

# WS-SecurityPolicy Examples

## Committee Draft 04

13 October 2010

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-04.doc>  
(Authoritative)  
<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-04.pdf>  
<http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples-cd-04.html>

#### Previous Version:

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

#### Latest Version:

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

### Technical Committee:

OASIS Web Services Secure Exchange TC

### Chair(s):

Kelvin Lawrence, IBM  
Chris Kaler, Microsoft

### Editor(s):

Rich Levinson, Oracle Corporation  
Tony Gullotta, SOA Software Inc.  
Symon Chang, Oracle Corporation.  
Martin Raeppe, SAP AG

### Related work:

N/A

### Declared XML Namespace(s):

N/A

### Abstract:

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

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

in order to demonstrate how different options and security bindings impact the structure of the policies.

**Status:**

This document was last revised or approved by the WS-SX TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

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

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

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

---

## Notices

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

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

---

# Table of Contents

1	Introduction.....	6
1.1	Terminology and Concepts.....	6
1.1.1	Actors .....	7
1.1.2	Concepts .....	10
1.1.2.1	X509 Certificates .....	10
1.2	Terminology.....	11
1.3	Normative References .....	11
1.4	Non-Normative References .....	12
1.5	Specifications.....	13
1.6	Interops and Sample Messages .....	13
2	Scenarios .....	14
2.1	UsernameToken .....	14
2.1.1	UsernameToken – no security binding .....	14
2.1.1.1	UsernameToken with plain text password .....	14
2.1.1.2	UsernameToken without password .....	15
2.1.1.3	UsernameToken with timestamp, nonce and password hash .....	16
2.1.2	Use of SSL Transport Binding .....	17
2.1.2.1	UsernameToken as supporting token.....	17
2.1.3	(WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, Encrypt .....	19
2.1.3.1	(WSS 1.0) Encrypted UsernameToken with X.509v3.....	22
2.1.4	(WSS 1.1), User Name with Certificates, Sign, Encrypt .....	25
2.2	X.509 Token Authentication Scenario Assertions .....	29
2.2.1	(WSS1.0) X.509 Certificates, Sign, Encrypt .....	29
2.2.2	(WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt.....	32
2.2.2.1	(WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption .....	35
2.2.3	(WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt.....	39
2.2.4	(WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt.....	42
2.3	SAML Token Authentication Scenario Assertions.....	48
2.3.1	WSS 1.0 SAML Token Scenarios.....	49
2.3.1.1	(WSS1.0) SAML1.1 Assertion (Bearer) .....	49
2.3.1.2	(WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL.....	51
2.3.1.3	(WSS1.0) SAML1.1 Assertion (HK) over SSL.....	53
2.3.1.4	(WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional Encrypt .....	54
2.3.1.5	(WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt .....	60
2.3.2	WSS 1.1 SAML Token Scenarios.....	65
2.3.2.1	(WSS1.1) SAML 2.0 Bearer.....	65
2.3.2.2	(WSS1.1) SAML2.0 Sender Vouches over SSL .....	69
2.3.2.3	(WSS1.1) SAML2.0 HoK over SSL .....	71
2.3.2.4	(WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign, Encrypt.....	75
2.3.2.5	(WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt .....	80
2.4	Secure Conversation Scenarios .....	105
2.4.1	(WSS 1.0) Secure Conversation bootstrapped by Mutual Authentication with X.509 Certificates .....	105

3	Conformance.....	114
A.	Acknowledgements .....	115
B.	Revision History .....	118

---

# 1 Introduction

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

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

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

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

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

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

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

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

## 1.1 Terminology and Concepts

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

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

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

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

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

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

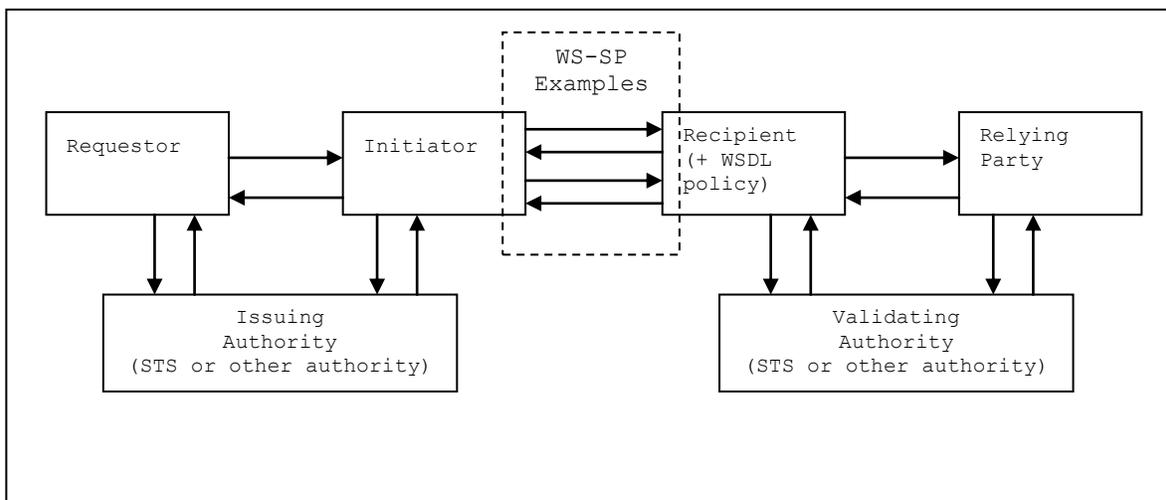
47 The reason for providing this context is to help interested readers understand that while the technology  
48 may be securing the integrity and confidentiality of the messages, there are additional questions about  
49 transactions such as who is liable for any commitments resulting from the transaction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

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

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

### 64 1.1.1 Actors

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



67  
68  
69

**Figure 1.**

70

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- 120
- 121 • **Requestor:** the person or entity requesting the service and who will be supplying credentials  
122 issued by the IssuingAuthority and will be validated by a ValidatingAuthority. The Requestor is the  
123 logical entity that supplies the credentials that ultimately get passed to the Recipient that will be  
124 trusted by the RelyingParty if they are validated by the ValidatingAuthority. (Note: the logistics of  
125 supplying the trusted credential is distinct from the logistics of packaging up the credentials within  
126 a WS-Security header in a SOAP message and transmitting that message to the Recipient. The  
127 latter logistics are covered by the Initiator, described below. The Requestor and Initiator may be  
128 combined into a single physical entity and this is a common occurrence, however they do not  
129 have to be and it is also a common occurrence that they are separate physical entities. The latter  
130 case is typified by the Requestor being a user at a browser that may be prompted for credentials  
131 by the Initiator on a server using HTTP to challenge the Requestor for a username and  
132 password.)

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

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

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

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

## 184 1.1.2 Concepts

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

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

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

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

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

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

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

225

### 226 1.1.2.1 X509 Certificates

227 The specifics of whom is trusted for X509 certificates depends on the specific organization's PKI (Public  
228 Key Infrastructure) setup. For this document, we shall assume the Subject of the X509 certificate  
229 identifies the actor, which may be the IssuingAuthority, or the Initiator, or the Requestor, depending on  
230 the example. We shall also assume that the issuer of the X509 certificates is a general Certificate  
231 Authority not directly involved in any authorization of the web service transactions, but is relied on for the  
232 validity of the X509 certificate in a manner out of scope of the scenarios covered. In addition, this  
233 document does not explicitly address the case of X509 certificates issued by the IssuingAuthority actor.  
234 Such use cases are generally implicitly covered if one assumes that such relations are automatically  
235 covered by the specifics of the organization PKI setups.

236 However, the IssuingAuthority may issue tokens, such as SAML holder-of-key that contain X509  
237 certificates. In these cases, the basis of trust is that the X509 Certificate of the IssuingAuthority was used  
238 to protect the X509 certificate of the Requestor which is contained in the signed SAML holder-of-key  
239 token. I.e. any “chaining” of tokens is done by referencing those tokens within signed XML structures and  
240 not by issuing actual X509 certificates

## 241 1.2 Terminology

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

## 245 1.3 Normative References

- 246 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
247 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 248 [WSS10-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.0 (WS-  
249 Security 2004)”, March 2004.  
250 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-  
251 security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 252 [WSS11-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.1”,  
253 OASIS Standard incorporating Approved Errata, 01 November 2006.  
254 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-  
255 SOAPMessageSecurity.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.pdf)
- 256 [WSS10-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.0”,  
257 March 2004.  
258 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-  
259 profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
- 260 [WSS11-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.1”,  
261 February 2006.  
262 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf>
- 263 [WSS10-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token  
264 Profile 1.0”, March 2004.  
265 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-  
266 1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
- 267 [WSS11-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token  
268 Profile 1.1”, OASIS Standard incorporating Approved Errata, 01 November 2006.  
269 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-  
270 x509TokenProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-x509TokenProfile.pdf)
- 271 [WSS10-SAML11-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.0”,  
272 December 2004.  
273 <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>
- 274 [WSS11-SAML1120-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.1”,  
275 OASIS Standard Incorporating Approved Errata, 1 November 2006.  
276 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-  
277 SAMLTOKENProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SAMLTOKENProfile.pdf)
- 278 [WSS11-LIBERTY-SAML20-PROFILE]  
279 [http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-  
280 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)
- 281 [WSS-TC-ALL-TOKEN-PROFILES] OASIS WS-Security TC links to all other WSS 1.0 and WSS 1.1  
282 token profiles  
283 <http://www.oasis-open.org/specs/index.php#wssv1.0>  
284 <http://www.oasis-open.org/specs/index.php#wssv1.1>  
285 and other documents (interop) in the TC repository:  
286 [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss)

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

## 310 1.4 Non-Normative References

311 WSS10-INTEROP-01: OASIS Working Draft 06, "Web Services Security: Interop 1 Scenarios",  
312 Jun 2003.  
313 [http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-](http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf)  
314 [06-merged-changes.pdf](http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf)  
315  
316 WSS10-INTEROP-02: OASIS Working Draft 06, "Web Services Security: Interop 2 Scenarios",  
317 Oct 2003.  
318 [http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-](http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc)  
319 [06-merged.doc](http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc)  
320  
321 WSS11-INTEROP-01: OASIS Working Draft 02, "Web Services Security 1.1: Interop Scenario",  
322 June 2005. [http://www.oasis-open.org/committees/download.php/12997/wss-11-](http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc)  
323 [interop-draft-01.doc](http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc)  
324  
325 WSS10-KERBEROS-INTEROP: OASIS Working Draft 02, "Web Services Security: Kerberos Token  
326 Profile Interop Scenarios", Jan 2005.  
327 [http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-](http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-interop.doc)  
328 [interop.doc](http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-interop.doc)  
329  
330 WSS10-SAML11-INTEROP: OASIS Working Draft 12, "Web Services Security: SAML Interop 1  
331 Scenarios", July 2004.  
332 [http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-](http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc)  
333 [draft-12.doc](http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc)  
334

335 WSS11-SAML1120-INTEROP: OASIS Working Draft 4, "Web Services Security: SAML 2.0 Interop  
336 Scenarios", January 2005.  
337 [http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-  
draft-v4.doc](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-<br/>338 draft-v4.doc)

339  
340 WSSX-PRE-INTEROP: OASIS Contribution Version 1.0d, "WS-SecureConversation and WS-Trust  
341 Interop Scenarios", September 29, 2004.  
342 [http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-  
examples.html](http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-<br/>343 examples.html)

344  
345 WSSX-WSTR-WSSC-INTEROP: OASIS White Paper Version ED-10, "WS-SX Interop Scenarios - Phase  
346 2 Scenarios for demonstration of WS-SX TC specifications", October 31, 2006.  
347 [http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-  
10.doc](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-<br/>348 10.doc)

349  
350 WS-SECURE-INTEROP: OASIS White Paper Version ED-01, "Examples of Secure Web Service  
351 Message Exchange", July 11, 2008.  
352 [http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-  
message-examples.doc](http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-<br/>353 message-examples.doc)

354  
355 WSI-SCM-SAMPLEAPPL: <http://java.sun.com/webservices/docs/1.4/wsi-sampleapp/index.html> (login  
356 required)

357  
358

## 359 **1.5 Specifications**

## 360 **1.6 Interops and Sample Messages**

---

## 361 2 Scenarios

### 362 2.1 UsernameToken

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

#### 365 2.1.1 UsernameToken – no security binding

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

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

##### 372 2.1.1.1 UsernameToken with plain text password

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

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

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

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

384

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

393

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

395

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

```

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

```

417

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

424

### 425 2.1.1.2 UsernameToken without password

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

432

```

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

```

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

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

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

449

```

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

```

```

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

```

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

469

### 470 2.1.1.3 UsernameToken with timestamp, nonce and password hash

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

476

```

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

```

489

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

491

```

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

```

514

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

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

## 520 2.1.2 Use of SSL Transport Binding

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

527

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

### 550 2.1.2.1 UsernameToken as supporting token

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

554

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

```

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

```

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

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

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

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

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

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

```

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

```

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

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

622

623

### 624 2.1.3 (WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, 625 Encrypt

626

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

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

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

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

640 The policy is as follows:

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

837

### 838 2.1.3.1 (WSS 1.0) Encrypted UsernameToken with X.509v3

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

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

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

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

849 The policy is as follows:

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

```

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

```

```

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

```

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

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

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

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

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

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

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

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

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

950

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

```

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

```

```

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

```

998 Line (M020) is an encryption data reference that references the encrypted UsernameToken on lines  
999 (M023) – (M028) which was required to be included by the EncryptedSupportingTokens assertion. Lines  
1000 (M009) – (M014) hold a KeyIdentifier of the recipient’s token used to encrypt the UsernameToken as  
1001 required by the AsymmetricBinding assertion. Because the InitiatorEncryptionAssertion disallowed the  
1002 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included  
1003 token.

1004 Line (M019) is an encryption data reference that references the encrypted body of the message on lines  
1005 (M035) – (M040). The encryption was required by the EncryptedParts assertion of the input message  
1006 policy. It also uses the recipient token as identified by the KeyIdentifier.

1007 Lines (M029) – (M031) contain a timestamp for the message as required by the IncludeTimestamp  
1008 assertion.

1009 Because the username token was encrypted its content prior to encryption is included below to better  
1010 illustrate the reference.

```

1011 (M043) <wsse:UsernameToken wsu:Id="usernameToken">
1012 (M044) <wsse:Username>Chris</wsse:Username>
1013 (M045) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1014 wss-username-token-profile-1.0#PasswordDigest">oY...=</wsse:Password>
1015 (M046) <wsse:Nonce EncodingType="...#Base64Binary">pN...=</wsse:Nonce>
1016 (M047) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
1017 (M048) </wsse:UsernameToken>

```

1018 Line (M046) contains the Nonce element and Line (M047) contains a timestamp. It is recommended that  
1019 these two elements should also be included in the PasswordText case for better security  
1020 [\[WSS10-USERNAME\]](#).

1021

## 1022 2.1.4 (WSS 1.1), User Name with Certificates, Sign, Encrypt

1023 This scenario is based on the “Examples of Secure Web Service Message Exchange Document”  
1024 [\[WS-SECURE-INTEROP\]](#).

1025 The use case here is the following: the Initiator generates a symmetric key; the symmetric key is  
1026 encrypted using the Recipient’s certificate and placed in an encrypted key element. The UsernameToken  
1027 identifying the Requestor and message body are signed using the symmetric key. The body and

1028 UsernameToken are also encrypted. The Authority for this request is generally the Subject of the  
1029 Initiator's X509 certificate.

1030 We can use the symmetric security binding [WSSP] with X509token as the protection token to illustrate  
1031 this case. If derived keys are to be used, then the derived keys property of X509Token should be set.

1032 The policy is as follows:

```
1033 (P001) <wsp:Policy wsu:Id="WSS11UsernameWithCertificates_policy">
1034 (P002)   <wsp:ExactlyOne>
1035 (P003)   <wsp>All>
1036 (P004)     <sp:SymmetricBinding>
1037 (P005)       <wsp:Policy>
1038 (P006)         <sp:ProtectionToken>
1039 (P007)           <wsp:Policy>
1040 (P008)             <sp:X509Token sp:IncludeToken="http://docs.oasis-
1041 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
1042 (P009)               <wsp:Policy>
1043 (P010)                 <sp:RequireThumbprintReference/>
1044 (P011)                 <sp:WssX509V3Token11/>
1045 (P012)                 </wsp:Policy>
1046 (P013)                 </sp:X509Token>
1047 (P014)                 </wsp:Policy>
1048 (P015)             </sp:ProtectionToken>
1049 (P016)           <sp:AlgorithmSuite>
1050 (P017)             <wsp:Policy>
1051 (P018)               <sp:Basic256/>
1052 (P019)               </wsp:Policy>
1053 (P020)             </sp:AlgorithmSuite>
1054 (P021)           <sp:Layout>
1055 (P022)             <wsp:Policy>
1056 (P023)               <sp:Strict/>
1057 (P024)               </wsp:Policy>
1058 (P025)             </sp:Layout>
1059 (P026)           <sp:IncludeTimestamp/>
1060 (P027)           <sp:OnlySignEntireHeadersAndBody/>
1061 (P028)           </wsp:Policy>
1062 (P029)         </sp:SymmetricBinding>
1063 (P030)       <sp:SignedEncryptedSupportingTokens>
1064 (P031)         <wsp:Policy>
1065 (P032)           <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
1066 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1067 (P033)             <wsp:Policy>
1068 (P034)               <sp:WssUsernameToken11/>
1069 (P035)               </wsp:Policy>
1070 (P036)             </sp:UsernameToken>
1071 (P037)             </wsp:Policy>
1072 (P038)           </sp:SignedEncryptedSupportingTokens>
1073 (P039)         <sp:Wss11>
1074 (P040)           <wsp:Policy>
1075 (P041)             <sp:MustSupportRefKeyIdentifier/>
1076 (P042)             <sp:MustSupportRefIssuerSerial/>
1077 (P043)             <sp:MustSupportRefThumbprint/>
1078 (P044)             <sp:MustSupportRefEncryptedKey/>
1079 (P045)             </wsp:Policy>
1080 (P046)           </sp:Wss11>
1081 (P047)         </wsp>All>
1082 (P048)       </wsp:ExactlyOne>
1083 (P049)     </wsp:Policy>
1084
1085 (P050) <wsp:Policy wsu:Id="UsernameForCertificates_input_policy">
1086 (P051)   <wsp:ExactlyOne>
1087 (P052)   <wsp>All>
1088 (P053)     <sp:SignedParts>
1089 (P054)       <sp:Body/>
1090 (P055)     </sp:SignedParts>
```

```

1091 (P056) <sp:EncryptedParts>
1092 (P057) <sp:Body/>
1093 (P058) </sp:EncryptedParts>
1094 (P059) </wsp:All>
1095 (P060) </wsp:ExactlyOne>
1096 (P061) </wsp:Policy>
1097
1098 (P062) <wsp:Policy wsu:Id="UsernameForCertificate_output_policy">
1099 (P063) <wsp:ExactlyOne>
1100 (P064) <wsp:All>
1101 (P065) <sp:SignedParts>
1102 (P066) <sp:Body/>
1103 (P067) </sp:SignedParts>
1104 (P068) <sp:EncryptedParts>
1105 (P069) <sp:Body/>
1106 (P070) </sp:EncryptedParts>
1107 (P071) </wsp:All>
1108 (P072) </wsp:ExactlyOne>
1109 (P073) </wsp:Policy>

```

1110 Lines (P004) – (P297) contain a SymmetricBinding assertion which indicates the use of one token to both  
1111 sign and encrypt a message.

1112 Lines (P006) – (P015) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P012)  
1113 indicate that the protection token must be an X.509 token that must never be included in any messages in  
1114 the message exchange. Line (P010) dictates the X.509 token must be an X.509v3 security token as  
1115 described in the WS-Security 1.1 X.509 Token Profile. Line (P011) dicates a thumbprint reference must  
1116 be used to identify the token in any message.

1117 Line (P026) requires the inclusion of a timestamp.

1118 Lines (P030) – (P038) contain a SignedEncryptedSupportingTokens assertion which identifies the  
1119 inclusion of an additional token which must be included in the message signature and encrypted. Lines  
1120 (P032) – (P036) indicate that the supporting token must be a UsernameToken and must be included in all  
1121 messages to the recipient. Line (P034) dictates the UsernameToken must conform to the WS-Security 1.1  
1122 UsernameToken Profile.

1123 Lines (P040) – (P046) contain some WS-Security 1.1 related interoperability requirements, specifically  
1124 support for key identifier, issuer serial number, thumbprint, and encrypted key references.

1125 Lines (P050) – (P061) contain a policy that is attached to the input message. Lines (P053) – (P055)  
1126 require that the body of the input message must be signed. Lines (P056) – (P058) require the body of the  
1127 input message must be encrypted.

1128 Lines (P062) – (P073) contain a policy that is attached to the output message. Lines (P065) – (P067)  
1129 require that the body of the output message must be signed. Lines (P068) – (P070) require the body of  
1130 the output message must be encrypted.

1131 An example of an input message that conforms to the above stated policy is as follows.

```

1132 (M001) <?xml version="1.0" encoding="utf-8" ?>
1133 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1134 (M003) <soap:Header>
1135 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..."
1136 xmlns:wsu="...">
1137 (M005) <xenc:EncryptedKey wsu:Id="EK">
1138 (M006) <xenc:EncryptionMethod
1139 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
1140 (M007) <ds:KeyInfo>
1141 (M008) <wsse:SecurityTokenReference>
1142 (M009) <wsse:KeyIdentifier ValueType="http://docs.oasis-
1143 open.org/wss/oasis-wss-soap-message-security-1.1#ThumbPrintSHA1">
1144 (M010) LKiQ/CmFrJDJqCLFcjlhIsmZ/+0=
1145 (M011) </wsse:KeyIdentifier>
1146 (M012) </wsse:SecurityTokenReference>
1147 (M013) </ds:KeyInfo>
1148 (M014) </xenc:EncryptedKey>

```

```

1149 (M015) <xenc:ReferenceList>
1150 (M016) <xenc:DataReference URI="#encUT"/>
1151 (M017) <xenc:DataReference URI="#encBody"/>
1152 (M018) </xenc:ReferenceList>
1153 (M019) <wsu:Timestamp wsu:Id="T0">
1154 (M020) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1155 (M021) </wsu:Timestamp>
1156 (M022) <xenc:EncryptedData wsu:Id="encUT">
1157 (M023) <xenc:EncryptionMethod
1158 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1159 (M024) <ds:KeyInfo>
1160 (M025) <wsse:SecurityTokenReference>
1161 (M026) <wsse:Reference URI="#EK"/>
1162 (M027) </wsse:SecurityTokenReference>
1163 (M028) </ds:KeyInfo>
1164 (M029) <xenc:CipherData>
1165 (M030) <xenc:CipherValue>...</xenc:CipherValue>
1166 (M031) </xenc:CipherData>
1167 (M032) </xenc:EncryptedData>
1168 (M033) <ds:Signature>
1169 (M034) <ds:SignedInfo>...
1170 (M035) <ds:Reference URI="#T0">...</ds:Reference>
1171 (M036) <ds:Reference URI="#usernameToken">...</ds:Reference>
1172 (M037) <ds:Reference URI="#body">...</ds:Reference>
1173 (M038) </ds:SignedInfo>
1174 (M039) <ds:SignatureValue>HFLP...</ds:SignatureValue>
1175 (M040) <ds:KeyInfo>
1176 (M041) <wsse:SecurityTokenReference>
1177 (M042) <wsse:Reference URI="#EK"/>
1178 (M043) </wsse:SecurityTokenReference>
1179 (M044) </ds:KeyInfo>
1180 (M045) </ds:Signature>
1181 (M046) </wsse:Security>
1182 (M047) </soap:Header>
1183 (M048) <soap:Body wsu:Id="body">
1184 (M049) <xenc:EncryptedData wsu:Id="encBody">
1185 (M050) <xenc:EncryptionMethod
1186 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1187 (M051) <ds:KeyInfo>
1188 (M052) <wsse:SecurityTokenReference>
1189 (M053) <wsse:Reference URI="#EK"/>
1190 (M054) </wsse:SecurityTokenReference>
1191 (M055) </ds:KeyInfo>
1192 (M056) <xenc:CipherData>
1193 (M057) <xenc:CipherValue>...</xenc:CipherValue>
1194 (M058) </xenc:CipherData>
1195 (M059) </xenc:EncryptedData>
1196 (M060) </soap:Body>
1197 (M061) </soap:Envelope>

```

1198 Lines (M005) – (M014) contain the encrypted symmetric key as required by the use of the  
1199 SymmetricBinding assertion starting on line (P004) with an X509Token ProtectionToken assertion. Line  
1200 (M006) references the KwRsaOaep Asymmetric Key Wrap algorithm dictated by the Basic 256 Algorithm  
1201 Suite assertion on line (P018). Lines (M009) – (M011) hold a KeyIdentifier of the protection token used to  
1202 encrypt the symmetric key as required by the SymmetricBinding assertion. Because the ProtectionToken  
1203 assertion disallowed the token from being inserted into the message and instead required a thumbprint  
1204 reference, a thumbprint reference is included to identify the token.

1205 Line (M016) is an encryption data reference that references the encrypted supporting UsernameToken on  
1206 lines (M022) – (M032). The encryption was required by the SignedEncryptedSupportingTokens assertion  
1207 on line (P038). Line (M023) references the Aes256 Encryption algorithm dictated by the Basic 256  
1208 Algorithm Suite assertion on line (P018). The encrypted symmetric key is used to encrypt the  
1209 UsernameToken as referenced on line (M026).

1210 Line (M017) is an encryption data reference that references the encrypted body of the message on lines  
1211 (M049) – (M059). The encryption was required by the EncryptedParts assertion of the input message  
1212 policy. The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line  
1213 (M053).

1214 Lines (M019) – (M021) contain a timestamp for the message as required by the IncludeTimestamp  
1215 assertion.

1216 Lines (M033) – (M045) contain the message signature.

1217 Line (M035) indicates the message timestamp is included in the signature as required by the  
1218 IncludeTimestamp assertion definition.

1219 Line (M036) indicates the supporting UsernameToken is included in the signature as required by the  
1220 SignedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is  
1221 included below to better illustrate the reference.

```
1222 (M062) <wsse:UsernameToken wsu:Id="usernameToken">  
1223 (M063) <wsse:Username>Chris</wsse:Username>  
1224 (M064) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-  
1225 200401-wss-username-token-profile-1.0#PasswordText ">sirhC</wsse:Password>  
1226 (M065) </wsse:UsernameToken>
```

1227 Line (M037) indicates the message body is included in the signature as required by the SignedParts  
1228 assertion of the input message policy.

1229 Line (M042) references the encrypted symmetric key for signing as dictated by the SymmetricBinding  
1230 assertion.

