wst:Entropy>
6068      (M096)      </wst:RequestSecurityTokenResponse>
6069      (M097)      </wst:RequestSecurityTokenResponseCollection>
6070      (M098)      ### End unencrypted RSTR -->
6071      (M099)      </xenc:CipherValue>
6072      (M100)      </xenc:CipherData>
6073      (M101)      </xenc:EncryptedData>
6074      (M102)      </soap:Body>
6075      (M103)      </soap:Envelope>

```

- 6076 Line (M005) indicates to the initiator that this is the final response to the RST in an RSTRC.
- 6077 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 6078 Lines (M012) – (M014) contain the recipient's X.509 token as required by the RecipientToken assertion's value (".../AlwaysToInitiator").
- 6079
- 6080 Lines (M015) – (M035) hold the message signature.
- 6081 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.
- 6082
- 6083 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 6084

6085 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp
6086 assertion definition.

6087 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts
6088 assertion.

6089 Lines (M031) – (M033) hold the Security Token Reference pointing to the requestor’s X.509 certificate.

6090 Lines (M056) – (M102) contain the SOAP Body of the message with the encrypted RSTR collection
6091 element. Commented out is the unencrypted form of the RSTR collection in Lines (M063) – (M097), which
6092 includes one RSTR (Lines (M064) – (M096)). Lines (M065) – (M069) contain the new Security Context
6093 Token. The accompanying RequestedProofToken in Lines (M070) – (M084) include the encrypted secret
6094 key (Lines (M071) – (M083)) of the security context. The key is encrypted using the initiator’s public key
6095 that was already used to verify its signature in the incoming request.

6096 According to the Trust13 assertion, the response includes entropy provided by the recipient as indicated
6097 by Lines (M092) – (M095).

6098 An Example of an SCT-secured application message sent from the initiator to the recipient according to
6099 the message input policy (WSS10SecureConversation_input_policy) is as follows:

```

6100 (M001) <?xml version="1.0" encoding="UTF-8"?>
6101 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
6102         xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="..." xmlns:wsm="...">
6103 (M003) <soap:Header>
6104 (M004) <wsa:To wsu:Id="to">http://acme.com/ping/</wsa:To>
6105 (M005) <wsa:MessageID wsu:Id="msgid">
6106 (M006)     http://acme.com/guid/12318731edh-CA47-1067-B31D-10662DA
6107 (M007) </wsa:MessageID>
6108 (M008) <wsa:Action wsu:Id="action">urn:Ping</wsa:Action>
6109 (M009) <wsse:Security>
6110 (M010)     <wsu:Timestamp wsu:Id="timestamp">
6111 (M011)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6112 (M012)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6113 (M013)     </wsu:Timestamp>
6114 (M014)     <wsc:SecurityContextToken wsu:Id="SCT">
6115 (M015)         <wsc:Identifier>uuid:91f50600-60cc-11da-8e67-000000000000
6116 (M016)         </wsc:Identifier>
6117 (M017)     </wsc:SecurityContextToken>
6118 (M018)     <wsc:DerivedKeyToken wsu:Id="DKT">
6119 (M019)         <wsse:SecurityTokenReference>
6120 (M020)             <wsse:Reference URI="#SCT" ValueType="http://docs.oasis-
6121                 open.org/ws-sx/ws-secureconversation/200512/sct"/>
6122 (M021)         </wsse:SecurityTokenReference>
6123 (M022)         <wsc:Offset>0</wsc:Offset>
6124 (M023)         <wsc:Length>24</wsc:Length>
6125 (M024)         <wsc:Label>
6126 (M025)             WSSecure ConversationWSSecure Conversation
6127 (M026)         </wsc:Label>
6128 (M027)         <wsc:Nonce>yln04kFBJesy2U2SQL6ezI3SCak=</wsc:Nonce>
6129 (M028)     </wsc:DerivedKeyToken>
6130 (M029)     <ds:Signature xmlns:ds="...">
6131 (M030)         <ds:SignedInfo>
6132 (M031)             <ds:CanonicalizationMethod
6133                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6134 (M032)             <ds:SignatureMethod
6135                 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6136 (M033)             <ds:Reference URI="#msgid">
6137 (M034)                 <ds:Transforms>
6138 (M035)                     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
6139                         exc-c14n#"/>
6140 (M036)                 </ds:Transforms>
6141 (M037)                 <ds:DigestMethod
6142                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6143 (M038)                 <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6144 (M039)             </ds:Reference>

```

```

6145 (M039) <ds:Reference URI="#to">...</ds:Reference>
6146 (M040) <ds:Reference URI="#action">...</ds:Reference>
6147 (M041) <ds:Reference URI="#timestamp">...</ds:Reference>
6148 (M042) <ds:Reference URI="#body">...</ds:Reference>
6149 (M043) </ds:SignedInfo>
6150 (M044) <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6151 (M045) <ds:KeyInfo>
6152 (M046) <wsse:SecurityTokenReference>
6153 (M047) <wsse:Reference URI="#DKT" ValueType="http://docs.oasis-
6154 open.org/ws-sx/ws-secureconversation/200512/dk"/>
6155 (M048) </wsse:SecurityTokenReference>
6156 (M049) </ds:KeyInfo>
6157 (M050) </ds:Signature>
6158 (M051) </wsse:Security>
6159 (M052) </soap:Header>
6160 (M053) <soap:Body wsu:Id="body">
6161 (M054) <pns:Ping xmlns:pns="http://tempuri.org/">
6162 <pns:Text>abc</pns:Text>
6163 </pns:Ping>
6164 </soap:Body>
6165 </soap:Envelope>

```

6166

6167 Lines (M004) – (M008) contain the WS-Addressing headers according to the UsingAddressing assertion.

6168 Lines (M010) – (M013) hold the Timestamp as specified by the IncludeTimestamp assertion within the
6169 SymmetricBinding assertion.

6170 Lines (M014) – (M017) contain the Security Context Token which has been issued by the recipient in the
6171 previous message. Based on this token, the initiator included a Derived Key Token (Lines (M018) –
6172 (M027)) as indicated by the RequireDerivedKey assertion in the policy.

6173 Lines (M028) – (M050) hold the message signature that uses the Derived Key Token (Lines (M046) –
6174 (M048)) to sign the WS-Addressing headers (Lines (M032) – (M040)), the timestamp (Line (M041)) and
6175 the body (Line (M042)) of the message, according to the SignedParts assertion of the message input
6176 policy (WSS10SecureConversation_input_policy).

6177 **3 Conformance**

6178 This document contains non-normative examples of usage of WS-SecurityPolicy and other related
6179 specifications.

6180 Therefore **there are no conformance statements that apply to this document.**

6181 Refer to the referenced specifications [[References](#)], which will individually contain conformance
6182 requirements for WS-SecurityPolicy and related specifications.

6183

6184 **A. Acknowledgements**

6185 The following individuals have participated in the creation of this specification and are gratefully
6186 acknowledged:

6187 **Participants:**

6188 Ashok Malhotra, Oracle
6189 Prateek Mishra, Oracle
6190 Ramana Turlapati, Oracle
6191 Martin Gudgin, Microsoft
6192 Marc Goodner, Microsoft
6193 Paul Cotton, Microsoft
6194 Hal Lockhart, BEA
6195 Symon Chang, BEA
6196 Frederick Hirsch, Nokia
6197 Greg Whitehead, Hewlett-Packard
6198 Ching-Yun (C.Y.) Chao, IBM
6199 Anthony Nadalin, IBM
6200 Tony Gullotta, SOA

6201

B. Revision History

6202 [optional; should not be included in OASIS Standards]

6203

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

6204