## 1231 2.2 X.509 Token Authentication Scenario Assertions

### 1232 2.2.1 (WSS1.0) X.509 Certificates, Sign, Encrypt

1233 This use-case corresponds to the situation where both parties have X.509v3 certificates (and public-  
1234 private key pairs). The requestor identifies itself to the service. The message exchange is integrity  
1235 protected and encrypted.

1236 This modeled by use of an asymmetric security binding assertion.

1237 The message level policies in this and subsequent sections cover a different scope of the web service  
1238 definition than the security binding level policy and so appear as separate policies and are attached at  
1239 WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a  
1240 set of coordinated policies one with endpoint subject and two with message subjects to achieve this use  
1241 case.

1242 The policies are as follows:

```
1243 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >  
1244 (P002) <wsp:ExactlyOne>  
1245 (P003) <wsp:All>  
1246 (P004) <sp:AsymmetricBinding>  
1247 (P005) <wsp:Policy>  
1248 (P006) <sp:InitiatorToken>  
1249 (P007) <wsp:Policy>  
1250 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1251 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
1252 (P009) <wsp:Policy>  
1253 (P010) <sp:WssX509V3Token10/>  
1254 (P011) </wsp:Policy>  
1255 (P012) </sp:X509Token>  
1256 (P013) </wsp:Policy>  
1257 (P014) </sp:InitiatorToken>  
1258 (P015) <sp:RecipientToken>  
1259 (P016) <wsp:Policy>  
1260 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1261 sx/ws-securitypolicy/200702/IncludeToken/Never">
```

```

1262 (P018) <wsp:Policy>
1263 (P019) <sp:WssX509V3Token10/>
1264 (P020) </wsp:Policy>
1265 (P021) </sp:X509Token>
1266 (P022) </wsp:Policy>
1267 (P023) </sp:RecipientToken>
1268 (P024) <sp:AlgorithmSuite>
1269 (P025) <wsp:Policy>
1270 (P026) <sp:Basic256/>
1271 (P027) </wsp:Policy>
1272 (P028) </sp:AlgorithmSuite>
1273 (P029) <sp:Layout>
1274 (P030) <wsp:Policy>
1275 (P031) <sp:Strict/>
1276 (P032) </wsp:Policy>
1277 (P033) </sp:Layout>
1278 (P034) <sp:IncludeTimestamp/>
1279 (P035) <sp:OnlySignEntireHeadersAndBody/>
1280 (P036) </wsp:Policy>
1281 (P037) </sp:AsymmetricBinding>
1282 (P038) <sp:Wss10>
1283 (P039) <wsp:Policy>
1284 (P040) <sp:MustSupportRefKeyIdentifier/>
1285 (P041) </wsp:Policy>
1286 (P042) </sp:Wss10>
1287 (P043) </wsp:All>
1288 (P044) </wsp:ExactlyOne>
1289 (P045) </wsp:Policy>
1290
1291 (P046) <wsp:Policy wsu:Id="WSS10Anonymous_with_Certificates_input_policy">
1292 (P047) <wsp:ExactlyOne>
1293 (P048) <wsp:All>
1294 (P049) <sp:SignedParts>
1295 (P050) <sp:Body/>
1296 (P051) </sp:SignedParts>
1297 (P052) <sp:EncryptedParts>
1298 (P053) <sp:Body/>
1299 (P054) </sp:EncryptedParts>
1300 (P055) </wsp:All>
1301 (P056) </wsp:ExactlyOne>
1302 (P057) </wsp:Policy>
1303
1304 (P058) <wsp:Policy wsu:Id="WSS10anonymous_with_certs_output_policy">
1305 (P059) <wsp:ExactlyOne>
1306 (P060) <wsp:All>
1307 (P061) <sp:SignedParts>
1308 (P062) <sp:Body/>
1309 (P063) </sp:SignedParts>
1310 (P064) <sp:EncryptedParts>
1311 (P065) <sp:Body/>
1312 (P066) </sp:EncryptedParts>
1313 (P067) </wsp:All>
1314 (P068) </wsp:ExactlyOne>
1315 (P069) </wsp:Policy>

```

1316 Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator’s token  
1317 must be used for the message signature and the recipient’s token must be used for message encryption.

1318 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)  
1319 indicate that the initiator token must be an X.509 token that must be included with all messages sent to  
1320 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the  
1321 WS-Security 1.0 X.509 Token Profile.

1322 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)  
1323 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token

1324 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to  
 1325 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the  
 1326 token in any messages where the token is used.

1327 Line (P034) requires the inclusion of a timestamp.

1328 Lines (P046) – (P057) contain a policy that is attached to the input message. Lines (P049) – (P051)  
 1329 require that the body of the input message must be signed. Lines (P052) – (P054) require the body of the  
 1330 input message must be encrypted.

1331 Lines (P058) – (P069) contain a policy that is attached to the output message. Lines (P061) – (P063)  
 1332 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of  
 1333 the output message must be encrypted.

1334 An example of an input message that conforms to the above stated policy is as follows.

```

1335 (M001) <?xml version="1.0" encoding="utf-8" ?>
1336 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1337 (M003)   <soap:Header>
1338 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
1339 (M005)       <xenc:EncryptedKey >
1340 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1341 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1342 (M008)         </xenc:EncryptionMethod>
1343 (M009)         <ds:KeyInfo>
1344 (M010)           <wsse:SecurityTokenReference >
1345 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1346 (M011) ValueType="...#X509SubjectKeyIdentifier">
1347 (M012)               MIGfMa0GCSq...
1348 (M013)             </wsse:KeyIdentifier>
1349 (M014)           </ds:KeyInfo>
1350 (M015)         <xenc:CipherData>
1351 (M016)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1352 (M017)         </xenc:CipherData>
1353 (M018)         <xenc:ReferenceList>
1354 (M019)           <xenc:DataReference URI="#encBody"/>
1355 (M020)         </xenc:ReferenceList>
1356 (M021)       </xenc:EncryptedKey>
1357 (M022)       <wsu:Timestamp wsu:Id="T0">
1358 (M023)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1359 (M024)       </wsu:Timestamp>
1360 (M025)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="...#X509v3"
1361 (M025) EncodingType="...#Base64Binary">
1362 (M026)         MIIEZzCCA9CgAwIBAgIQEmtJZc0...
1363 (M027)       </wsse:BinarySecurityToken>
1364 (M028)       <ds:Signature>
1365 (M029)         <ds:SignedInfo>...
1366 (M030)           <ds:Reference URI="#T0">...</ds:Reference>
1367 (M031)           <ds:Reference URI="#body">...</ds:Reference>
1368 (M032)         </ds:SignedInfo>
1369 (M033)         <ds:SignatureValue>HFLP...</ds:SignatureValue>
1370 (M034)       <ds:KeyInfo>
1371 (M035)         <wsse:SecurityTokenReference>
1372 (M036)           <wsse:Reference URI="#binaryToken"/>
1373 (M037)         </wsse:SecurityTokenReference>
1374 (M038)       </ds:KeyInfo>
1375 (M039)     </ds:Signature>
1376 (M040)   </wsse:Security>
1377 (M041) </soap:Header>
1378 (M042) <soap:Body wsu:Id="body">
1379 (M043)   <xenc:EncryptedData wsu:Id="encBody">
1380 (M044)     <xenc:CipherData>
1381 (M045)       <xenc:CipherValue>...</xenc:CipherValue>
1382 (M046)     </xenc:CipherData>
1383 (M047)   </xenc:EncryptedData>
1384 (M048) </soap:Body>

```

1385 (M049) </soap:Envelope>

1386 Line (M019) is an encryption data reference that references the encrypted body of the message on lines  
 1387 (M043) – (M047). The encryption was required by the EncryptedParts assertion of the input message  
 1388 policy. Lines (M011) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as  
 1389 required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the  
 1390 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included  
 1391 token.

1392 Lines (M022) – (M024) contain a timestamp for the message as required by the IncludeTimestamp  
 1393 assertion.

1394 Lines (M025) – (M027) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as  
 1395 required by the InitiatorToken assertion.

1396 Lines (M028) – (M039) contain the message signature.

1397 Line (M030) indicates the message timestamp is included in the signature as required by the  
 1398 IncludeTimestamp assertion definition.

1399 Line (M031) indicates the message body is included in the signature as required by the SignedParts  
 1400 assertion of the input message policy.

1401 Note that the initiator’s BinarySecurityToken is not included in the message signature as it was not  
 1402 required by policy.

1403 Lines (M035) – (M037) references the initiator’s BinarySecurityToken included in the message for  
 1404 identifying the key used for signing as dictated by the AsymmetricBinding assertion.

## 1405 2.2.2 (WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt

1406 This scenario is based on WSS Interop, Scenario 3, [Web Services Security: Interop 1](#), Draft 06, Editor,  
 1407 Hal Lockhart, BEA Systems

1408 This use case corresponds to the situation where both parties have X.509v3 certificates (and public-  
 1409 private key pairs). The requestor wishes to identify itself to the service using its X.509 credential (strong  
 1410 authentication). The message exchange needs to be integrity protected and encrypted as well. The  
 1411 difference from previous use case is that the X509 token inserted by the client is included in the message  
 1412 signature (see <ProtectTokens />).

1413 The policy is as follows:

```

1414 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1415 (P002)   <wsp:ExactlyOne>
1416 (P003)     <wsp:All>
1417 (P004)       <sp:AsymmetricBinding>
1418 (P005)         <wsp:Policy>
1419 (P006)           <sp:InitiatorToken>
1420 (P007)             <wsp:Policy>
1421 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1422 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1423 (P009)                 <wsp:Policy>
1424 (P010)                   <sp:WssX509V3Token10/>
1425 (P011)                     </wsp:Policy>
1426 (P012)                   </sp:X509Token>
1427 (P013)                   </wsp:Policy>
1428 (P014)                 </sp:InitiatorToken>
1429 (P015)               <sp:RecipientToken>
1430 (P016)                 <wsp:Policy>
1431 (P017)                   <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1432 sx/ws-securitypolicy/200702/IncludeToken/Never">
1433 (P018)                     <wsp:Policy>
1434 (P019)                       <sp:WssX509V3Token10/>
1435 (P020)                         </wsp:Policy>
1436 (P021)                       </sp:X509Token>
1437 (P022)                     </wsp:Policy>
1438 (P023)                   </sp:RecipientToken>

```

```

1439 (P024) <sp:AlgorithmSuite>
1440 (P025) <wsp:Policy>
1441 (P026) <sp:Basic256/>
1442 (P027) </wsp:Policy>
1443 (P028) </sp:AlgorithmSuite>
1444 (P029) <sp:Layout>
1445 (P030) <wsp:Policy>
1446 (P031) <sp:Strict/>
1447 (P032) </wsp:Policy>
1448 (P033) </sp:Layout>
1449 (P034) <sp:IncludeTimestamp/>
1450 (P035) <sp:ProtectTokens />
1451 (P036) <sp:OnlySignEntireHeadersAndBody/>
1452 (P037) </wsp:Policy>
1453 (P038) </sp:AsymmetricBinding>
1454 (P039) <sp:Wss10>
1455 (P040) <wsp:Policy>
1456 (P041) <sp:MustSupportRefKeyIdentifier/>
1457 (P042) </wsp:Policy>
1458 (P043) </sp:Wss10>
1459 (P044) </wsp:All>
1460 (P045) </wsp:ExactlyOne>
1461 (P046) </wsp:Policy>
1462
1463 (P047) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1464 (P048) <wsp:ExactlyOne>
1465 (P049) <wsp:All>
1466 (P050) <sp:SignedParts>
1467 (P051) <sp:Body/>
1468 (P052) </sp:SignedParts>
1469 (P053) <sp:EncryptedParts>
1470 (P054) <sp:Body/>
1471 (P055) </sp:EncryptedParts>
1472 (P056) </wsp:All>
1473 (P057) </wsp:ExactlyOne>
1474 (P058) </wsp:Policy>
1475
1476 (P059) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1477 (P060) <wsp:ExactlyOne>
1478 (P061) <wsp:All>
1479 (P062) <sp:SignedParts>
1480 (P063) <sp:Body/>
1481 (P064) </sp:SignedParts>
1482 (P065) <sp:EncryptedParts>
1483 (P066) <sp:Body/>
1484 (P067) </sp:EncryptedParts>
1485 (P068) </wsp:All>
1486 (P069) </wsp:ExactlyOne>
1487 (P070) </wsp:Policy>

```

1488 Lines (P004) – (P038) contain the AsymmetricBinding assertion which indicates that the initiator's token  
1489 must be used for the message signature and the recipient's token must be used for message encryption.

1490 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)  
1491 indicate that the initiator token must be an X.509 token that must be included with all messages sent to  
1492 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the  
1493 WS-Security 1.0 X.509 Token Profile.

1494 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)  
1495 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token  
1496 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to  
1497 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the  
1498 token in any messages where the token is used.

1499 Line (P034) requires the inclusion of a timestamp.

1500 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the  
 1501 token used to generate that signature.

1502 Lines (P047) – (P058) contain a policy that is attached to the input message. Lines (P050) – (P052)  
 1503 require that the body of the input message must be signed. Lines (P053) – (P055) require the body of the  
 1504 input message must be encrypted.

1505 Lines (P059) – (P070) contain a policy that is attached to the output message. Lines (P062) – (P064)  
 1506 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of  
 1507 the output message must be encrypted.

1508 An example of an input message that conforms to the above stated policy is as follows.

```

1509 (M001) <?xml version="1.0" encoding="utf-8" ?>
1510 (M002) <soapenv:Envelope xmlns:soapenv="..."
1511 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="..." xmlns:xenc="..." ...>
1512 (M003)   <soapenv:Header>
1513 (M004)     <wsse:Security xmlns:wsse="..." xmlns:ds="..."
1514 soapenv:mustUnderstand="1">
1515 (M005)       <xenc:EncryptedKey >
1516 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1517 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1518 (M008)         </xenc:EncryptionMethod>
1519 (M009)         <ds:KeyInfo>
1520 (M010)           <wsse:SecurityTokenReference >
1521 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1522 ValueTypes="...#X509SubjectKeyIdentifier">CuJd...=</wsse:KeyIdentifier>
1523 (M012)           </wsse:SecurityTokenReference>
1524 (M013)         </ds:KeyInfo>
1525 (M014)         <xenc:CipherData>
1526 (M015)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1527 (M016)         </xenc:CipherData>
1528 (M017)         <xenc:ReferenceList>
1529 (M018)           <xenc:DataReference URI="#encBody"/>
1530 (M019)         </xenc:ReferenceList>
1531 (M020)       </xenc:EncryptedKey>
1532 (M021)       <wsu:Timestamp wsu:Id="Timestamp" >
1533 (M022)         <wsu:Created>2007-03-26T16:53:39Z</wsu:Created>
1534 (M023)       </wsu:Timestamp>
1535 (M024)       <wsse:BinarySecurityToken wsu:Id="bst" ValueTypes="...#X509v3"
1536 EncodingType="...#Base64Binary">MIID...=</wsse:BinarySecurityToken>
1537 (M025)       <ds:Signature>
1538 (M026)         <ds:SignedInfo>
1539 (M027)           <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#" />
1540 (M028)           <ds:SignatureMethod Algorithm="...#rsa-sha1"/>
1541 (M029)           <ds:Reference URI="#Timestamp">
1542 (M030)             <ds:Transforms>... </ds:Transforms>
1543 (M031)             <ds:DigestMethod Algorithm="...#sha1"/>
1544 (M032)             <ds:DigestValue>+g0I...=</ds:DigestValue>
1545 (M033)           </ds:Reference>
1546 (M034)           <ds:Reference URI="#Body">...</ds:Reference>
1547 (M035)           <ds:Reference URI="#bst">...</ds:Reference>
1548 (M036)         </ds:SignedInfo>
1549 (M037)         <ds:SignatureValue>RRT...=</ds:SignatureValue>
1550 (M038)       </ds:Signature>
1551 (M039)       <wsse:SecurityTokenReference >
1552 (M040)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1553 ValueTypes="...#X509SubjectKeyIdentifier">Xeg5...=</wsse:KeyIdentifier>
1554 (M041)       </wsse:SecurityTokenReference>
1555 (M042)     </ds:KeyInfo>
1556 (M043)   </ds:Signature>
1557 (M044) </wsse:Security>
1558 (M045) </soapenv:Header>
1559 (M046) <soapenv:Body wsu:Id="Body" >
1560 (M047)   <xenc:EncryptedData Id="encBody" Type="...#Content"
1561 (M048)     MimeTypes="text/xml" Encoding="UTF-8" >

```

```

1562 (M049) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1563 (M050) <xenc:CipherData>
1564 (M051) <xenc:CipherValue>W84fn...1</xenc:CipherValue>
1565 (M052) </xenc:CipherData>
1566 (M053) </xenc:EncryptedData>
1567 (M054) </soapenv:Body>
1568 (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.

### 1591 **2.2.2.1 (WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption**

1592 This scenario is based on WSS Interop, Scenario 4, [Web Services Security: Interop 2](#).

1593 A common variation on the previous example is where X.509 Certificates are still used for authentication,  
 1594 but a mutually agreed upon symmetric key is used for encryption.

1595 In this use case the policy is the same except that the InitiatorToken and RecipientToken are now each  
 1596 just SignatureTokens.

1597 A second variation in this use case is that the mutually agreed upon symmetric encryption key is  
 1598 characterized as an “IssuedToken” with a mutually agreed upon URI used to identify the out of band (not  
 1599 included in the message) token, which is included in a SupportingTokens element.

1600 The policy is as follows:

```

1601 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1602 (P002) <wsp:ExactlyOne>
1603 (P003) <wsp:All>
1604 (P004) <sp:AsymmetricBinding>
1605 (P005) <wsp:Policy>
1606 (P006) <sp:InitiatorSignatureToken>
1607 (P007) <wsp:Policy>
1608 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1609 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1610 (P009) <wsp:Policy>
1611 (P010) <sp:WssX509V3Token10/>
1612 (P011) </wsp:Policy>
1613 (P012) </sp:X509Token>
1614 (P013) </wsp:Policy>

```

```

1615 (P014) </sp:InitiatorSignatureToken>
1616 (P015) <sp:RecipientSignatureToken>
1617 (P016) <wsp:Policy>
1618 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1619 sx/ws-securitypolicy/200702/IncludeToken/Never">
1620 (P018) <wsp:Policy>
1621 (P019) <sp:WssX509V3Token10/>
1622 (P020) </wsp:Policy>
1623 (P021) </sp:X509Token>
1624 (P022) </wsp:Policy>
1625 (P023) </sp:RecipientSignatureToken>
1626 (P024) <sp:AlgorithmSuite>
1627 (P025) <wsp:Policy>
1628 (P026) <sp:Basic256/>
1629 (P027) </wsp:Policy>
1630 (P028) </sp:AlgorithmSuite>
1631 (P029) <sp:Layout>
1632 (P030) <wsp:Policy>
1633 (P031) <sp:Strict/>
1634 (P032) </wsp:Policy>
1635 (P033) </sp:Layout>
1636 (P034) <sp:IncludeTimestamp/>
1637 (P035) <sp:ProtectTokens />
1638 (P036) <sp:OnlySignEntireHeadersAndBody/>
1639 (P037) </wsp:Policy>
1640 (P038) </sp:AsymmetricBinding>
1641 (P039) <sp:SupportingTokens>
1642 (P040) <wsp:Policy>
1643 (P041) <sp:IssuedToken>
1644 (P042) <sp:Issuer>SomeMutuallyAgreedURI</sp:Issuer>
1645 (P043) <sp:RequireExternalReference/>
1646 (P044) </sp:IssuedToken>
1647 (P045) <sp:EncryptedParts>
1648 (P046) <sp:Body/>
1649 (P047) </sp:EncryptedParts>
1650 (P048) </wsp:Policy>
1651 (P049) </sp:SupportingTokens>
1652 (P050) <sp:Wss10>
1653 (P051) <wsp:Policy>
1654 (P052) <sp:MustSupportRefKeyIdentifier/>
1655 (P053) </wsp:Policy>
1656 (P054) </sp:Wss10>
1657 (P055) </wsp:All>
1658 (P056) </wsp:ExactlyOne>
1659 (P057) </wsp:Policy>
1660
1661 (P058) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1662 (P059) <wsp:ExactlyOne>
1663 (P060) <wsp:All>
1664 (P061) <sp:SignedParts>
1665 (P062) <sp:Body/>
1666 (P063) </sp:SignedParts>
1667 (P064) <sp:EncryptedParts>
1668 (P065) <sp:Body/>
1669 (P066) </sp:EncryptedParts>
1670 (P067) </wsp:All>
1671 (P068) </wsp:ExactlyOne>
1672 (P069) </wsp:Policy>
1673
1674 (P070) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1675 (P071) <wsp:ExactlyOne>
1676 (P072) <wsp:All>
1677 (P073) <sp:SignedParts>
1678 (P074) <sp:Body/>

```

```

1679 (P075)      </sp:SignedParts>
1680 (P076)      <sp:EncryptedParts>
1681 (P077)      <sp:Body/>
1682 (P078)      </sp:EncryptedParts>
1683 (P079)      </wsp:All>
1684 (P080)      </wsp:ExactlyOne>
1685 (P081) </wsp:Policy>

```

1686 Lines (P004) – (P038) contain the AsymmetricBindingAssertion which indicates that the initiator’s token  
1687 must be used for the message signature and the recipient’s token must be used for message encryption.

1688 Lines (P006) – (P014) contain the InitiatorSignatureToken assertion. Within that assertion lines (P008) –  
1689 (P012) indicate that the initiator token must be an X.509 token that must be included with all messages  
1690 sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as  
1691 described in the WS-Security 1.0 X.509 Token Profile. By specifying that this is an  
1692 InitiatorSignatureToken, it will only be used to sign the message and not used for encryption.

1693 Lines (P015) – (P023) contain the RecipientSignatureToken assertion. Within that assertion lines (P017)  
1694 – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0  
1695 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead,  
1696 according to the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to  
1697 identify the token in any messages where the token is used. By specifying that this is an  
1698 RecipientSignatureToken, it will only be used to sign the response and not used for encryption.

1699 Line (P034) requires the inclusion of a timestamp.

1700 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the  
1701 token or token reference associated with the generation of that signature.

1702 Lines (P039) – (P049) contain a SupportingTokens assertion that contains an IssuedToken (P041) –  
1703 (P044), which contains an Issuer element (P042) that identifies an explicit URI  
1704 (“SomeMutuallyAgreedURI”) that must be used to indicate what token will be used. (Since the content of  
1705 the IssuedToken element is specific to the Issuer, any mechanism can be used to identify the key, and in  
1706 this case the simplest method of identifying the issuer with the explicit key has been chosen only to  
1707 illustrate one possible method, but not to recommend as opposed to any other method.) Line (P043),  
1708 RequireExternalReference indicates that the IssuedToken requires a token that is referenced external to  
1709 the message. Lines (P045) – (P047) indicate that the token is to be used for encrypting parts of the  
1710 message, explicitly, line (P046), the Body of the message.

1711 Lines (P058) – (P069) contain a policy that is attached to the input message. Lines (P061) – (P063)  
1712 require that the body of the input message must be signed. Lines (P064) – (P066) require the body of the  
1713 input message must be encrypted.

1714 Lines (P070) – (P081) contain a policy that is attached to the output message. Lines (P073) – (P075)  
1715 require that the body of the output message must be signed. Lines (P076) – (P078) require the body of  
1716 the output message must be encrypted.

1717

1718 The following example message is derived from the WSS Interop2 document.

1719

```

1720 (M001)      <?xml version="1.0" encoding="utf-8" ?>
1721 (M002)      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1722 (M003)          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1723 (M004)          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1724 (M005)      <soap:Header>
1725 (M006)          <wsse:Security soap:mustUnderstand="1"
1726 (M007)              xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1727 (M008)              <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1728 (M009)                  <xenc:DataReference URI="#enc" />
1729 (M010)              </xenc:ReferenceList>
1730 (M011)          <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1731 (M012)              wsu:Id="timestamp">
1732 (M013)              <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1733 (M014)          </wsu:Timestamp>
1734 (M015)          <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1735 (M016)              EncodingType="wsse:Base64Binary"

```

```

1736 (M017)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1737 (M018)      wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1738 (M019)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1739 (M020)      <SignedInfo>
1740 (M021)      <CanonicalizationMethod
1741 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1742 (M023)      <SignatureMethod
1743 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1744 (M025)      <Reference URI="#body">
1745 (M026)      <Transforms>
1746 (M027)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1747 (M028)      </Transforms>
1748 (M029)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1749 (M030)      <DigestValue>QTV...dw</DigestValue>
1750 (M031)      </Reference>
1751 (M032)      <Reference URI="#myCert">
1752 (M033)      <Transforms>
1753 (M034)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1754 (M035)      </Transforms>
1755 (M036)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1756 (M037)      <DigestValue>XYZ...ab</DigestValue>
1757 (M038)      </Reference>
1758 (M039)      <Reference URI="#timestamp">
1759 (M040)      <Transforms>
1760 (M041)      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1761 (M042)      </Transforms>
1762 (M043)      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1763 (M044)      <DigestValue>XYZ...ab</DigestValue>
1764 (M045)      </Reference>
1765 (M046)      </SignedInfo>
1766 (M047)      <SignatureValue>H+x0...gUw</SignatureValue>
1767 (M048)      <KeyInfo>
1768 (M049)      <wsse:SecurityTokenReference>
1769 (M050)      <wsse:Reference URI="#myCert" />
1770 (M051)      </wsse:SecurityTokenReference>
1771 (M052)      </KeyInfo>
1772 (M053)      </Signature>
1773 (M054)      </wsse:Security>
1774 (M055)      </soap:Header>
1775 (M056)      <soap:Body wsu:Id="body"
1776 (M057)      xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1777 (M058)      <xenc:EncryptedData Id="enc"
1778 (M059)      Type="http://www.w3.org/2001/04/xmlenc#Content"
1779 (M060)      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1780 (M061)      <xenc:EncryptionMethod
1781 (M062)      Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
1782 (M063)      <xenc:KeyInfo>
1783 (M064)      <xenc:KeyName>SomeMutuallyAgreedURI</xenc:KeyName>
1784 (M065)      </xenc:KeyInfo>
1785 (M066)      <xenc:CipherData>
1786 (M067)      <xenc:CipherValue>AYb...Y8</xenc:CipherValue>
1787 (M068)      </xenc:CipherData>
1788 (M069)      </xenc:EncryptedData>
1789 (M070)      </soap:Body>
1790 (M071)      </soap:Envelope>

```

1791 In the above example, the main point of attention is line (M064), where the KeyName of the  
1792 EncryptedData element is "SomeMutuallyAgreedURI". As indicated above the specific techniques used to  
1793 identify and apply this token are totally within agreed upon methods define by the mutual parties.

1794 Lines (M005) – (M055) contain the SOAP Header element, which contains the WS-Security header  
1795 (M006) – (M054).

1796 The ReferenceList (M008) – (M010) contains an internal reference (M009), "enc", identifying the  
1797 EncryptedData element Id, line (M058). This EncryptedData (M058) – (M069) inside the SOAP Body,  
1798 (M056) – (M070), was required to be encrypted by the policy (P058) – (P069) as described above.

1799 The Timestamp (M011) – (M014) is required by the policy line (P034).

1800 The BinarySecurityToken (M015) – (M018) contains the actual certificate used to sign the request as  
1801 required by the policy, (P008), IncludeToken/AlwaysToRecipient.

1802 The Signature (M019) – (M053) contains a SignedInfo (M020) – (M046) that identifies the “#body” (M025)  
1803 the “#myCert” (M032), and the “#timestamp” (M039) as the elements covered by the signature. The  
1804 Reference (M032) – (M038) with URI=“#myCert” that covers the BinarySecurityToken (M015) – (M018)  
1805 was required by ProtectTokens in the policy (P035).

1806 The KeyInfo (M048) – (M052) uses a SecurityTokenReference to point to the “myCert”  
1807 BinarySecurityToken.

1808 The EncryptedData (M058) – (M069) was described above, where it was pointed out that line (M064)  
1809 contains the external reference (“SomeMutuallyAgreedURI”) to the mutually shared symmetric key used  
1810 for encryption.

### 1811 2.2.3 (WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt

1812 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”  
1813 [WS-SECURE-INTEROP] (see also sec 2.1.4)

1814 In this use case the Request is signed using DerivedKeyToken1(K), then encrypted using a  
1815 DerivedKeyToken2(K) where K is ephemeral key protected for the server's certificate. Response is signed  
1816 using DKT3(K), (if needed) encrypted using DKT4(K). The requestor does not wish to identify himself; the  
1817 message exchange is protected using derived symmetric keys. As a simpler, but less secure, alternative,  
1818 ephemeral key K (instead of derived keys) could be used for message protection by simply omitting the  
1819 sp:RequireDerivedKeys assertion.

1820 The policy is as follows:

```
1821 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">  
1822 (P002)   <wsp:ExactlyOne>  
1823 (P003)     <wsp:All>  
1824 (P004)       <sp:SymmetricBinding>  
1825 (P005)         <wsp:Policy>  
1826 (P006)           <sp:ProtectionToken>  
1827 (P007)             <wsp:Policy>  
1828 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1829 sx/ws-securitypolicy/200702/IncludeToken/Never">  
1830 (P009)                 <wsp:Policy>  
1831 (P010)                   <sp:RequireDerivedKeys/>  
1832 (P011)                   <sp:RequireThumbprintReference/>  
1833 (P012)                   <sp:WssX509V3Token11/>  
1834 (P013)                 </wsp:Policy>  
1835 (P014)               </sp:X509Token>  
1836 (P015)             </wsp:Policy>  
1837 (P016)           </sp:ProtectionToken>  
1838 (P017)         <sp:AlgorithmSuite>  
1839 (P018)           <wsp:Policy>  
1840 (P019)             <sp:Basic256/>  
1841 (P020)           </wsp:Policy>  
1842 (P021)         </sp:AlgorithmSuite>  
1843 (P022)       <sp:Layout>  
1844 (P023)         <wsp:Policy>  
1845 (P024)           <sp:Strict/>  
1846 (P025)         </wsp:Policy>  
1847 (P026)       </sp:Layout>  
1848 (P027)     <sp:IncludeTimestamp/>  
1849 (P028)     <sp:OnlySignEntireHeadersAndBody/>  
1850 (P029)   </wsp:Policy>  
1851 (P030) </sp:SymmetricBinding>  
1852 (P031) <sp:Wss11>  
1853 (P032)   <wsp:Policy>  
1854 (P033)     <sp:MustSupportRefKeyIdentifier/>  
1855 (P034)     <sp:MustSupportRefIssuerSerial/>  
1856 (P035)     <sp:MustSupportRefThumbprint/>  
1857 (P036)     <sp:MustSupportRefEncryptedKey/>  
1858 (P037)     <sp:RequireSignatureConfirmation/>  
1859 (P038)   </wsp:Policy>
```

```

1860 (P039)      </sp:Wss11>
1861 (P040)      </wsp:All>
1862 (P041)      </wsp:ExactlyOne>
1863 (P042) </wsp:Policy>
1864
1865 (P043) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_input_policy">
1866 (P044)   <wsp:ExactlyOne>
1867 (P045)   <wsp:All>
1868 (P046)   <sp:SignedParts>
1869 (P047)   <sp:Body/>
1870 (P048)   </sp:SignedParts>
1871 (P049)   <sp:EncryptedParts>
1872 (P050)   <sp:Body/>
1873 (P051)   </sp:EncryptedParts>
1874 (P052)   </wsp:All>
1875 (P053)   </wsp:ExactlyOne>
1876 (P054) </wsp:Policy>
1877
1878 (P055) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_output_policy">
1879 (P056)   <wsp:ExactlyOne>
1880 (P057)   <wsp:All>
1881 (P058)   <sp:SignedParts>
1882 (P059)   <sp:Body/>
1883 (P060)   </sp:SignedParts>
1884 (P061)   <sp:EncryptedParts>
1885 (P062)   <sp:Body/>
1886 (P063)   </sp:EncryptedParts>
1887 (P064)   </wsp:All>
1888 (P065)   </wsp:ExactlyOne>
1889 (P066) </wsp:Policy>

```

1890 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token  
1891 must be used for both message signature and message encryption.

1892 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)  
1893 indicate that the ProtectionToken must be an X.509 token that must not be included with any message  
1894 sent between the Initiator and Recipient.

1895 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an  
1896 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the  
1897 MustSupportRefThumbprint assertion on line (P035) and RequireThumbprintReference on line (P011), a  
1898 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token  
1899 is used.

1900 Line (P027) requires the inclusion of a timestamp.

1901 Lines (P031) – (P039) contain some WS-Security 1.1 related interoperability requirements, specifically  
1902 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires  
1903 signature confirmation on the response.

1904 Lines (P043) – (P054) contain a policy that is attached to the input message. Lines (P045) – (P048)  
1905 require that the body of the input message must be signed. Lines (P049) – (P051) require the body of the  
1906 input message must be encrypted.

1907 Lines (P055) – (P066) contain a policy that is attached to the output message. Lines (P057) – (P060)  
1908 require that the body of the output message must be signed. Lines (P061) – (P063) require the body of  
1909 the output message must be encrypted.

1910 An example of an input message that conforms to the above stated policy is as follows.

1911 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,  
1912 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the  
1913 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-  
1914 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```

1915 (M001) <?xml version="1.0" encoding="utf-8" ?>
1916 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">

```

```

1917 (M003) <env:Header>
1918 (M004) <wsse:Security xmlns:wsse="..." xmlns:wssell="..." xmlns:wsc="..."
1919 env:mustUnderstand="1">
1920 (M005) <xenc:EncryptedKey Id="encKey" >
1921 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1922 (M007) <ds:DigestMethod Algorithm...#sha1" />
1923 (M008) </xenc:EncryptionMethod>
1924 (M009) <ds:KeyInfo >
1925 (M010) <wsse:SecurityTokenReference >
1926 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1927 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
1928 (M012) </wsse:SecurityTokenReference>
1929 (M013) </ds:KeyInfo>
1930 (M014) <xenc:CipherData>
1931 (M015) <xenc:CipherValue>TE...=</xenc:CipherValue>
1932 (M016) </xenc:CipherData>
1933 </xenc:EncryptedKey>
1934 (M018) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey1">
1935 (M019) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1936 (M020) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1937 (M021) </wsse:SecurityTokenReference>
1938 (M022) <wsc:Generation>0</wsc:Generation>
1939 (M023) <wsc:Length>32</wsc:Length>
1940 (M024) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1941 (M025) <wsc:Nonce>39...=</wsc:Nonce>
1942 </wsc:DerivedKeyToken>
1943 (M027) <xenc:ReferenceList>
1944 (M028) <xenc:DataReference URI="#encBody"/>
1945 (M029) </xenc:ReferenceList>
1946 (M030) <wsc:DerivedKeyToken Algorithm=".../p_shal" wsu:Id="DKey2">
1947 (M031) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1948 (M032) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1949 (M033) </wsse:SecurityTokenReference>
1950 (M034) <wsc:Generation>0</wsc:Generation>
1951 (M035) <wsc:Length>32</wsc:Length>
1952 (M036) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1953 (M037) <wsc:Nonce>...=</wsc:Nonce>
1954 </wsc:DerivedKeyToken>
1955 (M039) <wsu:Timestamp wsu:Id="Timestamp" >
1956 (M040) <wsu:Created>2007-03-26T23:43:13Z</wsu:Created>
1957 </wsu:Timestamp>
1958 (M042) <ds:Signature >
1959 (M043) <ds:SignedInfo>
1960 (M044) ...
1961 (M045) <ds:Reference URI="#Timestamp">...</ds:Reference>
1962 (M046) <ds:Reference URI="#Body">...</ds:Reference>
1963 </ds:SignedInfo>
1964 (M048) <ds:SignatureValue>Yu...=</ds:SignatureValue>
1965 (M049) <ds:KeyInfo>
1966 (M050) <wsse:SecurityTokenReference >
1967 (M051) <wsse:Reference URI="#DKey2" ValueType=".../dk"/>
1968 (M052) </wsse:SecurityTokenReference>
1969 (M053) </ds:KeyInfo>
1970 (M054) </ds:Signature>
1971 (M055) </wsse:Security>
1972 (M056) </env:Header>
1973 (M057) <env:Body wsu:Id="Body" >
1974 (M058) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
1975 Encoding="UTF-8" >
1976 (M059) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1977 (M060) <ds:KeyInfo >
1978 (M061) <wsse:SecurityTokenReference >
1979 (M062) <wsse:Reference URI="#DKey1" ValueType=".../dk"/>
1980 (M063) </wsse:SecurityTokenReference>
1981 (M064) </ds:KeyInfo>
1982 (M065) <xenc:CipherData>
1983 (M066) <xenc:CipherValue>p53...f</xenc:CipherValue>
1984 (M067) </xenc:CipherData>
1985 (M068) </xenc:EncryptedData>
1986 (M069) </env:Body>

```

1987 (M070) </env:Envelope>

1988

1989 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1  
 1990 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.  
 1991 Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is  
 1992 used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value  
 1993 type of #ThumbprintSHA1 for Thumbprint Reference, it is required by the policy line (P011) of the  
 1994 Thumbprint Reference.

1995 Lines (M018) – (M022) is the first wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines  
 1996 (M005) – (M017), which is referenced using the WSS1.1 #EncryptedKey SecurityTokenReference  
 1997 mechanism contained in lines (M019) – (M021), and used for body encryption and it is referenced on line  
 1998 (M062).

1999 Lines (M027) – (M029) is an encryption data reference that references the encrypted body of the  
 2000 message on lines (M058) – (M068). The encryption was required by the EncryptedParts assertion of the  
 2001 input message policy.

2002 Lines (M030) – (M038) is the second wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines  
 2003 (M005) – (M017) and used for signature referenced on line (M051).

2004 Lines (M039) – (M041) contain a timestamp for the message as required by the IncludeTimestamp  
 2005 assertion.

2006 Lines (M042) – (M054) contain the message signature.

2007 Line (M045) indicates the message timestamp is included in the signature as required by the  
 2008 IncludeTimestamp assertion definition.

2009 Line (M046) indicates the message body is included in the signature as required by the SignedParts  
 2010 assertion of the input message policy.

## 2011 2.2.4 (WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt

2012 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”  
 2013 [[WS-SECURE-INTEROP](#)] (see also sec 2.1.4)

2014 Client and server X509 certificates are used for client and server authorization respectively. Request is  
 2015 signed using K, then encrypted using K, K is ephemeral key protected for server’s certificate. Signature  
 2016 corresponding to K is signed using client certificate. Response is signed using K, encrypted using K,  
 2017 encrypted key K is not included in response. Alternatively, derived keys can be used for each of the  
 2018 encryption and signature operations by simply adding an sp:RequireDerivedKeys assertion.

2019 The policy is as follows:

```

2020
2021 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">
2022 (P002)   <wsp:ExactlyOne>
2023 (P003)     <wsp:All>
2024 (P004)       <sp:SymmetricBinding>
2025 (P005)         <wsp:Policy>
2026 (P006)           <sp:ProtectionToken>
2027 (P007)             <wsp:Policy>
2028 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2029 sx/ws-securitypolicy/200702/IncludeToken/Never">
2030 (P009)                 <wsp:Policy>
2031 (P010)                   <sp:RequireDerivedKeys/>
2032 (P011)                   <sp:RequireThumbprintReference/>
2033 (P012)                   <sp:WssX509V3Token11/>
2034 (P013)                 </wsp:Policy>
2035 (P014)               </sp:X509Token>
2036 (P015)             </wsp:Policy>
2037 (P016)           </sp:ProtectionToken>
2038 (P017)         <sp:AlgorithmSuite>
2039 (P018)       </wsp:Policy>

```

```

2040 (P019)         <sp:Basic256/>
2041 (P020)         </wsp:Policy>
2042 (P021)         </sp:AlgorithmSuite>
2043 (P022)         <sp:Layout>
2044 (P023)         <wsp:Policy>
2045 (P024)         <sp:Strict/>
2046 (P025)         </wsp:Policy>
2047 (P026)         </sp:Layout>
2048 (P027)         <sp:IncludeTimestamp/>
2049 (P028)         <sp:OnlySignEntireHeadersAndBody/>
2050 (P029)         </wsp:Policy>
2051 (P030)         </sp:SymmetricBinding>
2052 (P031)         <sp:EndorsingSupportingTokens>
2053 (P032)         <wsp:Policy>
2054 (P033)         <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2055 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2056 (P034)         <wsp:Policy>
2057 (P035)         <sp:RequireThumbprintReference/>
2058 (P036)         <sp:WssX509V3Token11/>
2059 (P037)         </wsp:Policy>
2060 (P038)         </sp:X509Token>
2061 (P039)         </wsp:Policy>
2062 (P040)         </sp:EndorsingSupportingTokens>
2063 (P041)         <sp:Wss11>
2064 (P042)         <wsp:Policy>
2065 (P043)         <sp:MustSupportRefKeyIdentifier/>
2066 (P044)         <sp:MustSupportRefIssuerSerial/>
2067 (P045)         <sp:MustSupportRefThumbprint/>
2068 (P046)         <sp:MustSupportRefEncryptedKey/>
2069 (P047)         <sp:RequireSignatureConfirmation/>
2070 (P048)         </wsp:Policy>
2071 (P049)         </sp:Wss11>
2072 (P050)         </wsp:All>
2073 (P051)         </wsp:ExactlyOne>
2074 (P052) </wsp:Policy>
2075
2076 (P053) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_input_policy">
2077 (P054)   <wsp:ExactlyOne>
2078 (P055)   <wsp:All>
2079 (P056)     <sp:SignedParts>
2080 (P057)       <sp:Body/>
2081 (P058)     </sp:SignedParts>
2082 (P059)     <sp:EncryptedParts>
2083 (P060)       <sp:Body/>
2084 (P061)     </sp:EncryptedParts>
2085 (P062)   </wsp:All>
2086 (P063) </wsp:ExactlyOne>
2087 (P064) </wsp:Policy>
2088
2089 (P065) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_output_policy">
2090 (P066)   <wsp:ExactlyOne>
2091 (P067)   <wsp:All>
2092 (P068)     <sp:SignedParts>
2093 (P069)       <sp:Body/>
2094 (P070)     </sp:SignedParts>
2095 (P071)     <sp:EncryptedParts>
2096 (P072)       <sp:Body/>
2097 (P073)     </sp:EncryptedParts>
2098 (P074)   </wsp:All>
2099 (P075) </wsp:ExactlyOne>
2100 (P076) </wsp:Policy>

```

2101 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token  
2102 must be used for both message signature and message encryption.

2103 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)  
2104 indicate that the initiator token must be an X.509 token that must not be included with all messages sent  
2105 between the recipient and Recipient.

2106 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an  
2107 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the  
2108 MustSupportRefThumbprint assertion on line (P043) and RequireThumbprintReference on line (P011), a  
2109 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token  
2110 is used.

2111 Line (P027) requires the inclusion of a timestamp.

2112 Lines (P031) – (P040) contain the EndorsingSupportingTokens assertion which indicates that the message  
2113 signature should be endorsed by client's X509 certificate on line (P033) – (P038). Line (P033)  
2114 IncludeToken=".../AlwaysToRecipient" indicates the endorsing is only required when the message is sent to recipient.  
2115 Line (P036) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security  
2116 1.1 X.509 Token Profile. and RequireThumbprintReference on line (P035), a Thumbprint Reference of  
2117 KeyIdentifier must be used to identify the token in any messages where the token is used.

2118 Lines (P041) – (P049) contain some WS-Security 1.1 related interoperability requirements, specifically  
2119 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires  
2120 signature confirmation on the response.

2121 Lines (P053) – (P064) contain a policy that is attached to the input message. Lines (P055) – (P058)  
2122 require that the body of the input message must be signed. Lines (P059) – (P061) require the body of the  
2123 input message must be encrypted.

2124 Lines (P065) – (P076) contain a policy that is attached to the output message. Lines (P067) – (P070)  
2125 require that the body of the output message must be signed. Lines (P071) – (P073) require the body of  
2126 the output message must be encrypted.

2127 An example of an input message that conforms to the above stated policy is as follows.

2128 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,  
2129 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the  
2130 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-  
2131 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
2132 (M001) <?xml version="1.0" encoding="utf-8" ?>
2133 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
2134 (M003) <env:Header>
2135 (M004) <wsse:Security xmlns:wsse="..." xmlns:wssell="..." env:mustUnderstand="1">
2136 (M005) <xenc:EncryptedKey Id="encKey" >
2137 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
2138 (M007) <ds:DigestMethod Algorithm="...#sha1" />
2139 (M008) </xenc:EncryptionMethod>
2140 (M009) <ds:KeyInfo >
2141 (M010) <wsse:SecurityTokenReference >
2142 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2143 (M012) Value="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
2144 (M013) </wsse:SecurityTokenReference>
2145 (M014) </ds:KeyInfo>
2146 (M015) <xenc:CipherData>
2147 (M016) <xenc:CipherValue>eI...=</xenc:CipherValue>
2148 (M017) </xenc:CipherData>
2149 (M018) </xenc:EncryptedKey>
2150 (M019) <wsse:BinarySecurityToken ValueType="...#X509v3"
2151 EncodingType="...#Base64Binary">MI...=</wsse:BinarySecurityToken>
2152 (M020) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
2153 wsu:Id="derivedKeyToken1">
2154 (M021) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey" >
2155 (M022) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
2156 (M023) </wsse:SecurityTokenReference>
2157 (M024) <wsc:Generation>0</wsc:Generation>
2158 (M025) <wsc:Length>32</wsc:Length>
2159 (M026) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2160 (M027) <wsc:Nonce>+R...=</wsc:Nonce>
2161 (M028) </wsc:DerivedKeyToken>
```

```

2162 (M029) <wsu:Timestamp wsu:Id="Timestamp" >
2163 (M030) <wsu:Created>2007-03-31T04:27:21Z</wsu:Created>
2164 (M031) </wsu:Timestamp>
2165 (M032) <n1:ReferenceList xmlns:n1=".../xmlenc#">
2166 (M033) <n1:DataReference URI="#encBody"/>
2167 (M034) </n1:ReferenceList>
2168 (M035) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_shal"
2169 wsu:Id="derivedKeyToken2">
2170 (M036) <wsse:SecurityTokenReference wsse1:TokenType="...#EncryptedKey" >
2171 (M037) <wsse:Reference ValueType="...EncryptedKey" URI="#encKey"/>
2172 (M038) </wsse:SecurityTokenReference>
2173 (M039) <wsc:Generation>0</wsc:Generation>
2174 (M040) <wsc:Length>32</wsc:Length>
2175 (M041) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2176 (M042) <wsc:Nonce>wL...=</wsc:Nonce>
2177 (M043) </wsc:DerivedKeyToken>
2178 (M044) <ds:Signature Id="messageSignature">
2179 (M045) <ds:SignedInfo>
2180 (M046) ...
2181 (M047) <ds:Reference URI="#Timestamp">
2182 (M048) ...
2183 (M049) </ds:Reference>
2184 (M050) <ds:Reference URI="#Body">
2185 (M051) ...
2186 (M052) </ds:Reference>
2187 (M053) </ds:SignedInfo>
2188 (M054) <ds:SignatureValue>abcdefg</ds:SignatureValue>
2189 (M055) <ds:KeyInfo>
2190 (M056) <wsse:SecurityTokenReference >
2191 (M057) <wsse:Reference URI="#derivedKeyToken2" ValueType=".../dk"/>
2192 (M058) </wsse:SecurityTokenReference>
2193 (M059) </ds:KeyInfo>
2194 (M060) </ds:Signature>
2195 (M061) <ds:Signature >
2196 (M062) <ds:SignedInfo>
2197 (M063) ...
2198 (M064) <ds:Reference URI="#messageSignature">
2199 (M065) ...
2200 (M066) </ds:Reference>
2201 (M067) </ds:SignedInfo>
2202 (M068) <ds:SignatureValue>hijklmnop</ds:SignatureValue>
2203 (M069) <ds:KeyInfo>
2204 (M070) <wsse:SecurityTokenReference >
2205 (M071) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2206 ValueType="...#ThumbprintSHA1">Pj...=</wsse:KeyIdentifier>
2207 (M072) </wsse:SecurityTokenReference>
2208 (M073) </ds:KeyInfo>
2209 (M074) </ds:Signature>
2210 (M075) </wsse:Security>
2211 (M076) </env:Header>
2212 (M077) <env:Body wsu:Id="Body" >
2213 (M078) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
2214 Encoding="UTF-8" >
2215 (M079) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
2216 (M080) <ds:KeyInfo >
2217 (M081) <wsse:SecurityTokenReference >
2218 (M082) <wsse:Reference URI="#derivedKeyToken1" ValueType=".../dk"/>
2219 (M083) </wsse:SecurityTokenReference>
2220 (M084) </ds:KeyInfo>
2221 (M085) <xenc:CipherData>
2222 (M086) <xenc:CipherValue>70...=</xenc:CipherValue>
2223 (M087) </xenc:CipherData>
2224 (M088) </xenc:EncryptedData>
2225 (M089) </env:Body>
2226 (M090) </env:Envelope>
2227 (M091)

```

2228  
2229 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1  
2230 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.

2231 Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token  
 2232 from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.  
 2233 In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint  
 2234 Reference, and it is required by the policy line (P011) of the Thumbprint Reference.

2235 Line (M019) is an X509 BinarySecurityToken that contains the actual X509 certificate that is used by the  
 2236 endorsing signature described below. The token is required to be present in the message based on the  
 2237 line (P033) that requires this token for the message sent to the recipient.

2238 Lines (M020) – (M027) is the first derived key token. It is derived from the EncryptedKey of lines (M005) –  
 2239 (M017) and used for body encryption referenced on line (M081).

2240 Lines (M028) – (M030) contain a Timestamp element for the message as required by the  
 2241 IncludeTimestamp assertion.

2242 Lines (M031) – (M033) contain an encryption data reference that references the encrypted body of the  
 2243 message on lines (M077) – (M087). The encryption was required by the EncryptedParts assertion of the  
 2244 input message policy.

2245 Lines (M034) – (M042) is the second derived key token. It is derived from the EncryptedKey of lines  
 2246 (M005) – (M017) and used by the signature that references it on line (M056).

2247 Lines (M043) – (M059) contain the message signature. Lines (M054) – (M058) is its KeyInfo block that  
 2248 indicates the second derived key token should be used to verify the message signature.

2249 Line (M057) indicates the message timestamp is included in the signature as required by the  
 2250 IncludeTimestamp assertion definition.

2251 Line (M060) indicates the message body is included in the signature as required by the SignedParts  
 2252 assertion of the input message policy.

2253 Lines (M060) – (M073) contain the endorsing signature. It signs the message signature, referenced on  
 2254 line (M063), per EndorsingSupportingTokens assertion policy requirement on lines (P031) – (P040). Line  
 2255 (M070), the KeyIdentifier, has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is  
 2256 required by the policy line (P035) of the Thumbprint Reference. This Thumbprint Reference must match  
 2257 the Thumbprint of the X509 Certificate contained in the BinarySecurityToken on line (M019), based on the  
 2258 IncludeToken requirement line (P033).

2259

2260 An example of an output message that conforms to the above stated policy follows:

2261

```

2262 (R001) <env:Envelope xmlns:env="..." xmlns:wsu="...">
2263 (R002) <env:Header>
2264 (R003) <wsse:Security env:mustUnderstand="1" xmlns:wsse="...">
2265 (R004) <wsu:Timestamp wsu:Id="Timestamp" >
2266 (R005) <wsu:Created>2007-03-31T04:27:25Z</wsu:Created>
2267 (R006) </wsu:Timestamp>
2268 (R007) <wsc:DerivedKeyToken wsu:Id="derivedKeyToken1" xmlns:wsc=".../sc">
2269 (R008) <wsse:SecurityTokenReference>
2270 (R009) <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2271 (R010) >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2272 (R011) </wsse:SecurityTokenReference>
2273 (R012) <wsc:Offset>0</wsc:Offset>
2274 (R013) <wsc:Length>24</wsc:Length>
2275 (R014) <wsc:Nonce>NfSNXZLYAA8mocQz19KWjg==</wsc:Nonce>
2276 (R015) </wsc:DerivedKeyToken>
2277 (R016) <wsc:DerivedKeyToken wsu:Id="derivedKeyToken2" xmlns:wsc=".../sc">
2278 (R017) <wsse:SecurityTokenReference>
2279 (R018) <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2280 (R019) >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2281 (R020) </wsse:SecurityTokenReference>
2282 (R021) <wsc:Nonce>1F/CtyQ9d1Ro8E3+uZYmgQ==</wsc:Nonce>
2283 (R022) </wsc:DerivedKeyToken>
2284 (R023) <enc:ReferenceList xmlns:enc=".../xmlenc#">
2285 (R024) <enc:DataReference URI="#encBody"/>
2286 (R025) </enc:ReferenceList>
2287 (R026) <wsse11:SignatureConfirmation wsu:Id="sigconf1"
2288 (R027) Value="abcdefg=" xmlns:wsse11="..."/>

```

```

2289 (R028) <wssell:SignatureConfirmation wsu:Id="sigconf2"
2290 (R029) Value="hijklmnop=" xmlns:wssell="..."/>
2291 (R030) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
2292 (R031) <SignedInfo>
2293 (R032) <CanonicalizationMethod
2294 (R033) Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2295 (R034) <SignatureMethod
2296 (R035) Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
2297 (R036) <Reference URI="#msgBody">
2298 (R037) ...
2299 (R038) </Reference>
2300 (R039) <Reference URI="#Timestamp">
2301 (R040) ...
2302 (R041) </Reference>
2303 (R042) <Reference URI="#sigconf1">
2304 (R043) ...
2305 (R044) </Reference>
2306 (R045) <Reference URI="#sigconf2">
2307 (R046) ...
2308 (R047) </Reference>
2309 (R048) </SignedInfo>
2310 (R049) <SignatureValue>3rAxsfJ2LjF7liRQX2EH/0DBmzE=</SignatureValue>
2311 (R050) <KeyInfo>
2312 (R051) <wsse:SecurityTokenReference>
2313 (R052) <wsse:Reference URI="#derivedKeyToken1" />
2314 (R053) </wsse:SecurityTokenReference>
2315 (R054) </KeyInfo>
2316 (R055) </Signature>
2317 (R056) </wsse:Security>
2318 (R057) </env:Header>
2319 (R058) <env:Body wsu:Id="msgBody">
2320 (R059) <enc:EncryptedData Id="encBody" Type="...xmlenc#Content"
2321 (R060) xmlns:enc=".../xmlenc#">
2322 (R061) <enc:EncryptionMethod Algorithm=".../xmlenc#aes256-cbc" />
2323 (R062) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
2324 (R063) <wsse:SecurityTokenReference xmlns:wsse="...">
2325 (R064) <wsse:Reference URI="#derivedKeyToken2" />
2326 (R065) </wsse:SecurityTokenReference>
2327 (R066) </KeyInfo>
2328 (R067) <enc:CipherData>
2329 (R068) <enc:CipherValue>y+eV...pu</enc:CipherValue>
2330 (R069) </enc:CipherData>
2331 (R070) </enc:EncryptedData>
2332 (R071) </env:Body>
2333 (R072) </env:Envelope>
2334

```

2335 Lines (R001)-(R072) contain the Response message returned to the Initiator.

2336 Lines (R004)-(R006) contain the Timestamp required by the Policy (P027).

2337 Lines (R007)-(R015) and (R016)-(R022) contain the information necessary for the Initiator to derive the  
2338 keys using the WS-SecureConversation shared secret, the identified key (R010) for DK1 and (R019) for  
2339 DK2, which reference the same key, and the Nonce (R014) for DK1 and (R021) for DK2, which are  
2340 different for the two derived keys.

2341 Lines (R023)-(R025) contain a ReferenceList that indicates the message Body (R058) contains the data  
2342 to be encrypted. The encryption was required by the output policy (P079).

2343 Lines (R026)-(R027) contain the SignatureConfirmation for the SignatureValue in the request message  
2344 (M054) which was required to be included by the policy (P047) RequireSignatureConfirmation.

2345 Lines (R028)-(R029) contain the SignatureConfirmation for the 2<sup>nd</sup> SignatureValue in the request  
2346 message (M068), which is also required by the policy (P047).

2347 Lines (R030)-(R055) contain the response message signature that covers the Timestamp element  
2348 (R039)->(R004) required to be signed by the policy (P027), the two SignatureConfirmation elements  
2349 (R042)->(R026) and (R045)->(R028), which are required to be signed because they are required  
2350 elements of the policy (P047), and the message Body (R036)->(R058), which is required to be signed by  
2351 the output message policy (P076). The signature may be verified using derivedKeyToken1 as indicated  
2352 on line (R052).

2353 Lines (R059)-(R070) contain the encrypted data as required by the policy (P079) and may be decrypted  
2354 using derivedKeyToken2 as indicated on line (R064).

## 2355 2.3 SAML Token Authentication Scenario Assertions

2356 For SAML, the combination of SAML and WSS version numbers is supported (WssSamlV11Token10,  
2357 WssSamlV11Token11, WssSamlV20Token11).

2358 Instead of explicitly including the SAML Assertion, a wsse:KeyIdentifier reference can also be used. To  
2359 enable this last behavior, the IncludeToken attribute is set to [http://docs.oasis-open.org/ws-sx/ws-  
2360 securitypolicy/200702/IncludeToken/Never](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never).

2361 In all of the SAML scenarios, the SAML Assertion ConfirmationMethod expected to be used by the  
2362 Initiator is communicated implicitly by the context of the sp: security binding in use and type of sp:  
2363 element containing the SamlToken. There are 3 general SAML ConfirmationMethod use cases covered:  
2364 holder-of-key (hk), sender-vouches (sv), and bearer (br).

2365

2366 For the **holder-of-key** case, there is always a contained reference (or value) in the SAML  
2367 Assertion to key material that may be used for message protection in the scenario. The hk case  
2368 can be assumed if the WssSamlV\*\*Token\*\* element appears either in the sp:InitiatorToken  
2369 element of the sp:AsymmetricBinding element, or in the sp:ProtectionToken element of the  
2370 sp:SymmetricBinding element. In the sp:TransportBinding case, if the WssSamlV\*\*Token\*\*  
2371 element appears in an sp:EndorsingSupportingTokens or sp:SignedEndorsingSupportingTokens  
2372 element, the SAML Assertion type can also be assumed to be hk.

2373 The **sender-vouches** case can be assumed if the WssSamlV\*\*Token\*\* version will always  
2374 appear in an sp:SignedSupportingTokens element to indicate that the SAML Authority associated  
2375 with this token is the Initiator that signs the message.

2376 The **bearer** case can be assumed if the WssSamlV\*\*Token\*\* version appears in an  
2377 sp:SupportingTokens element (it may be signed as a SignedPart, but it is not involved in the  
2378 message protection).

2379

2380 It is recognized that other uses cases might exist, where these guidelines might need further elaboration  
2381 in order to address all contingencies, however that is outside the scope of the current version of this  
2382 document.

2383 Note: in the discussions below the term “SamlToken” or “SamlToken assertion” refers to the  
2384 policy requirement that a “SAML Assertion” be used as that token.

2385 Note: SAML Assertions have issuers. In addition, these Assertions are generally signed with an  
2386 X509 certificate. In general, the SAML IssuingAuthority may be regarded as the Subject of the  
2387 X509 certificate, which is trusted by a RelyingParty. In general, as well, there is usually a known  
2388 mapping between the Issuer identified within the SAML Assertion and the Subject of the X509  
2389 certificate used to sign that Assertion. It will be assumed in this document that the SAML  
2390 IssuingAuthority may be identified by either of these identities and that, in general, a RelyingParty  
2391 will check the details as necessary.

2392 The concept of the sp:“message signature” within the context of the **sp:TransportBinding** is not clearly  
2393 identified within the current version of [[WS-SecurityPolicy](#)], however, there are certain inferences that are  
2394 used in the use cases that follow that are identified here for reference.

- 2395 • Based on the diagrams in section 8 of [[WS-SecurityPolicy](#)], which show messages with a  
2396 “message signature” followed by a diagram showing use of transport security the only difference  
2397 is that the message signature diagrams contain a block labelled “Sig1”, which is the message  
2398 signature, and the transport security diagrams do not have this block.
- 2399 • Considering the fact that when [[SSL](#)] Client Certificates are used for client authentication, that the  
2400 client uses the client certificate private key to sign hash of SSL handshake messages in an SSL  
2401 CertificateVerify message that is part of the SSL protocol, the assumption is made that  
2402 subsequent data sent by the client on this link is effectively protected for the purposes of data

- 2403 integrity and confidentiality, and that the data sent on the link is authorized to be sent by the  
2404 holder of the private key associated with the client certificate.
- 2405 • Based on the considerations in the bullets above, and with intention of maintaining functional  
2406 consistency with the WS-SecurityPolicy notions of sp: message signature,  
2407 sp:SignedSupportingTokens, and sp:SignedEndorsingSupportingTokens, use of client certificates  
2408 with SSL will be considered equivalent to having a “virtual message signature” provided by the  
2409 Initiator’s client certificate which covers all the data in the SOAP request that the Initiator sends  
2410 on the SSL link.
  - 2411 • As such, in the above context, the client certificate may be regarded as providing both a virtual  
2412 Signing function for tokens and Timestamp that appear in the wsse:Security header, as well as a  
2413 virtual Endorsing function for tokens that contain the client certificate or a reference to the client  
2414 certificate that appear in the wsse:Security header.

2415 The net effect of the above assumptions for the SAML use cases in this section that use the  
2416 **sp:TransportBinding** is that if a **client certificate is required** to be used with the **sp:HttpsToken**  
2417 assertion then:

- 2418 1. A SAML **sender-vouches** Assertion contained in a wsse:Security header block may be  
2419 considered to be an **sp:SignedSupportingToken** when used in an sp:TransportBinding  
2420 using an sp:HttpsToken assertion that contains an sp:RequireClientCertificate assertion,  
2421 because the client certificate is being used as the IssuingAuthority behind the SAML  
2422 sender-vouches Assertion, which requires the IssuingAuthority to sign the combined  
2423 message and Assertion.
- 2424 2. A SAML **holder-of-key** Assertion contained in a wsse:Security header block may be  
2425 considered to be an **sp:SignedEndorsingSupportingToken** when used in an  
2426 sp:TransportBinding using an sp:HttpsToken assertion that contains an  
2427 sp:RequireClientCertificate assertion AND that the either the client certificate or a  
2428 reference to it is contained in the saml:SubjectConfirmationData/ds:KeyInfo element of  
2429 the SAML holder-of-key Assertion.
- 2430 3. A SAML **bearer** Assertion contained in a wsse:Security header block may only be  
2431 considered to be an **sp:SupportingToken**, because they are only “incidentally” covered  
2432 by the virtual message signature as a “pass through” token provided by the Requestor to  
2433 be evaluated by the Recipient that is being “passed through” by the Initiator, but which  
2434 the Initiator takes no responsibility.

2435 Ultimately, the responsibilities associated with the granting access to resources is determined by the  
2436 agreements between RelyingParties and IssuingAuthorities. Those agreements need to take into  
2437 consideration the security characteristics of the bindings which support access requests as to what kind  
2438 of bindings are required to “adequately protect” the requests for the associated business purposes. These  
2439 examples are intended to show how SAML Assertions can be used in a variety of WS-SecurityPolicy  
2440 contexts and how the different SAML token types (hk, sv, bearer) may be used in different configurations  
2441 relating the IssuingAuthority, the Requestor, the Initiator, the Recipient, and the RelyingParty.

## 2442 **2.3.1 WSS 1.0 SAML Token Scenarios**

### 2443 **2.3.1.1 (WSS1.0) SAML1.1 Assertion (Bearer)**

2444 Initiator adds a SAML assertion (bearer) representing the Requestor to the SOAP security header.

2445 Since the SamlToken is listed in the SupportingTokens element, it will not explicitly be covered by a  
2446 message signature. The Initiator may infer that a Saml Bearer Assertion is acceptable to meet this  
2447 requirement, because the Initiator is not required to explicitly cover a SupportingToken with a signature.

2448 The SAML assertion itself could be signed. The IssuingAuthority in this scenario is the Issuer (and signer,  
2449 if the Assertion is signed) of the SAML Assertion. The Initiator simply passes the token through and is not  
2450 actively involved in the trust relationship between the IssuingAuthority that issued the SAML Assertion  
2451 and the Requestor who is the Subject of the SAML Assertion.

2452 This scenario might be used in a SAML Federation application where a browser-based user with a SAML  
2453 Assertion indicating that user's SSO (Single Sign On) credential has submitted a request to a portal using  
2454 the Assertion as a credential (either directly or indirectly via a SAML Artifact [SAML11]), and the portal as  
2455 Initiator is generating a web service request on behalf of this user, but the trust that the Recipient has for  
2456 the Requestor is based on the Assertion and its IssuingAuthority, not the Initiator who delivered the  
2457 request.

2458

```
2459 (P001) <wsp:Policy>  
2460 (P002)   <sp:SupportingTokens>  
2461 (P003)   <wsp:Policy>  
2462 (P004)     <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
2463         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
2464 (P005)       <wsp:Policy>  
2465 (P006)         <sp:WssSamlV11Token10/>  
2466 (P007)       </wsp:Policy>  
2467 (P008)     </sp:SamlToken>  
2468 (P009)   </wsp:Policy>  
2469 (P010) </sp:SupportingTokens>  
2470 (P011) </wsp:Policy>
```

2471

2472 Lines (P001)-(P011) contain the endpoint wsp:Policy, which contains no bindings because none is  
2473 required to access the service protected by this policy, however the service does require that the  
2474 Requestor provide a Supporting Token in the sp:SupportingTokens element (P002)-(P010), which must  
2475 be an sp:SamlToken (P004)-(P008), which must be an sp:WssSamlV11Token10 (P006), which means  
2476 that the SamlToken must be a SAML 1.1 saml:Assertion that is sent in compliance with the  
2477 [WSS10-SAML11-PROFILE].

2478 As explained in section 2.3 above, the fact that the sp:SamlToken is simply in an sp:SupportingTokens  
2479 element indicates to the Initiator that a SAML bearer Assertion is what is expected to accompany the  
2480 request.

2481 The following is a sample request taken from [WSS11-SAML1120-PROFILE] that complies with the  
2482 WSS 1.0 compatible policy above

2483

```
2484 (M001) <S12:Envelope xmlns:S12="...">  
2485 (M002)   <S12:Header>  
2486 (M003)     <wsse:Security xmlns:wsse="...">  
2487 (M004)       <saml:Assertion xmlns:saml="..."  
2488 (M005)         AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"  
2489 (M006)         IssueInstant="2003-04-17T00:46:02Z"  
2490 (M007)         Issuer="www.opensaml.org"  
2491 (M008)         MajorVersion="1"  
2492 (M009)         MinorVersion="1">  
2493 (M010)       <saml:AuthenticationStatement>  
2494 (M011)         <saml:Subject>  
2495 (M012)           <saml:NameIdentifier  
2496 (M013)             NameQualifier="www.example.com"  
2497 (M014)             Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">  
2498 (M015)               uid=joe,ou=people,ou=saml-demo,o=baltimore.com  
2499 (M016)             </saml:NameIdentifier>  
2500 (M017)           <saml:SubjectConfirmation>  
2501 (M018)             <saml:ConfirmationMethod>  
2502 (M019)               urn:oasis:names:tc:SAML:1.0:cm:bearer  
2503 (M020)             </saml:ConfirmationMethod>  
2504 (M021)           </saml:SubjectConfirmation>  
2505 (M022)         </saml:Subject>  
2506 (M023)       </saml:AuthenticationStatement>  
2507 (M024)     </saml:Assertion>  
2508 (M025)   </wsse:Security>  
2509 (M026) </S12:Header>  
2510 (M027) <S12:Body>
```

```
2511 (M028) . . .
2512 (M029) </S12:Body>
2513 (M030) </S12:Envelope>
```

2514

2515 Lines (M001)-(M030) contains the SOAP:Envelope, which contains the SOAP:Header (M002)-(M026)  
2516 and the SOAP:Body (M027)-(M029).

2517 The SOAP Header contains a wsse:Security header block (M003)-(M025) which simply contains the  
2518 saml:Assertion.

2519 The saml:Assertion (M004)-(M024) is Version 1.1 (M008)-(M009), which is required by the policy  
2520 sp:WssSamlV11Token10 assertion (P006). The Assertion contains a saml:AuthenticationStatement  
2521 (M010)-(M023) that contains a saml:NameIdentifier (M012)-(M016) that identifies the saml:Subject, who is  
2522 the Requestor (who submitted the request from a browser) and it contains a saml:SubjectConfirmation  
2523 element (M017)-(M021), which specifies the saml:ConfirmationMethod to be "bearer" (M019), which  
2524 meets the policy requirement (P006).

2525 For general context to relate this scenario to Figure 1, the Requestor at a browser will have obtained  
2526 either the saml:Assertion above or an Artifact identifying that Assertion from an IssuingAuthority and sent  
2527 it to the portal in the context of some request the Requestor is trying to make. The portal recognizes that  
2528 to meet the needs of this request that it must invoke a web service that has publicized the policy above  
2529 (P001)-(P011). Therefore, the portal will now take on the role of Initiator (Fig 1) and assemble a SOAP  
2530 request (M001)-(M030) and submit it to the service as described in detail above.

### 2531 **2.3.1.2 (WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL**

2532 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #2](#)  
2533 section 4.4.4]

2534 Initiator adds a SAML Assertion (sv) to the SOAP Security Header. Because the TransportBinding  
2535 requires a Client Certificate AND the SAML token is in a SignedSupportingTokens element, when the  
2536 Initiator uses the Client Certificate to protect the message under SSL, the Initiator may be considered as  
2537 effectively "signing" the SAML sv Assertion and acting as a SAML Authority (i.e. the issuer of the sv  
2538 Assertion). By including the sv assertion in the header and using the Client Certificate to protect the  
2539 message, the Initiator takes responsibility for binding the Requestor, who is the Subject of the Assertion  
2540 to the contents of the message.

2541 Note: because SSL does not retain cryptographic protection of the message after the message is  
2542 delivered, messages protected using only this mechanism cannot be used as the basis for  
2543 non-repudiation.

2544

```
2545 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2546 wsu:Id="Wss10SamlSvV11Tran_policy">
2547 (P002) <sp:TransportBinding>
2548 (P003) <wsp:Policy>
2549 (P004) <sp:TransportToken>
2550 (P005) <wsp:Policy>
2551 (P006) <sp:HttpsToken>
2552 (P007) <wsp:Policy>
2553 (P008) <sp:RequireClientCertificate>
2554 (P009) </wsp:Policy>
2555 (P010) </sp:HttpsToken>
2556 (P011) </wsp:Policy>
2557 (P012) </sp:TransportToken>
2558 (P013) <sp:AlgorithmSuite>
2559 (P014) <wsp:Policy>
2560 (P015) <sp:Basic256 />
2561 (P016) </wsp:Policy>
2562 (P017) </sp:AlgorithmSuite>
2563 (P018) <sp:Layout>
2564 (P019) <wsp:Policy>
2565 (P020) <sp:Strict />
```

```

2566 (P021) </wsp:Policy>
2567 (P022) </sp:Layout>
2568 (P023) <sp:IncludeTimestamp />
2569 (P024) </wsp:Policy>
2570 (P025) </sp:TransportBinding>
2571 (P026) <sp:SignedSupportingTokens>
2572 (P027) <wsp:Policy>
2573 (P028) <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
2574 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2575 (P029) <wsp:Policy>
2576 (P030) <sp:WssSamlV11Token10/>
2577 (P031) </wsp:Policy>
2578 (P032) </sp:SamlToken>
2579 (P033) </wsp:Policy>
2580 (P034) </sp:SignedSupportingTokens>
2581 (P035) </wsp:Policy>

```

2582  
2583 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
2584 by a secure transport protocol such as SSL or TLS.

2585 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
2586 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
2587 transport token using the Basic256 algorithm suite (P015).

2588 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of  
2589 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section  
2590 4.3.1].

2591 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)  
2592 since there are no cryptographic tokens required to be present in the WS-Security header.

2593 A timestamp (P023) is required to be included in an acceptable message.

2594 Lines (P026)-(P034) contain a SignedSupportingTokens assertion, which indicates that the referenced  
2595 token is effectively "signed" by the combination of usage of the client certificate for SSL authentication  
2596 and the cryptographic protection of SSL. However, the "signed" characteristic will not be present after the  
2597 message is delivered from the transport layer to the recipient.

2598 Lines (P028)-(P032) indicate the signed token is a SAML Assertion (sender-vouches as described above  
2599 in section 2.3) and on line (P030) that it is a SAML 1.1 Assertion and that the WS-Security 1.0 SAML  
2600 Profile [[WSS10-SAML11-PROFILE](#)] is being used.

2601 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP](#) "Scenario #2 -  
2602 Sender-Vouches: Unsigned: SSL" section 4.4.4]

2603 Here is the example request from that scenario:

```

2604 (M001) <?xml version="1.0" encoding="utf-8" ?>
2605 (M002) <soap:Envelope
2606 (M003)   xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
2607 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2608 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2609 (M006)   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401
2610 (M007) -wss-wssecurity-secext-1.0.xsd"
2611 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
2612 (M009) -wssecurity-utility-1.0.xsd">
2613 (M010) <soap:Header>
2614 (M011)   <wss:Security soap:mustUnderstand="1">
2615 (M012)     <wsu:Timestamp>
2616 (M013)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2617 (M014)     </wsu:Timestamp>
2618 (M015)     <saml:Assertion
2619 (M016)       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2620 (M017)       MajorVersion="1" MinorVersion="1"
2621 (M018)       AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
2622 (M019)       Issuer=www.opensaml.org

```

```

2623 (M020) IssueInstant="2002-06-19T16:58:33.173Z">
2624 (M021) <saml:Conditions
2625 (M022) NotBefore="2002-06-19T16:53:33.173Z"
2626 (M023) NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2627 (M024) <saml:AuthenticationStatement
2628 (M025) AuthenticationMethod=
2629 (M026) "urn:oasis:names:tc:SAML:1.0:am:password"
2630 (M027) AuthenticationInstant="2002-06-19T16:57:30.000Z">
2631 (M028) <saml:Subject>
2632 (M029) <saml:NameIdentifier
2633 (M030) NameQualifier=www.example.com
2634 (M031) Format="">
2635 (M032) uid=joe,ou=people,ou=saml-demo,o=example.com
2636 (M033) </saml:NameIdentifier>
2637 (M034) <saml:SubjectConfirmation>
2638 (M035) <saml:ConfirmationMethod>
2639 (M036) urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2640 (M037) </saml:ConfirmationMethod>
2641 (M038) </saml:SubjectConfirmation>
2642 (M039) </saml:Subject>
2643 (M040) </saml:AuthenticationStatement>
2644 (M041) </saml:Assertion>
2645 (M042) </wsse:Security>
2646 (M043) </soap:Header>
2647 (M044) <soap:Body>
2648 (M045) <Ping xmlns="http://xmlsoap.org/Ping">
2649 (M046) <text>EchoString</text>
2650 (M047) </Ping>
2651 (M048) </soap:Body>
2652 (M049) </soap:Envelope>

```

2653

2654 Lines (M011)-(M042) contain the WS-Security 1.0 header required by the WssSamlV11Token10  
2655 assertion (P030).

2656 Lines (M012)-(M014) contain the wsu:Timestamp required by IncludeTimestamp assertion (P023).

2657 Lines (M015)-(M041) contain the SAML 1.1 Assertion required by the WssSamlV11Token10 assertion  
2658 (P030).

2659 Note that the additional requirements identified in the policy above are met by the SSL transport  
2660 capabilities and do not appear in any form in the SOAP message (M001)-(M049).

### 2661 **2.3.1.3 (WSS1.0) SAML1.1 Assertion (HK) over SSL**

2662 Initiator adds a SAML assertion (hk) to the SOAP Security Header. Because the TransportBinding  
2663 requires a Client Certificate AND the SAML token is in an EndorsingSupportingTokens element, the  
2664 Initiator may be considered to be authorized by the issuer of the hk SAML assertion to bind message  
2665 content to the Subject of the assertion. If the Client Certificate matches the certificate identified in the hk  
2666 assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor,  
2667 who would be the Subject of the hk assertion, to the content of the message.

2668 In this scenario, the IssuingAuthority is the issuer(signer) of the hk SAML Assertion. The Requestor is the  
2669 Subject of the Assertion and the Initiator is authorized by the Authority to bind the Assertion to the  
2670 message using the ClientCertificate identified in the SAML Assertion, which should also be used to sign  
2671 the timestamp of the message with a separate Signature included in the WS-Security header.

2672

```

2673 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2674 (P002) wsu:Id="Wss10SamlHkV11Tran_policy">>
2675 (P003) <sp:TransportBinding>
2676 (P004) <wsp:Policy>
2677 (P005) <sp:TransportToken>
2678 (P006) <wsp:Policy>
2679 (P007) <sp:HttpsToken>

```

```

2680 (P008)         <wsp:Policy>
2681 (P009)         <sp:RequireClientCertificate>
2682 (P010)         </wsp:Policy>
2683 (P011)         </sp:HttpsToken>
2684 (P012)         </wsp:Policy>
2685 (P013)         </sp:TransportToken>
2686 (P014)         <sp:AlgorithmSuite>
2687 (P015)         <wsp:Policy>
2688 (P016)         <sp:Basic256 />
2689 (P017)         </wsp:Policy>
2690 (P018)         </sp:AlgorithmSuite>
2691 (P019)         <sp:Layout>
2692 (P020)         <wsp:Policy>
2693 (P021)         <sp:Strict />
2694 (P022)         </wsp:Policy>
2695 (P023)         </sp:Layout>
2696 (P024)         <sp:IncludeTimestamp />
2697 (P025)         </wsp:Policy>
2698 (P026)         </sp:TransportBinding>
2699 (P027)         <sp:SignedEndorsingSupportingTokens>
2700 (P028)         <wsp:Policy>
2701 (P029)         <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2702 open.org/ws-sx/ws-
2703 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2704 (P030)         <wsp:Policy>
2705 (P031)         <sp:WssSamlV11Token10/>
2706 (P032)         </wsp:Policy>
2707 (P033)         </sp:SamlToken>
2708 (P034)         </wsp:Policy>
2709 (P035)         </sp:SignedEndorsingSupportingTokens>
2710 (P036)         <wsp:Policy>

```

2711

2712 Section 2.3.2.3 contains an example message that is similar to one that could be used for the policy  
2713 above, except the Signed Endorsing Supporting Token there is a SAML Version 2.0 token, and a SAML  
2714 Version 1.1 token would be required here.

#### 2715 **2.3.1.4 (WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional** 2716 **Encrypt**

2717 This scenario is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #3](#)].

2718 In this case, the SAML token is included as part of the message signature and sent only to the Recipient.  
2719 The message security is provided using X.509 certificates with both requestor and service having  
2720 exchanged these credentials via some out of band mechanism, which provides the basis of trust of each  
2721 others' certificates.

2722 In this scenario the SAML Authority is the Initiator who uses the message signature to both provide the  
2723 integrity of the message and to establish the Initiator as the SAML Authority based on the X509 certificate  
2724 used to sign the message and the SignedSupportingTokens SAML Assertion. Effectively, the SAML  
2725 Assertion is being "issued" as part of the message construction process. The Requestor is the Subject of  
2726 the SAML Assertion. The Initiator knows that the Recipient is expecting it to be the SAML Authority by the  
2727 fact that the policy specifies that the message requires a SignedSupportingTokens SamlToken.

2728

```

2729 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2730 (P002) wsu:Id="Wss10SamlSvV11Asymm_endpoint_policy">
2731 (P003) <wsp:ExactlyOne>
2732 (P004) <wsp>All>
2733 (P005) <sp:AsymmetricBinding>
2734 (P006) <wsp:Policy>
2735 (P007) <sp:InitiatorToken>
2736 (P008) <wsp:Policy>

```

```

2737 (P009) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2738 open.org/ws-sx/ws-
2739 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2740 <wsp:Policy>
2741 (P011) <sp:WssX509V3Token10/>
2742 (P012) </wsp:Policy>
2743 (P013) </sp:X509Token>
2744 (P014) </wsp:Policy>
2745 (P015) </sp:InitiatorToken>
2746 (P016) <sp:RecipientToken>
2747 (P017) <wsp:Policy>
2748 (P018) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2749 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
2750 (P019) <wsp:Policy>
2751 (P020) <sp:WssX509V3Token10/>
2752 (P021) </wsp:Policy>
2753 (P022) </sp:X509Token>
2754 (P023) </wsp:Policy>
2755 (P024) </sp:RecipientToken>
2756 (P025) <sp:AlgorithmSuite>
2757 (P026) <wsp:Policy>
2758 (P027) <sp:Basic256/>
2759 (P028) </wsp:Policy>
2760 (P029) </sp:AlgorithmSuite>
2761 (P030) <sp:Layout>
2762 (P031) <wsp:Policy>
2763 (P032) <sp:Strict/>
2764 (P033) </wsp:Policy>
2765 (P034) </sp:Layout>
2766 (P035) <sp:IncludeTimestamp/>
2767 (P036) <sp:OnlySignEntireHeadersAndBody/>
2768 (P037) </wsp:Policy>
2769 (P038) </sp:AsymmetricBinding>
2770 (P039) <sp:Wss10>
2771 (P040) <wsp:Policy>
2772 (P041) <sp:MustSupportRefKeyIdentifier/>
2773 (P042) </wsp:Policy>
2774 (P043) </sp:Wss10>
2775 (P044) <sp:SignedSupportingTokens>
2776 (P045) <wsp:Policy>
2777 (P046) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2778 open.org/ws-sx/ws-
2779 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2780 <wsp:Policy>
2781 (P048) <sp:WssSamlV11Token10/>
2782 (P049) </wsp:Policy>
2783 (P050) </sp:SamlToken>
2784 (P051) </wsp:Policy>
2785 (P052) </sp:SignedSupportingTokens>
2786 (P053) </wsp:All>
2787 (P054) </wsp:ExactlyOne>
2788 (P055) </wsp:Policy>
2789 (P056)
2790 (P057) <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_input_policy">
2791 (P058) <wsp:ExactlyOne>
2792 (P059) <wsp:All>
2793 (P060) <sp:SignedParts>
2794 (P061) <sp:Body/>
2795 (P062) </sp:SignedParts>
2796 (P063) <sp:EncryptedParts wsp:Optional="true">
2797 (P064) <sp:Body/>
2798 (P065) </sp:EncryptedParts>
2799 (P066) </wsp:All>
2800 (P067) </wsp:ExactlyOne>

```

```

2801 (P068) </wsp:Policy>
2802
2803 (P069) <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_output_policy">
2804 (P070) <wsp:ExactlyOne>
2805 (P071) <wsp:All>
2806 (P072) <sp:SignedParts>
2807 (P073) <sp:Body/>
2808 (P074) </sp:SignedParts>
2809 (P075) <sp:EncryptedParts>
2810 (P076) <sp:Body/>
2811 (P077) </sp:EncryptedParts>
2812 (P078) </wsp:All>
2813 (P079) </wsp:ExactlyOne>
2814 (P080) </wsp:Policy>

```

2815 Line (P004) is a `wsp:All` element whose scope carries down to line (P053), which means all 3 of the  
2816 contained assertions: (P005)-(P038) `AsymmetricBinding`, (P039)-(P043) `WSS10`, and (P044)-(P052)  
2817 `SignedSupportingTokens` must be complied with by message exchanges to a Web Service covered by  
2818 this policy.

2819 Lines (P005)-(P038) contain an `AsymmetricBinding` assertion which indicates that the Initiator's token  
2820 must be used for the message signature and that the recipient's token must be used for message  
2821 encryption.

2822 Lines (P007)-(P015) contain the `InitiatorToken` assertion. Within the `InitiatorToken` assertion, lines  
2823 (P009)-(P013) indicate that the Initiator's token must be an `X509Token` that must always be included as  
2824 part of the request message (as opposed to an external reference). Line (P011) indicates that the `X509`  
2825 token must be an `X.509 V3` signature-verification certificate and used in the manner described in  
2826 [\[WSS10-X509-PROFILE\]](#).

2827 Lines (P016)-(P023) contain the `RecipientToken` assertion. Again, this an `X.509 V3` certificate (P020) that  
2828 must be used as described in [\[WSS10-X509-PROFILE\]](#), however the token itself, must never be included  
2829 in messages in either direction (P018).

2830 Instead (momentarily skipping slightly ahead), policy lines (P039)-(P043), the `WSS10` assertion, indicate  
2831 that the Recipient's token must be referenced by a `KeyIdentifier` element, which is described in the `WS-`  
2832 `Security 1.0` core specification [\[WSS10-SOAPMSG\]](#).

2833 Lines (P025)-(P029) contain the `AlgorithmSuite` assertion, which specifies the `sp:Basic256` set of  
2834 cryptographic components. The relevant values for this example within the `sp:Basic256` components are  
2835 the asymmetric signing algorithm, `ds:rsa-sha1`, and the digest algorithm, `ds:sha1`, where "ds:" =  
2836 "http://www.w3.org/2000/09/xmlsig#" [\[XML-DSIG\]](#).

2837 Lines (P030)-(P034) contain the `sp:Layout` assertion, which is set to `sp:Strict` (P032), which governs the  
2838 required relative layout of various `Timestamp`, `Signature` and `Token` elements in the messages.

2839 Line (P035) `IncludeTimestamp` means a `Timestamp` element must be included in the `WS-Security` header  
2840 block and signed by the message signature for messages in both directions.

2841 Line (P036) `OnlySignEntireHeaderAndBody` indicates that the message signature must explicitly cover  
2842 the `SOAP:Body` element (not children), and if there are any signed elements in the `SOAP:Header` they  
2843 must be direct child elements of the `SOAP:Header`. However, the one exception where the latter condition  
2844 is not applied is that if the child of the `SOAP:Header` is a `WS-Security` header (i.e. a `wsse:Security`  
2845 element), then individual direct child only elements of that `Security` element may also be signed.

2846 Lines (P044)-(P052) contain a `SignedSupportingTokens` assertion, which contains only one token  
2847 (P046)-(P050), a `SamlToken`, which must always be sent to the Recipient (P046) and it must be a  
2848 `SAML 1.1 Assertion` token. Because the `SamlToken` is in a "SignedSupportingTokens" element, it is  
2849 implicitly a `SAML sender-vouches ConfirmationMethod` token as described in section 2.3 above.

2850 Lines (P057)-(P068) contain a policy that applies only to input messages from the Initiator to the  
2851 Recipient.

2852 Lines (P060)-(P062) specify that the `SOAP:Body` element of the input message must be signed by the  
2853 Initiator's message signature.

2854 Lines (P063)-(P065) specify that the SOAP:Body element of the input message may optionally be  
2855 encrypted (signified by `wsp:Optional="true"`) using the Recipient's encryption token (in this case (P020), it  
2856 is an X.509 certificate).

2857 Lines (P069)-(P080) contain a policy that applies only to output messages from the Recipient to the  
2858 Initiator.

2859 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be signed by the  
2860 Recipient's message signature.

2861 Lines (P075)-(P077) specify that the SOAP:Body element of the output message must be encrypted by  
2862 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

2863 Here is an example request:

2864

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
2865 (M002) <S12:Envelope
2866 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
2867 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2868 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2869 (M006)   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-
2870 (M007) 200401-wss-wssecurity-secext-1.0.xsd"
2871 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
2872 (M009) 200401-wss-wssecurity-utility-1.0.xsd"
2873 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#
2874 (M011)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
2875 (M012) <S12:Header>
2876 (M013)   <wss:Security S12:mustUnderstand="1">
2877 (M014)     <wsu:Timestamp wsu:Id="timestamp">
2878 (M015)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2879 (M016)     </wsu:Timestamp>
2880 (M017)     <saml:Assertion
2881 (M018)       AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2882 (M019)       IssueInstant="2003-04-17T00:46:02Z"
2883 (M020)       Issuer=www.opensaml.org
2884 (M021)       MajorVersion="1"
2885 (M022)       MinorVersion="1">
2886 (M023)       <saml:Conditions
2887 (M024)         NotBefore="2002-06-19T16:53:33.173Z"
2888 (M025)         NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2889 (M026)       <saml:AttributeStatement>
2890 (M027)         <saml:Subject>
2891 (M028)           <saml:NameIdentifier
2892 (M029)             NameQualifier=www.example.com
2893 (M030)             Format="">
2894 (M031)             uid=joe,ou=people,ou=saml-demo,o=example.com
2895 (M032)           </saml:NameIdentifier>
2896 (M033)           <saml:SubjectConfirmation>
2897 (M034)             <saml:ConfirmationMethod>
2898 (M035)               urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2899 (M036)             </saml:ConfirmationMethod>
2900 (M037)           </saml:SubjectConfirmation>
2901 (M038)         </saml:Subject>
2902 (M039)         <saml:Attribute>
2903 (M040)           ...
2904 (M041)         </saml:Attribute>
2905 (M042)         ...
2906 (M043)       </saml:AttributeStatement>
2907 (M044)     </saml:Assertion>
2908 (M045)     <wss:SecurityTokenReference wsu:id="STR1">
2909 (M046)       <wss:KeyIdentifier wsu:id="..."
2910 (M047)         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
2911 \(M048\) 2004XXwss-saml-token-profile-1.0#SAMLAssertionID">
2912 (M048)       _a75adf55-01d7-40cc-929f-dbd8372ebdfc
2913 (M049)     </wss:KeyIdentifier>
2914
```

```

2915 (M050) </wsse:SecurityTokenReference>
2916 (M051) <wsse:BinarySecurityToken
2917 (M052)   wsu:Id="attesterCert"
2918 (M053)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2919 (M054) 200401-wss-x509-token-profile-1.0#X509v3"
2920 (M055)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
2921 (M056) 200401-wss-soap-message-security-1.0#Base64Binary">
2922 (M057)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
2923 (M058) </wsse:BinarySecurityToken>
2924 (M059) <ds:Signature>
2925 (M060)   <ds:SignedInfo>
2926 (M061)     <ds:CanonicalizationMethod
2927 (M062)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2928 (M063)     <ds:SignatureMethod
2929 (M064)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2930 (M065)     <ds:Reference URI="#STR1">
2931 (M066)       <ds:Transforms>
2932 (M067)         <ds:Transform
2933 (M068)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
2934 (M069) 200401-wss-soap-message-security-1.0#STR-Transform">
2935 (M070)           <wsse:TransformationParameters>
2936 (M071)             <ds:CanonicalizationMethod
2937 (M072)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2938 (M073)             </wsse:TransformationParameters>
2939 (M074)           </ds:Transform>
2940 (M075)         </ds:Transforms>
2941 (M076)       <ds:DigestMethod
2942 (M077)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2943 (M078)       <ds:DigestValue>...</ds:DigestValue>
2944 (M079)     </ds:Reference>
2945 (M080)     <ds:Reference URI="#MsgBody">
2946 (M081)       <ds:DigestMethod
2947 (M082)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2948 (M083)       <ds:DigestValue>...</ds:DigestValue>
2949 (M084)     </ds:Reference>
2950 (M085)     <ds:Reference URI="#timestamp">
2951 (M086)       <ds:DigestMethod
2952 (M087)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2953 (M088)       <ds:DigestValue>...</ds:DigestValue>
2954 (M089)     </ds:Reference>
2955 (M090)     </ds:SignedInfo>
2956 (M091)     <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
2957 (M092)   <ds:KeyInfo>
2958 (M093)     <wsse:SecurityTokenReference wsu:id="STR2">
2959 (M094)       <wsse:Reference wsu:id="..." URI="#attesterCert" />
2960 (M095)     </wsse:SecurityTokenReference>
2961 (M096)   </ds:KeyInfo>
2962 (M097) </ds:Signature>
2963 (M098) </wsse:Security>
2964 (M099) </S12:Header>
2965 (M100) <S12:Body wsu:Id="MsgBody">
2966 (M101)   <ReportRequest>
2967 (M102)     <TickerSymbol>SUNW</TickerSymbol>
2968 (M103)   </ReportRequest>
2969 (M104) </S12:Body>
2970 (M105) </S12:Envelope>

```

2971

2972 Note: For the instructional purposes of this document, in order to be compliant with WS-  
2973 SecurityPolicy, this copy of the original message had to be modified from the original. In particular,  
2974 the modifications that are applied above are the inclusion of a wsu:Id attribute in the Timestamp  
2975 element start tag (M014) and the addition of a ds:Reference element and URI attribute identifying  
2976 that Timestamp element in the message signature (M085)-(M089). (The original message in

2977 [WSS10-SAML11-INTEROP scenario #3] is not compliant with WS-SecurityPolicy because the  
2978 Timestamp that it contains is not covered by the message signature in that document.)

2979 Lines (M002)-(M0105) contain the SOAP:Envelope, i.e. the whole SOAP message.

2980 Lines (M012)-(M099) contain the SOAP:Header, which is the header portion of the SOAP message.

2981 Lines (M0100)-(M0104) contain the SOAP:Body, which is just some dummy content for test purposes.

2982 Lines (M013)-(M098) contain the wsse:Security header, which is the primary focus of this example.

2983 Lines (M014)-(M016) contain the wsu:Timestamp, which is required by the policy (P035).

2984 Lines (M017)-(M044) contain the SAML Assertion, which is the sp:Sam1Token required by the policy  
2985 sp:SignedSupportingTokens (P046)-(P050).

2986 Lines (M021)-(M022) identify the SAML Assertion as being Version 1.1 as required by the policy (P048).

2987 Line (M035) identifies the SAML Assertion as having ConfirmationMethod sender-vouches, which is the  
2988 implicit requirement of the sp:Sam1Token being in the SignedSupportingTokens as described above in the  
2989 policy section covering (P044)-(P052).

2990 Lines (M045)-(M050) contain a wsse:SecurityTokenReference which contains a wsse:KeyIdentifier, which  
2991 is used to reference the SAML Assertion, as described in [WSS10-SAML11-PROFILE] and required by  
2992 the policy (P041).

2993 Lines (M051)-(M058) contain the Initiator's wsse:BinarySecurityToken, which is required by the policy  
2994 (P007)-(P015), and in particular lines (M053)-(M054) identify the token as an X.509 V3 token as required  
2995 by the policy (P011). Line (M057) contains a truncated portion of the Base64 representation of the actual  
2996 certificate, which is included in the message as required by the policy sp:IncludeToken (P009).

2997 Lines (M059)-(M0102) contain the ds:Signature, which is the sp: message signature as required by  
2998 various parts of the Endpoint Policy (P001)-(P055) and Input Message Policy (P057)-(P068).

2999 Lines (M060)-(M095) contain the ds:SignedInfo element which describes the signing algorithm used  
3000 (M064) and specific elements that are signed (M065)-(M094) as required by the policies, which are  
3001 described in detail immediately following.

3002 Lines (M061)-(M062) contain the ds:CanonicalizationMethod Algorithm, which is specified as xml-exc-  
3003 c14n#, which is the default value required by the policy sp:AlgorithmSuite (P025)-(P029).

3004 Line (M064) identifies the SignatureMethod Algorithm as ds:rsa-sha1, which is the asymmetric key  
3005 signing algorithm required by the policy sp:AlgorithmSuite (P025)-(P029).

3006 Lines (M065)-(M079) identify the SAML Assertion (M017)-(M044) as an element that is signed, which is  
3007 referenced through the URI "#STR1" on line (M065), which references the SecurityTokenReference  
3008 (M045)-(M050), which in turn uses the identifier on line (M048) to reference the actual SAML Assertion by  
3009 its AssertionID attribute on line (M018). The SAML Assertion is required to be signed because it is  
3010 identified in the policy within the SignedSupportingTokens element on line (P048).

3011 Lines (M077)-(M078) contain the digest algorithm, which is specified to be sha1, as required by the policy  
3012 sp:Basic256 assertion (P027) in the sp:AlgorithmSuite.

3013 Lines (M080)-(M084) identify the SOAP:Body (M0100)-(M0104) as an element that is signed, which is  
3014 referenced through the URI "#MsgBody" on line (M080). The SOAP Body is required to be signed by the  
3015 input message policy lines (P060)-(P062).

3016 Lines (M085)-(M089) identify the wsu:Timestamp (M014)-(M016) as an element that is signed, which is  
3017 referenced through the URI "#timestamp" on line (M085). The wsu:Timestamp is required to be signed by  
3018 the policy sp:IncludeTimestamp assertion (P035). Note that the wsu:Timestamp occurs in the  
3019 wsse:Security header prior to the ds:Signature as required by the sp:Strict layout policy (P032).

3020 Finally, lines (M092)-(M096) contain the ds:KeyInfo element that identifies the signing key associated with  
3021 this ds:Signature element. The actual signing key is referenced through the  
3022 wsse:SecurityTokenReference (M093)-(M095), with URI "#attesterCert", which identifies the  
3023 InitiatorToken identified by the policy (P011) and contained in the wsse:BinarySecurityToken element  
3024 (M051)-(M058), which occurs in the wsse:Security header prior to this ds:Signature element, as required  
3025 by the sp:Strict layout policy assertion (P032). (Note: this BinarySecurityToken would need to be included

3026 in the signature, if the sp:AsymmetricBinding assertion in the endpoint policy contained a  
3027 sp:ProtectTokens assertion, but it does not, so it does not need to be signed.)

### 3028 2.3.1.5 (WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt

3029 This example is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #4](#)].

3030 In this example the SAML token provides the key material for message security, hence acts as the  
3031 Initiator token, and therefore the Requestor and Initiator may be considered to be the same entity. The  
3032 SAML HK Assertion contains a reference to the public key of the signer of the message (the Initiator). The  
3033 Initiator knows Recipient's public key, which it may use to encrypt the request, but the Initiator does not  
3034 share a direct trust relation with the Recipient except indirectly through the SAML Assertion Issuer. The  
3035 Recipient has a trust relation with the Authority that issues the SAML HK Assertion. The indirect trust  
3036 relation between the Recipient and the Initiator is established by the fact that the Initiator's public key has  
3037 been signed by the Authority in the SAML HK Assertion. On the request the message body is signed  
3038 using Initiator's private key referenced in the SAML HK Assertion and it is optionally encrypted using  
3039 Recipient's server certificate. On the response, the server signs the message using its private key and  
3040 encrypts the message using the key provided within SAML HK Assertion.

3041 HK Note: there is a trust model aspect to the WS-Security holder-of-key examples that  
3042 implementors may want to be aware of. The [[SAML20-CORE](#)] specification defines the Subject of  
3043 the SAML Assertion such that the "presenter" of the Assertion is the entity that "attests" to the  
3044 information in the Assertion. "The attesting entity and the actual Subject **may or may not** be the  
3045 same entity." In general, these two choices map to the actors in [Figure 1](#) as follows: the  
3046 "attesting" entity may be regarded as the Initiator. The Subject entity, about whom the information  
3047 in the Assertion applies, may be regarded as the Requestor. In the case where the Subject and  
3048 attestor are one and the same, the Initiator and Requestor may be regarded as one and the  
3049 same. In this case the potential exists to use the Assertion for non-repudiation purposes with  
3050 respect to messages that the Requestor/Initiator signs. When the Subject and attestor are  
3051 separate entities, then holder-of-key is more similar to sender-vouches where the Initiator/attestor  
3052 is sending the message on behalf of the Subject. The mechanisms for determining whether the  
3053 Subject and attestor may be verified to be the same entity are dependent on the arrangements  
3054 among the business entities and the structure of the associated application scenarios.

3055

```
3056 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."  
3057 (P002)   wsu:Id="Wss10SamlHkV11Asymm_endpoint_policy">  
3058 (P003)   <wsp:ExactlyOne>  
3059 (P004)     <wsp>All>  
3060 (P005)       <sp:AsymmetricBinding>  
3061 (P006)         <wsp:Policy>  
3062 (P007)           <sp:InitiatorToken>  
3063 (P008)             <wsp:Policy>  
3064 (P009)               <sp:SamlToken sp:IncludeToken="http://docs.oasis-  
3065 open.org/ws-sx/ws-  
3066 securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3067 (P010)                 <wsp:Policy>  
3068 (P011)                   <wsp:WssSamlV11Token10/>  
3069 (P012)                 </wsp:Policy>  
3070 (P013)               </sp:SamlToken>  
3071 (P014)             </wsp:Policy>  
3072 (P015)           </sp:InitiatorToken>  
3073 (P016)         <sp:RecipientToken>  
3074 (P017)           <wsp:Policy>  
3075 (P018)             <sp:X509Token sp:IncludeToken="http://docs.oasis-  
3076 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
3077 (P019)               <wsp:Policy>  
3078 (P020)                 <sp:WssX509V3Token10/>  
3079 (P021)               </wsp:Policy>  
3080 (P022)             </sp:X509Token>  
3081 (P023)           </wsp:Policy>  
3082 (P024)         </sp:RecipientToken>
```

```

3083 (P025) <sp:AlgorithmSuite>
3084 (P026) <wsp:Policy>
3085 (P027) <sp:Basic256/>
3086 (P028) </wsp:Policy>
3087 (P029) </sp:AlgorithmSuite>
3088 (P030) <sp:Layout>
3089 (P031) <wsp:Policy>
3090 (P032) <sp:Strict/>
3091 (P033) </wsp:Policy>
3092 (P034) </sp:Layout>
3093 (P035) <sp:IncludeTimestamp/>
3094 (P036) <sp:OnlySignEntireHeadersAndBody/>
3095 (P037) </wsp:Policy>
3096 (P038) </sp:AsymmetricBinding>
3097 (P039) <sp:Wss10>
3098 (P040) <wsp:Policy>
3099 (P041) <sp:MustSupportRefKeyIdentifier/>
3100 (P042) </wsp:Policy>
3101 (P043) </sp:Wss10>
3102 (P044) </wsp:All>
3103 (P045) </wsp:ExactlyOne>
3104 (P046) </wsp:Policy>
3105 (P047)
3106 (P048) <wsp:Policy wsu:Id="WSS10SamlHok_input_policy">
3107 (P049) <wsp:ExactlyOne>
3108 (P050) <wsp:All>
3109 (P051) <sp:SignedParts>
3110 (P052) <sp:Body/>
3111 (P053) </sp:SignedParts>
3112 (P054) <wsp:ExactlyOne>
3113 (P055) <wsp:All>
3114 (P056) <sp:EncryptedParts>
3115 (P057) <sp:Body/>
3116 (P058) </sp:EncryptedParts>
3117 (P059) </wsp:All>
3118 (P060) <wsp:All/>
3119 (P061) </wsp:ExactlyOne>
3120 (P062) </wsp:All>
3121 (P063) </wsp:ExactlyOne>
3122 (P064) </wsp:Policy>
3123 (P065)
3124 (P066) <wsp:Policy wsu:Id="WSS10SamlHok_output_policy">
3125 (P067) <wsp:ExactlyOne>
3126 (P068) <wsp:All>
3127 (P069) <sp:SignedParts>
3128 (P070) <sp:Body/>
3129 (P071) </sp:SignedParts>
3130 (P072) <sp:EncryptedParts>
3131 (P073) <sp:Body/>
3132 (P074) </sp:EncryptedParts>
3133 (P075) </wsp:All>
3134 (P076) </wsp:ExactlyOne>
3135 (P077) </wsp:Policy>

```

3136

3137 Lines (P001)-(P046) contain the endpoint policy, and lines (P048)-(P064) and (P066)-(P077) contain the  
3138 message input and message output policies, respectively.

3139 Lines (P005)-(P038) contain the AsymmetricBinding assertion, which requires separate security tokens  
3140 for the Initiator and Recipient.

3141 Lines (P007)-(P015) contain the InitiatorToken, which is defined to be a SamlToken (P009)-(P013), which  
3142 must always be sent to the Recipient (P009). The SamlToken is further specified by the  
3143 WssSamlV11Token10 assertion (P011) that requires the token to be a SAML 1.1 Assertion and the token  
3144 must be used in accordance with the WS-Security [[WSS10-SAML11-PROFILE](#)]. Furthermore, as

3145 described in section 2.3 above, because the SamlToken assertion appears within an InitiatorToken  
3146 assertion, the SAML Assertion must use the holder-of-key ConfirmationMethod, whereby for the indicated  
3147 WS-Security profile, the SAML Assertion contains a reference to the Initiator's X.509 signing certificate or  
3148 equivalent.

3149 Lines (P016)-(P024) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that  
3150 must be used as described in [WSS10-SAML11-PROFILE], however the token itself, must never be  
3151 included in messages in either direction (P018) (i.e not only will the Recipient not send the token, but the  
3152 Initiator must not send the actual token, even when using it for encryption).

3153 Lines (P025)-(P029) AlgorithmSuite and (P030)-(P034) Layout are the same as described above in  
3154 section 2.3.1.4.

3155 Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header  
3156 block and signed by the message signature for messages in both directions.

3157 Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover  
3158 the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they  
3159 must be direct child elements of the SOAP:Header. However, the one exception where the latter condition  
3160 is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security  
3161 element), then individual direct child only elements of that Security element may also be signed.

3162

3163 Lines (P039)-(P043) contain the WSS10 assertion, which indicates that the Recipient's token must be  
3164 referenced by a KeyIdentifier element (P041), the usage of which is described in the WS-Security 1.0  
3165 core specification [[WSS10-SOAPMSG](#)].

3166 Lines (P048)-(P064) contain the message input policy that applies only to messages from the Initiator to  
3167 the Recipient.

3168 Lines (P051)-(P053) specify that the SOAP:Body element of the input message must be signed by the  
3169 Initiator's message signature.

3170 Lines (P054)-(P061) specify that the SOAP:Body element of the input message may optionally (signified  
3171 by the empty policy alternative on line (P060)) be encrypted using the Recipient's encryption token (in this  
3172 case (P020), it is an X.509 certificate).

3173 Note: Because the input policy above (P048-P064) has the EncryptedParts assertion  
3174 (P056-P058) contained in an <ExactlyOne> element (P054-P061), which also contains an empty  
3175 policy element (P060), either a message with an encrypted Body element or an unencrypted  
3176 Body element will be accepted.

3177 Lines (P066)-(P077) contain a message output policy that applies only to messages from the Recipient to  
3178 the Initiator.

3179 Lines (P069)-(P071) specify that the SOAP:Body element of the output message must be signed by the  
3180 Recipient's message signature.

3181 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be encrypted by  
3182 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

3183

3184 The following example request is taken from the [[WSS10-SAML11-INTEROP](#)] document scenario #4:

3185

```
3186 (M001) <?xml version="1.0" encoding="utf-8" ?>
3187 (M002) <S12:Envelope
3188 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
3189 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
3190 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
3191 (M006)   xmlns:wsse=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
3192 (M007)   wss-wssecurity-secext-1.0.xsd"
3193 (M008)   xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
3194 (M009)   wssecurity-utility-1.0.xsd"
3195 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#>
3196 (M011)   <S12:Header>
```

```

3197 (M012) <wsse:Security S12:mustUnderstand="1">
3198 (M013)   <wsu:Timestamp wsu:Id="timestamp">
3199 (M014)     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
3200 (M015)   </wsu:Timestamp>
3201 (M016)   <saml:Assertion
3202 (M017)     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3203 (M018)     IssueInstant="2003-04-17T00:46:02Z"
3204 (M019)     Issuer=www.opensaml.org
3205 (M020)     MajorVersion="1"
3206 (M021)     MinorVersion="1"
3207 (M022)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
3208 (M023)   <saml:Conditions
3209 (M024)     NotBefore="2002-06-19T16:53:33.173Z"
3210 (M025)     NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
3211 (M026)   <saml:AttributeStatement>
3212 (M027)     <saml:Subject>
3213 (M028)       <saml:NameIdentifier
3214 (M029)         NameQualifier=www.example.com
3215 (M030)         Format="">
3216 (M031)         uid=joe,ou=people,ou=saml-demo,o=example.com
3217 (M032)       </saml:NameIdentifier>
3218 (M033)     <saml:SubjectConfirmation>
3219 (M034)       <saml:ConfirmationMethod>
3220 (M035)         urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
3221 (M036)       </saml:ConfirmationMethod>
3222 (M037)     <ds:KeyInfo>
3223 (M038)       <ds:KeyValue>...</ds:KeyValue>
3224 (M039)     </ds:KeyInfo>
3225 (M040)   </saml:SubjectConfirmation>
3226 (M041) </saml:Subject>
3227 (M042)   <saml:Attribute
3228 (M043)     AttributeName="MemberLevel"
3229 (M044)     AttributeNamespace=
3230 (M045)       "http://www.oasis-open.org/Catalyst2002/attributes">
3231 (M046)     <saml:AttributeValue>gold</saml:AttributeValue>
3232 (M047)   </saml:Attribute>
3233 (M048)   <saml:Attribute
3234 (M049)     AttributeName="E-mail"
3235 (M050)     AttributeNamespace="http://www.oasis-
3236 (M051)       open.org/Catalyst2002/attributes">
3237 (M052)     <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
3238 (M053)   </saml:Attribute>
3239 (M054) </saml:AttributeStatement>
3240 (M055) <ds:Signature>...</ds:Signature>
3241 (M056) </saml:Assertion>
3242 (M057) <ds:Signature>
3243 (M058)   <ds:SignedInfo>
3244 (M059)     <ds:CanonicalizationMethod
3245 (M060)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
3246 (M061)     <ds:SignatureMethod
3247 (M062)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3248 (M063)     <ds:Reference URI="#MsgBody">
3249 (M064)       <ds:DigestMethod
3250 (M065)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3251 (M066)       <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
3252 (M067)     </ds:Reference>
3253 (M068)     <ds:Reference URI="#timestamp">
3254 (M069)       <ds:DigestMethod
3255 (M070)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3256 (M071)       <ds:DigestValue>...</ds:DigestValue>
3257 (M072)     </ds:Reference>
3258 (M073)   </ds:SignedInfo>
3259 (M074)   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
3260 (M075)   <ds:KeyInfo>

```

```

3261 (M076) <wsse:SecurityTokenReference wsu:id="STR1">
3262 (M077) <wsse:KeyIdentifier wsu:id="..."
3263 (M078)     ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
3264 (M079) 2004XX-wss-saml-token-profile-1.0#SAMLAssertionID">
3265 (M080)     _a75adf55-01d7-40cc-929f-dbd8372ebdfc
3266 (M081) </wsse:KeyIdentifier>
3267 (M082) </wsse:SecurityTokenReference>
3268 (M083) </ds:KeyInfo>
3269 (M084) </ds:Signature>
3270 (M085) </wsse:Security>
3271 (M086) </S12:Header>
3272 (M087) <S12:Body wsu:Id="MsgBody">
3273 (M088) <ReportRequest>
3274 (M089)   <TickerSymbol>SUNW</TickerSymbol>
3275 (M090) </ReportRequest>
3276 (M091) </S12:Body>
3277 (M092) </S12:Envelope>

```

3278  
3279 Note: for the instructional purposes of this document, it was necessary to make changes to the  
3280 original sample request to meet the requirements of WS-SecurityPolicy. The changes to the  
3281 message above include:

- 3282 • Line (M013): added wsu:Id="timestamp" attribute to sp:Timestamp element.
- 3283 • Lines (M068)-(M072): added a ds:Reference element to include the Timestamp in the  
3284 message signature required by the policy sp:IncludeTimestamp assertion (P035).

3285 Lines (M002)-(M092) contain the SOAP Envelope, i.e. the whole SOAP message.

3286 Lines (M011)-(M086) contain the SOAP Header.

3287 Lines (M087)-(M091) contain the unencrypted SOAP Body, which is allowed to be unencrypted based on  
3288 line (P060) of the input message policy, which is the alternative choice to encrypting based on lines  
3289 (P055)-(P059).

3290 Lines (M012)-(M080) contain the wsse:Security header entry, which is the only header entry in this SOAP  
3291 Header.

3292 Lines (M013)-(M015) contain the wsu:Timestamp which is required by the endpoint policy (P035).

3293 Lines (M016)-(M056) contain the SAML Assertion, which is required in the policy by the SamlToken  
3294 (P009)-(P013) contained in the InitiatorToken. The SAML Assertion is version 1.1 (M020)-(M021) as  
3295 required by the sp:WssSamlV11Token10 assertion (P011). The SAML Assertion is of type that uses the  
3296 holder-of-key saml:ConfirmationMethod [[SAML11-CORE](#)] (M034)-(M036) as required by the fact that the  
3297 SamlToken is contained in the InitiatorToken assertion of the policy (P009)-(P013), as explained in  
3298 section 2.3 above.

3299 Line (M055) contains the ds:Signature of the Issuer of the SAML Assertion. While the details of this  
3300 signature are not shown, it is this signature that the Recipient must ultimately trust in order to trust the rest  
3301 of the message.

3302 Lines (M057)-(M084) contain the message signature in a ds:Signature element.

3303 Lines (M058)-(M073) contain the ds:SignedInfo element, which identifies the elements to be signed.

3304 Lines (M063)-(M067) contains a ds:Reference to the SOAP:Body, which is signed in the same manner as  
3305 example section 2.3.1.4 above.

3306 Lines (M068)-(M072) contains a ds:Reference to the URI "timestamp", which again is signed in the same  
3307 manner as the wsu:Timestamp in section 2.3.1.4 above.

3308 Lines (M075)-(M083) contain the ds:KeyInfo element, which identifies the key that should be used to  
3309 verify this ds:Signature element. Lines (M076)-(M082) contain a wsse:SecurityTokenReference, which  
3310 contains a wsse:KeyIdentifier that identifies the signing key as being contained in a token of  
3311 wsse:ValueType "...wss-saml-token-profile-1.0#SAMLAssertionID", which means a SAML V1.1 Assertion  
3312 [[WSS10-SAML11-PROFILE](#)]. Line (M080) contains the saml:AssertionID of the SAML Assertion that  
3313 contains the signing key. This SAML Assertion is on lines (M016)-(M056) with the correct  
3314 saml:AssertionID on line (M017). Note that the fact that the referenced token (the SAML Assertion) occurs  
3315 in the wsse:Security header before this ds:KeyInfo element that uses it in compliance with the sp:Strict  
3316 layout policy (P032) and the wsse:KeyIdentifier is an acceptable token reference mechanism as specified  
3317 in the policy sp:Wss10 assertion containing a sp:MustSupportRefKeyIdentifier assertion (P041).

3318

## 3319 2.3.2 WSS 1.1 SAML Token Scenarios

3320 This section contains SamlToken examples that use WS-Security 1.1 SOAP Message Security [WSS11]  
3321 and the WS-Security 1.1 SAML Profile [WSS11-SAML1120-PROFILE].

### 3322 2.3.2.1 (WSS1.1) SAML 2.0 Bearer

3323 This example is based on the Liberty Alliance Identity Web Services Framework (ID-WSF 2.0) Security  
3324 Mechanism for the SAML 2.0 Profile for WSS 1.1 [WSS11-LIBERTY-SAML20-PROFILE], which itself is  
3325 based on the [WSS11-SAML1120-PROFILE].

3326 In this example, an AsymmetricBinding is used for message protection provided by the Initiator, however,  
3327 Recipient trust for the content is based only on the SAML bearer token provided by the Requestor.

3328

```
3329 (P001) <wsp:Policy>
3330 (P002)   <sp:AsymmetricBinding>
3331 (P003)     <wsp:Policy>
3332 (P004)       <sp:InitiatorToken>
3333 (P005)         <wsp:Policy>
3334 (P006)           <sp:X509Token sp:IncludeToken="http://docs.oasis-
3335 open.org/ws-sx/ws-
3336 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3337 (P007)             <wsp:Policy>
3338 (P008)               <sp:WssX509V3Token10/>
3339 (P009)             </wsp:Policy>
3340 (P010)           </sp:X509Token>
3341 (P011)         </wsp:Policy>
3342 (P012)       </sp:InitiatorToken>
3343 (P013)     <sp:RecipientToken>
3344 (P014)       <wsp:Policy>
3345 (P015)         <sp:X509Token sp:IncludeToken="http://docs.oasis-
3346 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
3347 (P016)           <wsp:Policy>
3348 (P017)             <sp:WssX509V3Token10/>
3349 (P018)           </wsp:Policy>
3350 (P019)         </sp:X509Token>
3351 (P020)       </wsp:Policy>
3352 (P021)     </sp:RecipientToken>
3353 (P022)   <sp:AlgorithmSuite>
3354 (P023)     <wsp:Policy>
3355 (P024)       <sp:Basic256/>
3356 (P025)     </wsp:Policy>
3357 (P026)   </sp:AlgorithmSuite>
3358 (P027) <sp:Layout>
3359 (P028)   <wsp:Policy>
3360 (P029)     <sp:Strict/>
3361 (P030)   </wsp:Policy>
3362 (P031) </sp:Layout>
3363 (P032) <sp:IncludeTimestamp/>
3364 (P033) <sp:OnlySignEntireHeadersAndBody/>
3365 (P034) </wsp:Policy>
3366 (P035) </sp:AsymmetricBinding>
3367 (P036) <sp:Wss11>
3368 (P037)   <wsp:Policy>
3369 (P038)     <sp:MustSupportRefKeyIdentifier/>
3370 (P039)     <sp:MustSupportRefIssuerSerial/>
3371 (P040)     <sp:MustSupportRefThumbprint/>
3372 (P041)     <sp:MustSupportRefEncryptedKey/>
3373 (P042)   </wsp:Policy>
3374 (P043) </sp:Wss11>
3375 (P044) <sp:SupportingTokens>
3376 (P045)   <wsp:Policy>
```

```

3377 (P046) <sp:SamIToken sp:IncludeToken="http://docs.oasis-open.org/ws-
3378 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3379 (P047) <wsp:Policy>
3380 (P048) <sp:WssSamIv20Token11/>
3381 (P049) </wsp:Policy>
3382 (P050) </sp:SamIToken>
3383 (P051) </wsp:Policy>
3384 (P052) </sp:SupportingTokens>
3385 (P053) </wsp:Policy>

```

3386 Lines (P001)-(P045) contain the endpoint policy, which contains 3 assertions: an sp:AsymmetricBinding  
3387 assertion, an sp:Wss11 assertion, and an sp:SupportingTokens assertion.

3388 Lines (P002)-(P035) contain the sp:AsymmetricBinding assertion, which is identical to the same assertion  
3389 that is described in section 2.1.3. Please refer to section 2.1.3 for details.

3390 Lines (P036)-(P043) contain an sp:Wss11 assertion, which contains a set of assertions that specify the  
3391 token referencing techniques that are required to be supported (P038)-(P041).

3392 Lines (P044)-(P052) contain an sp:SupportingTokens assertion, which contains an sp:SamIToken  
3393 (P046)-(P050), which specifies that it must always be sent to the Recipient. The sp:SamIToken contains  
3394 an sp:WssSamIv20Token11 assertion (P048), which means that the SamIToken provided must be a  
3395 SAML Version 2.0 Assertion that is submitted in compliance with the [\[WSS11-SAML1120-PROFILE\]](#).

3396 The fact that the sp:SamIToken is contained in an sp:SupportingTokens element indicates to the Initiator  
3397 that the SAML Assertion must use the saml2:SubjectConfirmation@Method "bearer" as described above  
3398 in introductory section 2.3.

3399 The following is an example request compliant with the above policy:

```

3400
3401 (M001) <?xml version="1.0" encoding="UTF-8"?>
3402 (M002) <s:Envelope xmlns:s=".../soap/envelope/" xmlns:sec="..."
3403 (M003) xmlns:wssse="..." xmlns:wsu="..." xmlns:wsa="...addressing"
3404 (M004) xmlns:sb="...liberty:sb" xmlns:pp="...liberty.id-sis-pp"
3405 (M005) xmlns:ds="...xmldsig#" xmlns:xenc="...xmlenc#">
3406 (M006) <s:Header>
3407 (M007) <!-- see Liberty SOAP Binding Spec for reqd, optional hdrs -->
3408 (M008) <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>
3409 (M009) <wsa:To wsu:Id="to">...</wsa:To>
3410 (M010) <wsa:Action wsu:Id="action">...</wsa:Action>
3411 (M011) <wsse:Security mustUnderstand="1">
3412 (M012) <wsu:Timestamp wsu:Id="ts">
3413 (M013) <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
3414 (M014) </wsu:Timestamp>
3415 (M015) <!-- this is the bearer token -->
3416 (M016) <saml2:Assertion xmlns:saml2="...SAML:2.0:assertion"
3417 (M017) Version="2.0" ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
3418 (M018) IssueInstant="2005-04-01T16:58:33.173Z">
3419 (M019) <saml2:Issuer>http://authority.example.com/</saml2:Issuer>
3420 (M020) <!-- signature by the issuer over the assertion -->
3421 (M021) <ds:Signature>...</ds:Signature>
3422 (M022) <saml2:Subject>
3423 (M023) <saml2:EncryptedID>
3424 (M024) <xenc:EncryptedData
3425 (M025) >U2XTCNvRX7B1lNK182nmY00TEk==</xenc:EncryptedData>
3426 (M026) <xenc:EncryptedKey>...</xenc:EncryptedKey>
3427 (M027) </saml2:EncryptedID>
3428 (M028) <saml2:SubjectConfirmation
3429 (M029) Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3430 (M030) </saml2:Subject>
3431 (M031) <!-- audience restriction on assertion can limit scope of
3432 (M032) which entity should consume the info in the assertion. -->
3433 (M033) <saml2:Conditions NotBefore="2005-04-01T16:57:20Z"
3434 (M034) NotOnOrAfter="2005-04-01T21:42:4 3Z">
3435 (M035) <saml2:AudienceRestrictionCondition>

```

```

3436 (M036) <saml2:Audience>http://wsp.example.com</saml2:Audience>
3437 (M037) </saml2:AudienceRestrictionCondition>
3438 (M038) </saml2:Conditions>
3439 (M039) <!-- The AuthnStatement carries info that describes authN
3440 (M040) event of the Subject to an Authentication Authority -->
3441 (M041) <saml2:AuthnStatement AuthnInstant="2005-04-01T16:57:30.000Z"
3442 (M042) SessionIndex="6345789">
3443 (M043) <saml2:AuthnContext>
3444 (M044) <saml2:AuthnContextClassRef
3445 (M045) >...:SAML:2.0:ac:classes:PasswordProtectedTransport
3446 (M046) </saml2:AuthnContextClassRef>
3447 (M047) </saml2:AuthnContext>
3448 (M048) </saml2:AuthnStatement>
3449 (M049) <!-- This AttributeStatement carries an EncryptedAttribute.
3450 (M050) Once this element decrypted with supplied key an <Attribute>
3451 (M051) element bearing endpoint reference can be found, specifying
3452 (M052) resources which invoker may access. Details on element can be
3453 (M053) found in discovery service specification. -->
3454 (M054) <saml2:AttributeStatement>
3455 (M055) <saml2:EncryptedAttribute>
3456 (M056) <xenc:EncryptedData
3457 (M057) Type="http://www.w3.org/2001/04/xmlenc#Element" >
3458 (M058) mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8
3459 (M059) ... hg6nZ5c0I6L6Gn9A=HCQY
3460 (M060) </xenc:EncryptedData>
3461 (M061) <xenc:EncryptedKey> ... </xenc:EncryptedKey>
3462 (M062) </saml2:EncryptedAttribute>
3463 (M063) </saml2:AttributeStatement>
3464 (M064) </saml2:Assertion>
3465 (M065) <!-- This SecurityTokenReference is used to reference the SAML
3466 (M066) Assertion from a ds:Reference -->
3467 (M067) <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsp="..."
3468 (M068) xmlns:wss1="..." wsp:Id="str1" wss1:TokenType=
3469 (M069) ".../wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
3470 (M070) <wsse:KeyIdentifier ValueType=
3471 (M071) ".../wss/oasis-wss-saml-token-profile-1.1#SAMLID"
3472 (M072) >sxJu9g/vvLG9sAN9bKp/8q0NKU=</wsse:KeyIdentifier>
3473 (M073) </wsse:SecurityTokenReference>
3474 (M074) <ds:Signature>
3475 (M075) <ds:SignedInfo>
3476 (M076) <!-- in general include a ds:Reference for each wsa:header
3477 (M077) added according to SOAP binding, plus timestamp, plus ref to
3478 (M078) assertion to avoid token substitution attacks, plus Body -->
3479 (M079) <ds:Reference URI="#to">...</ds:Reference>
3480 (M080) <ds:Reference URI="#action">...</ds:Reference>
3481 (M081) <ds:Reference URI="#mid">...</ds:Reference>
3482 (M082) <ds:Reference URI="#ts">...</ds:Reference>
3483 (M083) <ds:Reference URI="#Str1">
3484 (M084) <ds:Transform Algorithm="...#STR-Transform">
3485 (M085) <wsse:TransformationParameters>
3486 (M086) <ds:CanonicalizationMethod Algorithm=
3487 (M087) "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
3488 (M088) </wsse:TransformationParameters>
3489 (M089) </ds:Transform>
3490 (M090) </ds:Reference>
3491 (M091) <ds:Reference URI="#MsgBody">...</ds:Reference>
3492 (M092) </ds:SignedInfo>
3493 (M093) ...
3494 (M094) </ds:Signature>
3495 (M095) </wsse:Security>
3496 (M096) </s:Header>
3497 (M097) <s:Body wsp:Id="MsgBody">
3498 (M098) <pp:Modify>
3499 (M099) <!-- this is an ID-SIS-PP Modify message -->

```

```
3500 (M0100) </pp:Modify>
3501 (M0101) </s:Body>
3502 (M0102) </s:Envelope>
```

3503

3504 The sample message above was taken and cosmetically edited from the [[WSS11-LIBERTY-SAML20-](#)  
3505 [PROFILE](#)].

3506 Lines (M002)-(M0102) contain the SOAP:Envelope, which contains the SOAP:Header (M006)-(M096)  
3507 and the SOAP:Body (M097)-(M0101).

3508 The SOAP Header contains some WS-Addressing (wsa:) parameters (M008)-(M010) (not discussed  
3509 here) and a wsse:Security header.

3510 The wsse:Security header (M011)-(M095) contains a wsu:Timestamp (M012)-(M014), a saml2:Assertion  
3511 (M016)-(M064), a wsse:SecurityTokenReference (M067)-(M073), and a ds:Signature (M074)-(M094).

3512 The wsu:Timestamp (M012)-(M014) is required by the policy sp:IncludeTimestamp assertion (P032).

3513 The saml2:Assertion (M016)-(M064) is required by the policy sp:WssSamlV20Token11 (P048). The  
3514 saml2:Assertion is Version 2.0 (M017) and it is signed by the IssuingAuthority (M021). This is the  
3515 ds:Signature (M021) of the IssuingAuthority, identified in the saml2:Issuer element (M019), that the  
3516 Recipient trusts with respect to recognizing the Requestor and determining whether the request should be  
3517 granted. In addition, this saml2:Assertion uses the “bearer” saml2:SubjectConfirmation@Method, as  
3518 required by the policy sp:SamlToken in the sp:SupportingTokens element described above  
3519 (P044)-(P052).

3520 Note: the saml2:Assertion contains a saml2:EncryptedID (M023)-(M027), which contains the  
3521 identity of the Requestor and a saml2:EncryptedAttribute (M062), which contains information  
3522 about the Requestor. It is presumed that prior to issuing this request, that the Requestor  
3523 contacted the IssuingAuthority (directly or indirectly) and was granted permission to access the  
3524 web service that is now the target of this request. As such, the IssuingAuthority has knowledge of  
3525 this web service and presumably the public certificate associated with that service and has used  
3526 the public key contained in that certificate to encrypt the aforementioned portions of the  
3527 saml2:Assertion, which can only be decrypted by the RelyingParty who presumably has entrusted  
3528 the private key of the service with the Recipient, in order that the Recipient may decrypt the  
3529 necessary data.

3530 However, before the Recipient can evaluate these aspects of the request, the requirements of the policy  
3531 sp:AsymmetricBinding (P002)-(P035) must be met in terms of proper “presentation” of the request as  
3532 described below.

3533 Lines (M074)-(M094) contain the message signature ds:Signature element, which contains a  
3534 ds:SignedInfo that contains ds:Reference elements that cover the wsa: headers (M079)-(M081), the  
3535 wsu:Timestamp (M082) (required by the policy sp:IncludeTimestamp assertion (P032), the  
3536 saml2:Assertion (M083)-(M090), and the SOAP:Body (M091).

3537 The ds:Reference (M083)-(M090) covering the saml2:Assertion warrants further examination.

3538 The first point to note is that to reference the saml2:Assertion the ds:Reference uses an STR-  
3539 Transform (M084) to reference a wsse:SecurityTokenReference (M067)-(M073), which is in  
3540 compliance with policy sp:Wss11 sp:MustSupportRefKeyIdentifier assertion (P038).

3541 The second point to note about this ds:Reference is that is covering the saml2:Assertion even  
3542 though the policy references the sp:SamlToken in an sp:SupportingTokens element and not an  
3543 sp:SignedSupportingTokens element. The reason this is the case is that it is the fact that an  
3544 sp:SupportingTokens (P044)-(P052) element is used that tells the Initiator that a SAML bearer  
3545 Assertion is required. However, this only means that the SamlToken is not “required” to be  
3546 signed. It is the Initiator’s choice whether to sign it or not, and it is generally good practice to do  
3547 so in order to prevent a token substitution attack.

3548

3549

### 3550 2.3.2.2 (WSS1.1) SAML2.0 Sender Vouches over SSL

3551 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP Scenario](#)  
3552 #2].

3553 Similar to 2.3.1.2 except SAML token is of version 2.0.

3554

3555

```
3556 (P001) <wsp:Policy>  
3557 (P002)   <sp:TransportBinding>  
3558 (P003)   <wsp:Policy>  
3559 (P004)   <sp:TransportToken>  
3560 (P005)   <wsp:Policy>  
3561 (P006)   <sp:HttpsToken>  
3562 (P007)   <wsp:Policy>  
3563 (P008)   <sp:RequireClientCertificate>  
3564 (P009)   </wsp:Policy>  
3565 (P010)   </sp:HttpsToken>  
3566 (P011)   </wsp:Policy>  
3567 (P012)   </sp:TransportToken>  
3568 (P013)   <sp:AlgorithmSuite>  
3569 (P014)   <wsp:Policy>  
3570 (P015)   <sp:Basic256 />  
3571 (P016)   </wsp:Policy>  
3572 (P017)   </sp:AlgorithmSuite>  
3573 (P018)   <sp:Layout>  
3574 (P019)   <wsp:Policy>  
3575 (P020)   <sp:Strict />  
3576 (P021)   </wsp:Policy>  
3577 (P022)   </sp:Layout>  
3578 (P023)   <sp:IncludeTimestamp />  
3579 (P024)   </wsp:Policy>  
3580 (P025)   </sp:TransportBinding>  
3581 (P026)   <sp:SignedSupportingTokens>  
3582 (P027)   <wsp:Policy>  
3583 (P028)   <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
3584          sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3585 (P029)   <wsp:Policy>  
3586 (P030)   <sp:WssSamlV20Token11/>  
3587 (P031)   </wsp:Policy>  
3588 (P032)   </sp:SamlToken>  
3589 (P033)   </wsp:Policy>  
3590 (P034)   </sp:SignedSupportingTokens>  
3591 (P035)   </wsp:Policy>
```

3592 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by  
3593 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and  
3594 sp:SignedSupportingTokens (P026).

3595 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
3596 by a secure transport protocol such as SSL or TLS.

3597 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
3598 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
3599 transport token using the Basic256 algorithm suite (P015).

3600 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of  
3601 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section  
3602 4.3.1].

3603 The requirements for the sp:Layout assertion (P018)-(P022) should be automatically met (or may be  
3604 considered moot) since there are no cryptographic tokens required to be present in the WS-Security  
3605 header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout  
3606 would need to be considered.)

3607 A timestamp (P023) is required to be included in an acceptable message.

3608

3609 Here is an example request:

3610

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

3648 Lines (M002)-(M037) contain the SOAP:Envelope, which contains the SOAP:Header (M003)-(M031) and  
3649 the SOAP:Body (M032)-(M036).

3650 Lines (M009)-(M030) contain the wsse:Security header, which, in conjunction with the client  
3651 certificate (P008), is the primary basis for trust in this example.

3652 Lines (M010)-(M012) contain the wsu:Timestamp, which meets the sp:IncludeTimestamp assertion policy  
3653 requirement (P023).

3654 Lines (M013)-(M029) contain the saml2:Assertion, which is required to be Version 2.0 (M014) and to be  
3655 used within the [WSS11-SAML1120-PROFILE], because of the policy sp:SamIToken assertion  
3656 (P028)-(P032), which contains the sp:WssSamIv20Token11 (P030).

3657 Lines (M022)-(M023) indicate that the SAML Assertion uses the saml2:Method sender-vouches for the  
3658 purposes of saml2:SubjectConfirmation, which is required by the fact that the sp:SamIToken appears in  
3659 an sp:SignedSupportingTokens assertion (P026)-(P034) in conjunction with the client certificate required  
3660 by the sp:RequireClientCertificate assertion (P008) within the sp:TransportBinding as described in section  
3661 2.3 above. In addition, the Recipient should be able to correlate the saml2:Issuer (M015) as being  
3662 properly associated with the client certificate received from the HTTPS (SSL) connection.

3663

3664

### 3665 2.3.2.3 (WSS1.1) SAML2.0 HoK over SSL

3666 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)  
3667 Scenario #5].

3668 Similar to 2.3.1.3 except SAML token is of version 2.0.

3669 Initiator adds a SAML Assertion (hk) to the SOAP Security Header. In this policy the sp:TransportBinding  
3670 requires a Client Certificate AND the sp:Sam1Token is in an sp:SignedEndorsingSupportingTokens  
3671 element. A SAML holder-of-key Assertion meets these requirements because it is “virtually signed” by the  
3672 message signature as a result of the SSL client certificate authentication procedure as described in  
3673 section 2.3. Furthermore, the SAML hk Assertion in this case is a “virtually endorsing”, because the key  
3674 identified in the holder-of-key saml2:SubjectConfirmationData is also the client certificate, which is  
3675 virtually endorsing its own signature, under the authority of the IssuingAuthority who has signed the  
3676 SAML hk Assertion.

3677 As a result, the Initiator may be considered to be authorized by the saml2:Issuer of the hk SAML  
3678 Assertion to bind message content to the Subject of the Assertion. If the Client Certificate matches the  
3679 certificate identified in the hk Assertion, the Initiator may be regarded as executing SAML hk responsibility  
3680 of binding the Requestor, who would be the Subject of the hk Assertion, to the content of the message.

3681 (Note: the same considerations described in section 2.3.1.4 with respect to whether the Subject  
3682 of the Assertion and the Subject of the Client Certificate are the same entity determining whether  
3683 the Client Certificate is attesting for the Assertion Subject or whether the Client Certificate is  
3684 authenticating as the Assertion Subject apply here.)

3685 In this scenario, the IssuingAuthority is the saml2:Issuer(signer) of the hk SAML Assertion. The Requestor  
3686 is the Subject of the Assertion and the Initiator is authorized by the IssuingAuthority to bind the Assertion  
3687 to the message using the ClientCertificate identified in the SAML Assertion, which may also be  
3688 considered to be virtually signing the wsu:Timestamp of the message. Optionally, a separate Signature  
3689 may be used to sign the wsu:Timestamp, which the Recipient would also be required to verify was signed  
3690 by the client certificate in this example.

3691

```
3692 (P001) <wsp:Policy>  
3693 (P002)   <sp:TransportBinding>  
3694 (P003)   <wsp:Policy>  
3695 (P004)     <sp:TransportToken>  
3696 (P005)       <wsp:Policy>  
3697 (P006)         <sp:HttpsToken>  
3698 (P007)           <wsp:Policy>  
3699 (P008)             <sp:RequireClientCertificate>  
3700 (P009)               </wsp:Policy>  
3701 (P010)                 </sp:HttpsToken>  
3702 (P011)                   </wsp:Policy>  
3703 (P012)                     </sp:TransportToken>  
3704 (P013)                       <sp:AlgorithmSuite>  
3705 (P014)                         <wsp:Policy>  
3706 (P015)                           <sp:Basic256 />  
3707 (P016)                             </wsp:Policy>  
3708 (P017)                               </sp:AlgorithmSuite>  
3709 (P018)                                 <sp:Layout>  
3710 (P019)                                   <wsp:Policy>  
3711 (P020)                                     <sp:Strict />  
3712 (P021)                                       </wsp:Policy>  
3713 (P022)                                         </sp:Layout>  
3714 (P023)                                           <sp:IncludeTimestamp />  
3715 (P024)                                             </wsp:Policy>  
3716 (P025)                                               </sp:TransportBinding>  
3717 (P026)                                                 <sp:SignedEndorsingSupportingTokens>  
3718 (P027)                                                   <wsp:Policy>  
3719 (P028)                                                     <sp:Sam1Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
3720 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3721 (P029)                                                       </wsp:Policy>
```

```

3722 (P030)         <sp:WssSamlV20Token11/>
3723 (P031)         </wsp:Policy>
3724 (P032)         </sp:SamlToken>
3725 (P033)         </wsp:Policy>
3726 (P034)         </sp:SignedEndorsingSupportingTokens>
3727 (P035)         </wsp:Policy>

```

3728

3729 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by  
3730 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and  
3731 sp:EndorsingSupportingTokens (P026).

3732 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
3733 by a secure transport protocol such as SSL or TLS.

3734 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
3735 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
3736 transport token using the Basic256 algorithm suite (P015).

3737 In addition, because this is SAML holder-of-key, a client certificate is required (P008) as the basis of trust  
3738 for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section 4.3.1]. In  
3739 the holder-of-key case, there will be an additional certificate required in the trust chain, which is the  
3740 certificate used to sign the SAML hk Assertion, which will be contained or referenced in the Assertion.

3741 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)  
3742 since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a  
3743 signature element was included to cover the wsse:Timestamp, then the layout would need to be  
3744 considered.)

3745 A timestamp (P023) is required to be included in an acceptable message.

3746 Lines (P026)-(P034) contain an sp:SignedEndorsingSupportingTokens element, which means that the  
3747 contained supporting token (the SAML hk Assertion) references a key that will be “signing” (endorsing)  
3748 the message signature. The token itself may also be considered to be “signed”, because it is contained in  
3749 the message sent over the SSL link. In the case of sp:TransportBinding, there may be no actual  
3750 “message signature”, however, when a client certificate is used, the service can be assured that the  
3751 connection was set up by the client and that the SSL link guarantees the integrity of the data that is sent  
3752 on the link by the client, while it is on the link and when it is received from the link by using the key  
3753 referenced by the token. However, it does not guarantee the integrity after the data is received (i.e. after it  
3754 is received there is no way to tell whether any changes have been made to it since it has been received).  
3755 In any event, within this context a Signed Endorsing Supporting Token can be used to tell the Recipient  
3756 that the Issuer of the token is making claims related to the holder of the private key that is referenced by  
3757 the token, which in this case would be the private key associated with the client certificate used to set up  
3758 the SSL link, as described further below.

3759 Lines (P028)-(P032) contain an sp:SamlToken, which must always be sent to the Recipient.

3760 Line (P030) specifies that the token is of type WssSamlV20Token11, which means that the Endorsing  
3761 Supporting Token must be a SAML Version 2.0 token and it must be sent using WS-Security 1.1 using  
3762 the [[WSS11-SAML1120-PROFILE](#)]. Note that because the SamlToken is contained in an  
3763 sp:EndorsingSupportingTokens element, that it implicitly must be a holder-of-key token as described in  
3764 section 2.3 above.

3765 Here is an example request taken from the WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)  
3766 Scenario #5] containing only minor cosmetic modifications and corrections.

3767

```

3768 (M001) <?xml version="1.0" encoding="utf-8" ?>
3769 (M002) <S11:Envelope
3770 (M003)     xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3771 (M004)     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3772 (M005)     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3773 (M006)     xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3774 (M007)     xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3775 (M008)     xmlns:wssell=".../oasis-wss-wssecurity-secext-1.1.xsd"

```

```

3776 (M009)      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3777 (M010)      xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3778 (M011)      <S11:Header>
3779 (M012)      <wsse:Security S11:mustUnderstand="1">
3780 (M013)      <wsu:Timestamp>
3781 (M014)      <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3782 (M015)      </wsu:Timestamp>
3783 (M016)      <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3784 (M017)      IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3785 (M018)      <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3786 (M019)      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3787 (M020)      <ds:SignedInfo>
3788 (M021)      <ds:CanonicalizationMethod
3789 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3790 (M023)      <ds:SignatureMethod
3791 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3792 (M025)      <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3793 (M026)      <ds:Transforms>
3794 (M027)      <ds:Transform Algorithm=
3795 (M028)      "http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
3796 (M029)      <ds:Transform Algorithm=
3797 (M030)      "http://www.w3.org/2001/10/xml-exc-c14n#">
3798 (M031)      <InclusiveNamespaces
3799 (M032)      PrefixList="#default saml ds xs xsi"
3800 (M033)      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3801 (M034)      </ds:Transform>
3802 (M035)      </ds:Transforms>
3803 (M036)      <ds:DigestMethod Algorithm=
3804 (M037)      "http://www.w3.org/2000/09/xmldsig#sha1" />
3805 (M038)      <ds:DigestValue
3806 (M039)      >Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3807 (M040)      </ds:Reference>
3808 (M041)      </ds:SignedInfo>
3809 (M042)      <ds:SignatureValue>
3810 (M043)      hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgzpkJN9CMLG8ENR4Nrw+n
3811 (M044)      7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmtpt3TD
3812 (M045)      MwuL/cBUj2OtBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3813 (M046)      </ds:SignatureValue>
3814 (M047)      <ds:KeyInfo>
3815 (M048)      <ds:X509Data>
3816 (M049)      <ds:X509Certificate>
3817 (M050)      MIICyjcCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxkCzAJBgNVBAYTA1VT
3818 (M051)      ...
3819 (M052)      8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREG8cCx3w/w==
3820 (M053)      </ds:X509Certificate>
3821 (M054)      </ds:X509Data>
3822 (M055)      </ds:KeyInfo>
3823 (M056)      </ds:Signature>
3824 (M057)      <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3825 (M058)      NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3826 (M059)      <saml2:Subject>
3827 (M060)      <saml2:NameID
3828 (M061)      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3829 (M062)      >uid=joe,ou=people,ou=saml-demo,o=example.com</saml2:NameID>
3830 (M063)      <saml2:SubjectConfirmation
3831 (M064)      Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" />
3832 (M065)      <saml2:SubjectConfirmationData
3833 (M066)      xsi:type="saml2:KeyInfoConfirmationDataType">
3834 (M067)      <ds:KeyInfo>
3835 (M068)      <ds:X509Data>
3836 (M069)      <ds:X509IssuerName>
3837 (M070)      C=ZA, ST=Western Cape, L=Cape Town,
3838 (M071)      O=Thawte Consulting cc,
3839 (M072)      OU=Certification Services Division,

```

```

3840 (M073)          CN=Thawte Server CA/Email=server-certs@thawte.com
3841 (M074)          </ds:X509IssuerName>
3842 (M075)          <X509SerialNumber>12345678</X509SerialNumber>
3843 (M076)          </ds:X509Data>
3844 (M077)          </ds:KeyInfo>
3845 (M078)          </saml2:SubjectConfirmationData>
3846 (M079)          </saml2:Subject>
3847 (M080)          <saml2:AttributeStatement>
3848 (M081)          <saml2:Attribute Name="MemberLevel">
3849 (M082)          <saml2:AttributeValue>gold</saml2:AttributeValue>
3850 (M083)          </saml2:Attribute>
3851 (M084)          <saml2:Attribute Name="E-mail">
3852 (M085)          <saml2:AttributeValue
3853 (M086)          >joe@yahoo.com</saml2:AttributeValue>
3854 (M087)          </saml2:Attribute>
3855 (M088)          </saml2:AttributeStatement>
3856 (M089)          </saml2:Assertion>
3857 (M090)          </wsse:Security>
3858 (M091)          </S11:Header>
3859 (M092)          <S11:Body wsu:Id="MsgBody">
3860 (M093)          <ReportRequest>
3861 (M094)          <TickerSymbol>SUNW</TickerSymbol>
3862 (M095)          </ReportRequest>
3863 (M096)          </S11:Body>
3864 (M097)          </S11:Envelope>

```

3865 Lines (M002)-(M097) contain the SOAP:Envelope, which contains the SOAP:Header (M011)-(M091) and  
3866 the SOAP:Body (M092)-(M096).

3867 Lines (M012)-(M090) contain the wsse:Security header, which contains a wsu:Timestamp (M013)-(M015)  
3868 and a saml2:Assertion (M016)-(M089).

3869 The wsu:Timestamp (M013)-(M015) may be considered to be virtually signed by the client certificate as  
3870 explained above and in section 2.3.

3871 The saml2:Assertion was issued by an IssuingAuthority identified in the saml2:Issuer element (M018).

3872 The saml2:Issuer has used the private key of its enterprise certificate to produce the ds:Signature  
3873 element (M019)-(M056) that is an enveloped-signature (M028) that covers its parent XML element, the  
3874 saml2:Assertion (M016)-(M089). The ds:KeyInfo (M047)-(M055) within this ds:Signature contains an  
3875 actual copy of the Issuer's enterprise certificate, that the Recipient can verify for authenticity to establish  
3876 that this message is in conformance with any agreement the RelyingParty may have with the  
3877 IssuingAuthority. The details of this verification are outside the scope of this document, however they  
3878 involve the structures and systems the RelyingParty has in place recognize and verify certificates and  
3879 signatures it receives that are associated with business arrangements with the holders of the private keys  
3880 of those certificates.

3881 The saml2:Subject of the saml2:Assertion is contained in lines (M059)-(M079). Within the saml2:Subject  
3882 is the saml2:NameID (M060)-(M062), which contains the official name of the Subject of this Assertion and  
3883 this entity may be considered to the Requestor associated with this request.

3884 The saml2:SubjectConfirmation (M063)-(M064) has Method "holder-of-key" which implies that there is a  
3885 saml2:SubjectConfirmationData element (M065)-(M078) which will contain information that identifies a  
3886 signing key that the Initiator will use to bind this saml2:Assertion to a message that is associated with the  
3887 Requestor. In this context, the Initiator is acting as "attesting entity" with respect to the Subject as defined  
3888 in [SAML20], which means that the Initiator is authorized by the IssuingAuthority to present  
3889 saml2:Assertions pertaining to saml2:Subjects to Recipients/RelyingParties. In this example there is a  
3890 ds:KeyInfo (M067)-(M077) that identifies a specific certificate (ds:X509IssuerName (M069)-(M074) and  
3891 ds:SerialNumber (M075)) that the Initiator must prove possession of the associated private key to verify  
3892 this message. In this policy that private key is the one associated with the client certificate that the Initiator  
3893 is required to use (P008).

3894 The saml2:AttributeStatement (M080)-(M088) contains some information about the Subject that is  
3895 officially being provided in this saml2:Assertion by the IssuingAuthority that may have some significance  
3896 to the Relying Party in terms of determining whether access is granted for this request.

3897

3898 **2.3.2.4 (WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign,**  
3899 **Encrypt**

3900 Here the message and SAML Assertion are signed using a key derived from the ephemeral key K. The  
3901 ephemeral key is encrypted using the Recipient's public key for the request input message and the same  
3902 shared ephemeral key, K, is encrypted using the Initiators public key for the response output message.

3903 Alternatively, derived keys can be used for each of signing and encryption operations.

3904 In this scenario the Authority is the Initiator who signs the message with the generated key. In order to  
3905 establish trust in the generated key, the Initiator must sign the message signature with a second signature  
3906 using an X509 certificate, which is indicated as the EndorsingSupportingToken. This X509 certificate  
3907 establishes the Initiator as the SAML Authority.

3908 (Note: there are instructive similarities and differences between this example and example 2.3.1.4, where  
3909 the difference is that: here the binding is symmetric and the WSS1.1 is used, whereas in 2.3.1.4 the  
3910 binding is asymmetric and WSS1.0 is used. One particular item is that in 2.3.1.4 an  
3911 EndorsingSupportingToken is not needed because in 2.3.1.4 the asymmetric binding uses X.509  
3912 certificates, which may be inherently trusted as opposed to the ephemeral key, K, used here.)

3913

```
3914 (P001) <wsp:Policy wsu:Id="WSS11SamlWithCertificates_policy">
3915 (P002)   <wsp:ExactlyOne>
3916 (P003)   <wsp>All>
3917 (P004)     <sp:SymmetricBinding>
3918 (P005)       <wsp:Policy>
3919 (P006)         <sp:ProtectionToken>
3920 (P007)           <wsp:Policy>
3921 (P008)             <sp:X509Token sp:IncludeToken=
3922 "http://docs.oasis-open.org/ws-sx/ws-
3923 securitypolicy/200702/IncludeToken/Never">
3924 (P009)               <wsp:Policy>
3925 (P010)                 <sp:RequireThumbprintReference/>
3926 (P011)                 <sp:RequireDerivedKeys wsp:Optional="true"/>
3927 (P012)                 <sp:WssX509V3Token10/>
3928 (P013)               </wsp:Policy>
3929 (P014)             </sp:X509Token>
3930 (P015)           </wsp:Policy>
3931 (P016)         </sp:ProtectionToken>
3932 (P017)       <sp:AlgorithmSuite>
3933 (P018)         <wsp:Policy>
3934 (P019)           <sp:Basic256/>
3935 (P020)         </wsp:Policy>
3936 (P021)       </sp:AlgorithmSuite>
3937 (P022)     <sp:Layout>
3938 (P023)       <wsp:Policy>
3939 (P024)         <sp:Strict/>
3940 (P025)       </wsp:Policy>
3941 (P026)     </sp:Layout>
3942 (P027)   <sp:IncludeTimestamp/>
3943 (P028) <sp:OnlySignEntireHeadersAndBody/>
3944 (P029) </wsp:Policy>
3945 (P030) </sp:SymmetricBinding>
3946 (P031) <sp:SignedSupportingTokens>
3947 (P032)   <wsp:Policy>
3948 (P033)     <sp:SamlToken sp:IncludeToken=
3949 "http://docs.oasis-open.org/ws-sx/ws-
3950 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3951 (P034)       <wsp:Policy>
3952 (P035)         <sp:WssSamlV11Token11/>
3953 (P036)       </wsp:Policy>
3954 (P037)     </sp:SamlToken>
3955 (P038)   </wsp:Policy>
3956 (P039) </sp:SignedSupportingTokens>
```

```

3957 (P040) <sp:EndorsingSupportingTokens>
3958 (P041) <wsp:Policy>
3959 (P042) <sp:X509Token sp:IncludeToken="AlwaysToRecipient">
3960 (P043) <wsp:Policy>
3961 (P044) <sp:WssX509V3Token11/>
3962 (P045) </wsp:Policy>
3963 (P046) </sp:X509Token>
3964 (P047) </wsp:Policy>
3965 (P048) </sp:EndorsingSupportingTokens>
3966 (P049) <sp:Wss11>
3967 (P050) <wsp:Policy>
3968 (P051) <sp:MustSupportRefKeyIdentifier/>
3969 (P052) <sp:MustSupportRefIssuerSerial/>
3970 (P053) <sp:MustSupportRefThumbprint/>
3971 (P054) <sp:MustSupportRefEncryptedKey/>
3972 (P055) </wsp:Policy>
3973 (P056) </sp:Wss11>
3974 (P057) </wsp:All>
3975 (P058) </wsp:ExactlyOne>
3976 (P059) </wsp:Policy>
3977 (P060)
3978 (P061) <wsp:Policy wsu:Id="SamlForCertificates_input_policy">
3979 (P062) <wsp:ExactlyOne>
3980 (P063) <wsp:All>
3981 (P064) <sp:SignedParts>
3982 (P065) <sp:Body/>
3983 (P066) </sp:SignedParts>
3984 (P067) <sp:EncryptedParts>
3985 (P068) <sp:Body/>
3986 (P069) </sp:EncryptedParts>
3987 (P070) </wsp:All>
3988 (P071) </wsp:ExactlyOne>
3989 (P072) </wsp:Policy>
3990 (P073)
3991 (P074) <wsp:Policy wsu:Id="SamlForCertificate_output_policy">
3992 (P075) <wsp:ExactlyOne>
3993 (P076) <wsp:All>
3994 (P077) <sp:SignedParts>
3995 (P078) <sp:Body/>
3996 (P079) </sp:SignedParts>
3997 (P080) <sp:EncryptedParts>
3998 (P081) <sp:Body/>
3999 (P082) </sp:EncryptedParts>
4000 (P083) </wsp:All>
4001 (P084) </wsp:ExactlyOne>
4002 (P085) </wsp:Policy>

```

4003 Lines (P001)-(P059) contain the policy that requires all the contained assertions to be complied with by  
4004 the Initiator. In this case there are 4 assertions: sp:SymmetricBinding (P004)-(P030),  
4005 sp:SignedSupportingTokens (P031)-(P039), sp:EndorsingSupportingTokens (P040)-(P048), and  
4006 sp:Wss11 (P049)-(P056).

4007 The sp:SymmetricBinding assertion (P004) contains an sp:ProtectionToken (P006)-(P016), which  
4008 indicates that both the Initiator and Recipient must use each other's public key respectively associated  
4009 with the X509Token (P008)-(P014) associated with the respective message receiver to encrypt the  
4010 shared ephemeral symmetric key as explained at the beginning of this section. The messages may either  
4011 be encrypted and signed using keys derived from the ephemeral key as indicated by the  
4012 sp:RequireDerivedKeys assertion (P011) or the encryption and signing can be done using the ephemeral  
4013 key itself without deriving keys, because the RequireDerivedKey assertion is an optional requirement as  
4014 indicated by the wsp:Optional attribute on line (P011).

4015 The sp:SignedSupportingTokens assertion (P031) contains an sp:SamIToken assertion (P033)-(P037),  
4016 which indicates that a signed SAML Assertion must always be included in the Initiator's request to the  
4017 Recipient (AlwaysToRecipient (P033)). This SAML Assertion must be included in the WS-Security header

4018 and referenced and signed as described in the WS-Security 1.1 Profile for SAML [\[WSS11-SAML1120-](#)  
4019 [PROFILE\]](#) as indicated by the sp:WssSamlV11Token11 assertion (P036), which indicates the SAML 1.1  
4020 option in that profile (as opposed to the SAML 2.0 option, which would have been indicated by  
4021 sp:WssSamlV20Token11).

4022 The sp:EndorsingSupportingToken assertion (P040) is needed because there are no guarantees that the  
4023 ephemeral key, K, is still being used by the Initiator with whom the Recipient would have collaborated  
4024 originally to establish the ephemeral key as a shared secret. The purpose of the endorsing token is to  
4025 sign the signature made by the ephemeral key, using the Initiator's private key, which will explicitly  
4026 guarantee the content of this particular Initiator request. The endorsing token in this policy is an  
4027 sp:X509Token (P042)-(P046), which, in particular, is an sp:WssX509V3Token11, which must be used in  
4028 accordance with the WS-Security 1.1 Profile for X509 Tokens [\[WSS11-X509-PROFILE\]](#). (Note: it may be  
4029 the case that this X509 certificate is the same X509 certificate referred to in the ProtectionToken  
4030 assertion (P012). However, it is important to keep in mind that line (P012) only indicates that the Initiator's  
4031 token will be used by the Recipient to protect the ephemeral key for the symmetric binding. Therefore, the  
4032 fact that the token is identified in line (P012) as an sp:X509V3Token10 is not directly related to the fact  
4033 that the same key may be used for the additional purpose of explicitly signing the request message).

4034 The sp:Wss11 assertion (P049-P056) indicates that WS-Security 1.1 constructs are accepted. (Note also  
4035 that eitherWssX509V3Token10 or WssX509V3Token11 may be used with the Wss11 since both WS-  
4036 Security 1.0 and WS-Security 1.1 Profiles are supported by WS-Security 1.1)

4037 There are also 2 Policys, one each for the input message and the output message, each of which  
4038 contains an assertion indicating the message SOAP Body must be signed (P064)-(P066) for the input  
4039 message and (P077)-(P079) for the output message, and each contains an assertion that the message  
4040 SOAP Body must be encrypted (P067)-(P069) for the input message and (P080)-(P082) for the output  
4041 message.

4042 The following is a sample request that is compliant with this policy.

```
4043 (M001) <?xml version="1.0" encoding="utf-8" ?>
4044 (M002) <S12:Envelope
4045 (M003)   xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
4046 (M004)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4047 (M005)   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4048 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4049 wss-wssecurity-secext-1.0.xsd"
4050 (M007)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4051 wssecurity-utility-1.0.xsd"
4052 (M008)   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4053 (M009)   xmlns:xenc="..."
4054 (M010)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
4055 (M011) <S12:Header>
4056 (M012) <wsse:Security S12:mustUnderstand="1">
4057 (M013) <wsu:Timestamp wsu:Id="timestamp">
4058 (M014) <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
4059 (M015) </wsu:Timestamp>
4060 (M016) <xenc:EncryptedKey Id="encKey" >
4061 (M017) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
4062 (M018) <ds:DigestMethod Algorithm="...#sha1"/>
4063 (M019) </xenc:EncryptionMethod>
4064 (M020) <ds:KeyInfo >
4065 (M021) <wsse:SecurityTokenReference >
4066 (M022) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
4067 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
4068 (M023) </wsse:SecurityTokenReference>
4069 (M024) </ds:KeyInfo>
4070 (M025) <xenc:CipherData>
4071 (M026) <xenc:CipherValue>TE...=</xenc:CipherValue>
4072 (M027) </xenc:CipherData>
4073 (M028) </xenc:EncryptedKey>
4074 (M029) <n1:ReferenceList xmlns:n1=".../xmlenc#">
4075 (M030) <n1:DataReference URI="#encBody"/>
4076 (M031) </n1:ReferenceList>
```

```

4077 (M032) <saml:Assertion
4078 (M033)   AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
4079 (M034)   IssueInstant="2003-04-17T00:46:02Z"
4080 (M035)   Issuer="www.opensaml.org"
4081 (M036)   MajorVersion="1"
4082 (M037)   MinorVersion="1">
4083 (M038)   <saml:Conditions
4084 (M039)     NotBefore="2002-06-19T16:53:33.173Z"
4085 (M040)     NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
4086 (M041)   <saml:AttributeStatement>
4087 (M042)     <saml:Subject>
4088 (M043)       <saml:NameIdentifier
4089 (M044)         NameQualifier="www.example.com"
4090 (M045)         Format="">
4091 (M046)         uid=joe,ou=people,ou=saml-demo,o=example.com
4092 (M047)       </saml:NameIdentifier>
4093 (M048)     <saml:SubjectConfirmation>
4094 (M049)       <saml:ConfirmationMethod>
4095 (M050)         urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
4096 (M051)       </saml:ConfirmationMethod>
4097 (M052)     </saml:SubjectConfirmation>
4098 (M053)   </saml:Subject>
4099 (M054)   <saml:Attribute>
4100 (M055)     ...
4101 (M056)   </saml:Attribute>
4102 (M057)   ...
4103 (M058) </saml:AttributeStatement>
4104 (M059) </saml:Assertion>
4105 (M060) <wsse:SecurityTokenReference wsu:id="STR1">
4106 (M061)   <wsse:KeyIdentifier wsu:id="..."
4107 (M062)     ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
4108 profile-1.0#SAMLAssertionID">
4109 (M063)     a75adf55-01d7-40cc-929f-dbd8372ebdfc
4110 (M064)   </wsse:KeyIdentifier>
4111 (M065) </wsse:SecurityTokenReference>
4112 (M066) <wsse:BinarySecurityToken
4113 (M067)   wsu:Id="attesterCert"
4114 (M068)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4115 wss-x509-token-profile-1.0#X509v3"
4116 (M069)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
4117 200401-wss-soap-message-security-1.0#Base64Binary">
4118 (M070)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
4119 (M071) </wsse:BinarySecurityToken>
4120 (M072) <ds:Signature wsu:Id="message-signature">
4121 (M073)   <ds:SignedInfo>
4122 (M074)     <ds:CanonicalizationMethod
4123 (M075)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
4124 (M076)     <ds:SignatureMethod
4125 (M077)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
4126 (M078)     <ds:Reference URI="#STR1">
4127 (M079)       <ds:Transforms>
4128 (M080)         <ds:Transform
4129 (M081)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
4130 200401-wss-soap-message-security-1.0#STR-Transform">
4131 (M082)       <wsse:TransformationParameters>
4132 (M083)         <ds:CanonicalizationMethod
4133 (M084)           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
4134 (M085)         </wsse:TransformationParameters>
4135 (M086)       </ds:Transform>
4136 (M087)     </ds:Transforms>
4137 (M088)     <ds:DigestMethod
4138 (M089)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4139 (M090)     <ds:DigestValue>...</ds:DigestValue>
4140 (M091)   </ds:Reference>

```

```

4141 (M092) <ds:Reference URI="#MsgBody">
4142 (M093) <ds:DigestMethod
4143 (M094) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4144 (M095) <ds:DigestValue>...</ds:DigestValue>
4145 (M096) </ds:Reference>
4146 (M097) <ds:Reference URI="#timestamp">
4147 (M098) <ds:DigestMethod
4148 (M099) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4149 (M100) <ds:DigestValue>...</ds:DigestValue>
4150 (M101) </ds:Reference>
4151 (M102) </ds:SignedInfo>
4152 (M103) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
4153 (M104) <ds:KeyInfo>
4154 (M105) <wsse:SecurityTokenReference wsu:id="STR2">
4155 (M106) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
4156 (M107) </wsse:SecurityTokenReference>
4157 (M108) </ds:KeyInfo>
4158 (M109) </ds:Signature>
4159 (M110) <ds:Signature wsu:Id="endorsing-signature">
4160 (M111) <ds:SignedInfo>
4161 (M112) ...
4162 (M113) <ds:Reference URI="#message-signature">
4163 (M114) ...
4164 (M115) </ds:Reference>
4165 (M116) </ds:SignedInfo>
4166 (M117) <ds:SignatureValue>Bu ...=</ds:SignatureValue>
4167 (M118) <ds:KeyInfo>
4168 (M119) <wsse:SecurityTokenReference >
4169 (M120) <wsse:Reference URI="#attesterCert" ValueType="...#X509v3"/>
4170 (M121) </wsse:SecurityTokenReference>
4171 (M122) </ds:KeyInfo>
4172 (M123) </ds:Signature>
4173 (M124) </wsse:Security>
4174 (M125) </S12:Header>
4175 (M126) <S12:Body wsu:Id="MsgBody">
4176 (M127) <xenc:EncryptedData Id="encBody" Type="...#Content"
4177 MimeType="text/xml" Encoding="UTF-8" >
4178 (M128) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
4179 (M129) <ds:KeyInfo >
4180 (M130) <wsse:SecurityTokenReference>
4181 (M131) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
4182 (M132) </wsse:SecurityTokenReference>
4183 (M133) </ds:KeyInfo>
4184 (M134) <xenc:CipherData>
4185 (M135) <xenc:CipherValue>70...=</xenc:CipherValue>
4186 (M136) </xenc:CipherData>
4187 (M137) </xenc:EncryptedData>
4188 (M138) </S12:Body>
4189 (M139) </S12:Envelope>

```

4190

4191 The message above contains a request compliant with the policy described in (P001)-(P072), which  
4192 includes the endpoint policy and the input policy.

4193 Lines (M016)-(M028) contain an xenc:EncryptedKey element that contains an Initiator-generated  
4194 symmetric signing and encryption key, K, that can be used to decrypt the xenc:EncryptedData contained  
4195 in the SOAP S12:Body (M0126) as indicated by the wsse:SecurityTokenReference (M0130)-(M0132) that  
4196 uses a direct reference to the Id of the xenc:EncryptedKey, "encKey" on lines (M0131) and (M016).

4197 The xenc:EncryptedKey contains a dsig KeyInfo (M020)-(M024) reference to the Thumbprint of the  
4198 Recipient's public X509 certificate (M022). This complies with the policy (P068) that requires the sp:Body  
4199 to of the input message to be encrypted. It also complies with the policy (P013) to protect using  
4200 encryption based on X509 token and to refer to it by Thumbprint (P014).

4201 Lines (M029)-(M031) contain an xenc:ReferenceList referring to the xenc:EncryptedData in the S12:Body  
4202 (M0127)-(M0137).

4203 Lines (M032)-(M059) contain the saml:Assertion as a SignedSupportingToken as required by the  
4204 endpoint policy (P035). A saml:Assertion used as a SignedSupportingToken uses the “sender-vouches”  
4205 ConfirmationMethod (M049)-(M051) as described in the introductory section 2.3.

4206 Lines (M060)-(M065) contain a WS-Security wsse:SecurityTokenReference that has a KeyIdentifier of  
4207 ValueType “...1.0#SAMLAssertionID”, which indicates a SAML 1.1 Assertion as described in the WS-  
4208 Security 1.1 [[SAML1120\\_TOKEN\\_PROFILE](#)].

4209 Lines (M066)-(M071) contain a wsse:BinarySecurityToken that contains the Initiator’s X509 certificate for  
4210 use as an EndorsingSupportingToken as required by line (P042) of the sp:EndorsingSupportingTokens  
4211 assertion (P040)-(P048).

4212 Lines (M072)-(M0109) contain the message signature. The dsig Signature covers the following:

- 4213 ➤ The SAML Assertion using the ds:Reference (M078)-(M091), which uses the WS-Security STR-  
4214 Transform technique to refer to the SAML Assertion indirectly through the  
4215 wsse:SecurityTokenReference (M060) described above. This signature is required by the  
4216 sp:SignedSupportingTokens assertion (P031)-(P039).
- 4217 ➤ The message sp:Body (M0126)-(M0138) using the ds:Reference (M092)-(M096) as required by  
4218 the input message policy (P065).
- 4219 ➤ The message wsu:Timestamp (M013)-(M015) using the ds:Reference (M097)-(M0101) as  
4220 required by the endpoint policy (P027).

4221 The key used to sign the message signature is referenced in the dsig KeyInfo (M0104)-(M0108), which  
4222 contains a SecurityTokenReference with a direct URI reference to the xenc:EncryptedKey, “encKey”,  
4223 which contains the Initiator-generated signing and encryption key, K, as described above.

4224 Lines (M0110)-(M0123) contain the endorsing signature. The dsig endorsing Signature covers the  
4225 following:

- 4226 ➤ The message Signature (M072)-(M0109) using the ds:Reference (M0113)-(M0115), which is a  
4227 direct URI reference to the Id “message-signature” on line (M072).

4228 This signature is required by the policy EndorsingSupportingTokens assertion (P040)-(P048) to be an  
4229 X509 certificate (P044). The dsig KeyInfo (M0118)-(M0122) contains a WS-Security  
4230 SecurityTokenReference with a direct URI to the Initiator’s X509 certificate on line (M066)-(M071) with Id  
4231 = “attesterCert”.

4232 Lines (M0127)-(M0137) contain the xenc:EncryptedData, which contains the SOAP S12:Body message  
4233 payload that is encrypted using the encryption key, K, contained in the xenc:EncryptedKey CipherValue  
4234 (M026), which can only be decrypted using the Recipient’s X509 certificate referred to by  
4235 ThumbprintSHA1 on line (M022).

### 4236 **2.3.2.5 (WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt**

4237 This scenario is based on WS-SX Interop Scenarios Phase 2 (October 31, 2006) [[WSSX-WSTR-WSSC-](#)  
4238 [INTEROP](#)] Scenario 5 (Client and STS: Mutual Certificate WSS1.1 (section 3.5 of interop ref), Client and  
4239 Service: Issued SAML 1.1 Token for Certificate WSS1.1 (section 4.3 of interop ref)).

4240 In this scenario, the service specifies that the client must obtain an IssuedToken from a designated  
4241 Security Token Service (STS), which must be a SAML 1.1 Assertion. The Assertion contains an  
4242 ephemeral key K2 in an EncryptedKey element encrypted using service’s certificate. The client also  
4243 obtains the same ephemeral key K2 from the RequestedProofToken returned by the STS. The body of  
4244 the message from the client to the service is signed using DKT1(K2), encrypted using DKT2(K2), and  
4245 endorsed using DKT3(K2), which are keys the client derives from K2 using the algorithm specified in  
4246 section 7 of [[WS-SecureConversation](#)]. The response from the service is also signed using derived keys.  
4247 In a simpler alternative, ephemeral key K itself could be used for message protection.

4248 Note: In this scenario, in terms of Figure 1 [[Figure01](#)], the STS (Issuing Authority) is the  
4249 Issuer of the SAML 1.1 holder-of-key Assertion that contains the ephemeral symmetric  
4250 key K2. The service (combined Recipient/RelyingParty) can trust the client (combined

4251 Initiator/Requestor) that uses the ephemeral symmetric key K2 obtained from the  
4252 RequestedProofToken, because the same key gets delivered to the service in the SAML  
4253 1.1 holder-of-key Assertion, which is signed by the trusted Authority.

4254 This scenario is a 2-step sequence from a WS-SP perspective. These 2 steps will be described at a high  
4255 level first in order to explain the context for the policies and messages that follow.

4256 **In step 1** the following takes place:

- 4257 • the client accesses the service's WS-SP policy (P001 -> P116 below), and determines that it  
4258 needs to use an IssuedToken from a Security Token Service (STS), specified in the policy.  
4259 This IssuedToken will serve as the basis of trust by the service. Effectively, the service only trusts  
4260 the client because the client is able to obtain the IssuedToken from the STS.

4261 To complete step 1, the client will then do the following:

- 4262 • the client will access the STS and process the STS policy (PSTS-001 -> PSTS-098 below)
- 4263 • based on the STS policy, the client will send a request to the STS for an IssuedToken (MSTS-001  
4264 -> MSTS-0231 below)
- 4265 • the STS will send a response to the client containing the IssuedToken, which in this example is a  
4266 SAML 1.1 holder-of-key Assertion (RSTS-001 -> RSTS-0263 below)

4267 **In step 2** the following takes place:

- 4268 • because the client now has the IssuedToken it is now able fulfill the requirements of the service's  
4269 WS-SP policy (P001-P116 below) that it accessed in step 1
- 4270 • based on the service's policy the clients sends a request to the service (M001-M0229 below)
- 4271 • the service will send a response to the client containing the requested resource data (R001-  
4272 R0153 below)

4273 As an aid to understanding the security properties of the policies and messages in this example, the  
4274 following is a list of identifiers used in the text descriptions to reference the cryptographic keys that are  
4275 called for in the policies and used in the messages in this example:

- 4276 ➤ **X509T1**: Client's (Requestor/Initiator) X509 certificate used for authentication to the STS.
- 4277 ➤ **X509T2**: STS' (Issuing Authority) X509 certificate used to encrypt keys sent to STS, used to  
4278 authenticate STS to Service.
- 4279 ➤ **X509T3**: Service's (Recipient/RelyingParty/ValidatingAuthority) X509 certificate used to encrypt  
4280 keys sent to Service.
- 4281 ➤ **K1**: Client-generated ephemeral symmetric key for crypto communication to the STS.
- 4282 ➤ **K2**: Client-generated ephemeral symmetric key for crypto communication between the Client and  
4283 the Service.
- 4284 ➤ **K3**: STS-generated proof key for use by Client to authenticate with Service via saml:Assertion.
- 4285 ➤ **DKT1(K2)**: Client-generated derived key used for signing requests to the Service.
- 4286 ➤ **DKT2(K2)**: Client-generated derived key used for encrypting requests to the Service.
- 4287 ➤ **DKT3(K3)**: Client-generated derived key used for endorsing signed requests to the Service.
- 4288 ➤ **DKT4(K2)**: Service-generated derived key used for encrypting responses to the Client.
- 4289 ➤ **DKT5(K2)**: Service-generated derived key used for signing responses to the Client.

4290

4291 Here is the WS-SP Policy that the Service presents to a Client:

4292

```
4293 (P001) <wsp:Policy wsu:Id="Service5-Policy"  
4294 (P002)  
4295 (P003) xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4296 (P004) xmlns:sp"..."  
4297 (P005)
```

```

4298 (P006)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4299 wssecurity-utility-1.0.xsd"
4300 (P007)      >
4301 (P008)      <wsp:ExactlyOne>
4302 (P009)      <wsp:All>
4303 (P010)      <sp:SymmetricBinding>
4304 (P011)      <wsp:Policy>
4305 (P012)      <sp:ProtectionToken>
4306 (P013)      <wsp:Policy>
4307 (P014)      <sp:X509Token IncludeToken=
4308 (P015)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4309 /Never">
4310 (P016)      <wsp:Policy>
4311 (P017)      <sp:RequireDerivedKeys/>
4312 (P018)      <sp:RequireThumbprintReference/>
4313 (P019)      <sp:WssX509V3Token10/>
4314 (P020)      </wsp:Policy>
4315 (P021)      </sp:X509Token>
4316 (P022)      </wsp:Policy>
4317 (P023)      </sp:ProtectionToken>
4318 (P024)      <sp:AlgorithmSuite>
4319 (P025)      <wsp:Policy>
4320 (P026)      <sp:Basic256/>
4321 (P027)      </wsp:Policy>
4322 (P028)      </sp:AlgorithmSuite>
4323 (P029)      <sp:Layout>
4324 (P030)      <wsp:Policy>
4325 (P031)      <sp:Strict/>
4326 (P032)      </wsp:Policy>
4327 (P033)      </sp:Layout>
4328 (P034)      <sp:IncludeTimestamp/>
4329 (P035)      <sp:OnlySignEntireHeadersAndBody/>
4330 (P036)      </wsp:Policy>
4331 (P037)      </sp:SymmetricBinding>
4332 (P038)      <sp:EndorsingSupportingTokens>
4333 (P039)      <wsp:Policy>
4334 (P040)      <sp:IssuedToken IncludeToken=
4335 (P041)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4336 /AlwaysToRecipient">
4337 (P042)      <sp:Issuer>
4338 (P043)      <a:Address>http://example.com/STS</a:Address>
4339 (P044)      </sp:Issuer>
4340 (P045)      <sp:RequestSecurityTokenTemplate>
4341 (P046)      <t:TokenType
4342 (P047)      >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4343 1.1#SAMLV1.1</t:TokenType>
4344 (P048)      <t:KeyType
4345 (P049)      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:KeyTy
4346 pe>
4347 (P050)      <t:KeySize>256</t:KeySize>
4348 (P051)      <t:CanonicalizationAlgorithm
4349 (P052)      >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorithm>
4350 (P053)      <t:EncryptionAlgorithm
4351 (P054)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4352 (P055)      <t:EncryptWith
4353 (P056)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4354 (P057)      <t:SignWith
4355 (P058)      >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4356 (P059)      </sp:RequestSecurityTokenTemplate>
4357 (P060)      <wsp:Policy>
4358 (P061)      <sp:RequireDerivedKeys/>
4359 (P062)      <sp:RequireInternalReference/>
4360 (P063)      </wsp:Policy>
4361 (P064)      </sp:IssuedToken>

```

```

4362 (P065)         </wsp:Policy>
4363 (P066)         </sp:EndorsingSupportingTokens>
4364 (P067)         <sp:Wss11>
4365 (P068)         <wsp:Policy>
4366 (P069)         <sp:MustSupportRefKeyIdentifier/>
4367 (P070)         <sp:MustSupportRefIssuerSerial/>
4368 (P071)         <sp:MustSupportRefThumbprint/>
4369 (P072)         <sp:MustSupportRefEncryptedKey/>
4370 (P073)         <sp:RequireSignatureConfirmation/>
4371 (P074)         </wsp:Policy>
4372 (P075)         </sp:Wss11>
4373 (P076)         <sp:Trust13>
4374 (P077)         <wsp:Policy>
4375 (P078)         <sp:MustSupportIssuedTokens/>
4376 (P079)         <sp:RequireClientEntropy/>
4377 (P080)         <sp:RequireServerEntropy/>
4378 (P081)         </wsp:Policy>
4379 (P082)         </sp:Trust13>
4380 (P083)         </wsp:All>
4381 (P084)         </wsp:ExactlyOne>
4382 (P085)         </wsp:Policy>
4383
4384 (P086)         <wsp:Policy wsu:Id="InOut-Policy"
4385 (P087)
4386 (P088)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4387 (P089)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4388 (P090)
4389 (P091)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4390 (P092)         wssecurity-utility-1.0.xsd">
4391 (P092)         <wsp:ExactlyOne>
4392 (P093)         <wsp:All>
4393 (P094)         <sp:SignedParts>
4394 (P095)         <sp:Body/>
4395 (P096)         <sp:Header Name="To"
4396 (P097)         Namespace="http://www.w3.org/2005/08/addressing"/>
4397 (P098)         <sp:Header Name="From"
4398 (P099)         Namespace="http://www.w3.org/2005/08/addressing"/>
4399 (P100)         <sp:Header Name="FaultTo"
4400 (P101)         Namespace="http://www.w3.org/2005/08/addressing"/>
4401 (P102)         <sp:Header Name="ReplyTo"
4402 (P103)         Namespace="http://www.w3.org/2005/08/addressing"/>
4403 (P104)         <sp:Header Name="MessageID"
4404 (P105)         Namespace="http://www.w3.org/2005/08/addressing"/>
4405 (P106)         <sp:Header Name="RelatesTo"
4406 (P107)         Namespace="http://www.w3.org/2005/08/addressing"/>
4407 (P108)         <sp:Header Name="Action"
4408 (P109)         Namespace="http://www.w3.org/2005/08/addressing"/>
4409 (P110)         </sp:SignedParts>
4410 (P111)         <sp:EncryptedParts>
4411 (P112)         <sp:Body/>
4412 (P113)         </sp:EncryptedParts>
4413 (P114)         </wsp:All>
4414 (P115)         </wsp:ExactlyOne>
4415 (P116)         </wsp:Policy>

```

4416  
4417 When a Client encounters the above Service Policy, it determines that it must obtain an IssuedToken  
4418 from the designated Issuer. After obtaining the IssuedToken, the client may send the request in  
4419 accordance with the rest of the policy. Details of the Service Policy follow. The STS Policy its details  
4420 follow after the Service Policy.

4421 Lines (P001)-(P085) above contain the Service Endpoint wsp:Policy, which contains a wsp:All assertion  
4422 that requires all 4 of its contained assertions to be complied with: sp:SymmetricBinding (P010)-(P037),  
4423 sp:EndorsingSupportingTokens (P038)-(P066), sp:Wss11 (P067)-(P075), and sp:Trust13 (P076)-(P082).

4424 Lines (P010)-(P037) contain the **sp:SymmetricBinding assertion** that requires that derived keys (DKT1,  
4425 DKT2, DKT3) be used to protect the message (P017) and that the ephemeral key (K2) used to derive  
4426 these keys be encrypted using the service's X509 certificate (X509T3) as specified by the sp:X509Token  
4427 assertion (P014)-(P021), which also indicates that the X509 token, itself should not be sent (P015), but  
4428 that a Thumbprint reference (P018) to it be sent. Finally, the sp:SymmetricBinding specifies that the  
4429 sp:Basic256 sp:AlgorithmSuite be used (P024)-(P028), that sp:Strict sp:Layout be used (P029)-(P033),  
4430 that an sp:Timestamp be included (P034), and that only the complete message Body and Headers be  
4431 signed (P035), where the Headers part means that only direct child elements of the WS-Security SOAP  
4432 header element be signed.

4433 Lines (P038)-(P066) contain the **sp:EndorsingSupportingTokens assertion** that an sp:IssuedToken  
4434 (P040)-(P064) be used to sign the message signature and that the IssuedToken must be included with  
4435 the request (P040)-(P041). Lines (P042)-(P044) specify the address of the SecurityTokenService (STS)  
4436 from which the IssuedToken must be obtained. Lines (P045)-(P059) contain an  
4437 sp:RequestSecurityTokenTemplate (P046)-(P058) which contains explicit WS-Trust elements that the  
4438 client should directly copy to a t:SecondaryParameters element to include with the WS-Trust  
4439 t:RequestSecurityToken to the STS to obtain the sp:IssuedToken. Of particular interest here is that the  
4440 t:TokenType (P046)-(P047) requested is a SAML 1.1 Assertion, which will be used to contain the  
4441 ephemeral symmetric key (K2) (P048)-(P049). K2 will be used for communication between the Client and  
4442 the Service. K2 will be encrypted by the STS using the Service's X509 certificate (X509T3). The Client is  
4443 also informed by the IssuedToken assertion that the IssuedToken may only be referenced internally  
4444 within the message (P062) and that the ephemeral key (K2) associated with the IssuedToken be used by  
4445 the Client to derive the keys (DKT1(K2), DKT2(K2), DKT3(K2)) used in the Client's request to the Service.

4446 Lines (P067)-(P075) contain the **sp:Wss11 assertion** that indicates that WS-Security 1.1 will be used,  
4447 which includes Wss11-only features such as Thumbprint (P073), EncryptedKey (P074), and  
4448 SignatureConfirmation (P075).

4449 Lines (P076)-(P082) contain the **sp:Trust13 assertion** that indicates the Client should expect to use WS-  
4450 Trust 1.3 ([WSTRUST](#)) to obtain the IssuedToken from the STS.

4451 Lines (P086)-(P0116) contain the **operation input and output policies** that the client should use to  
4452 determine what parts of the messages are to be signed (P094)-(P0110) and encrypted (P0111)-(P0113).

4453

4454 Here is the WS-SP Policy that the STS presents to a Client:

```
4455 (PSTS-001) <wsp:Policy wsu:Id="STS5-Policy"
4456 (PSTS-002)
4457 (PSTS-003)   xmlns:wsp="http://www.w3.org/ns/ws-policy"
4458 (PSTS-004)   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4459 (PSTS-005)
4460 (PSTS-006)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4461 (PSTS-007)   wss-wssecurity-utility-1.0.xsd"
4462 (PSTS-007)   >
4463 (PSTS-008)   <wsp:ExactlyOne>
4464 (PSTS-009)     <wsp>All>
4465 (PSTS-010)       <sp:SymmetricBinding>
4466 (PSTS-011)         <wsp:Policy>
4467 (PSTS-012)           <sp:ProtectionToken>
4468 (PSTS-013)             <wsp:Policy>
4469 (PSTS-014)               <sp:X509Token IncludeToken=
4470 (PSTS-015)                 "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4471 (PSTS-016)                 ken/Never">
4472 (PSTS-016)                   <wsp:Policy>
4473 (PSTS-017)                     <sp:RequireThumbprintReference/>
4474 (PSTS-018)                     <sp:WssX509V3Token10/>
4475 (PSTS-019)                   </wsp:Policy>
4476 (PSTS-020)                 </sp:X509Token>
4477 (PSTS-021)               </wsp:Policy>
4478 (PSTS-022)             </sp:ProtectionToken>
4479 (PSTS-023)           <sp:AlgorithmSuite>
4480 (PSTS-024)             <wsp:Policy>
4481 (PSTS-025)               <sp:Basic256/>
```

```

4482 (PSTS-026) </wsp:Policy>
4483 (PSTS-027) </sp:AlgorithmSuite>
4484 (PSTS-028) <sp:Layout>
4485 (PSTS-029) <wsp:Policy>
4486 (PSTS-030) <sp:Strict/>
4487 (PSTS-031) </wsp:Policy>
4488 (PSTS-032) </sp:Layout>
4489 (PSTS-033) <sp:IncludeTimestamp/>
4490 (PSTS-034) <sp:OnlySignEntireHeadersAndBody/>
4491 (PSTS-035) </wsp:Policy>
4492 (PSTS-036) </sp:SymmetricBinding>
4493 (PSTS-037) <sp:EndorsingSupportingTokens>
4494 (PSTS-038) <wsp:Policy>
4495 (PSTS-039) <sp:X509Token IncludeToken=
4496 (PSTS-040) "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4497 ken/AlwaysToRecipient">
4498 (PSTS-041) <wsp:Policy>
4499 (PSTS-042) <sp:RequireThumbprintReference/>
4500 (PSTS-043) <sp:WssX509V3Token10/>
4501 (PSTS-044) </wsp:Policy>
4502 (PSTS-045) </sp:X509Token>
4503 (PSTS-046) </wsp:Policy>
4504 (PSTS-047) </sp:EndorsingSupportingTokens>
4505 (PSTS-048) <sp:Wss11>
4506 (PSTS-049) <wsp:Policy>
4507 (PSTS-050) <sp:MustSupportRefKeyIdentifier/>
4508 (PSTS-051) <sp:MustSupportRefIssuerSerial/>
4509 (PSTS-052) <sp:MustSupportRefThumbprint/>
4510 (PSTS-053) <sp:MustSupportRefEncryptedKey/>
4511 (PSTS-054) <sp:RequireSignatureConfirmation/>
4512 (PSTS-055) </wsp:Policy>
4513 (PSTS-056) </sp:Wss11>
4514 (PSTS-057) <sp:Trust13>
4515 (PSTS-058) <wsp:Policy>
4516 (PSTS-059) <sp:MustSupportIssuedTokens/>
4517 (PSTS-060) <sp:RequireClientEntropy/>
4518 (PSTS-061) <sp:RequireServerEntropy/>
4519 (PSTS-062) </wsp:Policy>
4520 (PSTS-063) </sp:Trust13>
4521 (PSTS-064) </wsp:All>
4522 (PSTS-065) </wsp:ExactlyOne>
4523 (PSTS-066) </wsp:Policy>
4524
4525 (PSTS-067) <wsp:Policy wsu:Id="InOut-Policy"
4526 (PSTS-068)
4527 (PSTS-069) xmlns:wsp="http://www.w3.org/ns/ws-policy"
4528 (PSTS-070) xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4529 (PSTS-071)
4530 (PSTS-072) xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4531 wss-wssecurity-utility-1.0.xsd"
4532 (PSTS-073) >
4533 (PSTS-074) <wsp:ExactlyOne>
4534 (PSTS-075) <wsp:All>
4535 (PSTS-076) <sp:SignedParts>
4536 (PSTS-077) <sp:Body/>
4537 (PSTS-078) <sp:Header Name="To"
4538 (PSTS-079) Namespace="http://www.w3.org/2005/08/addressing"/>
4539 (PSTS-080) <sp:Header Name="From"
4540 (PSTS-081) Namespace="http://www.w3.org/2005/08/addressing"/>
4541 (PSTS-082) <sp:Header Name="FaultTo"
4542 (PSTS-083) Namespace="http://www.w3.org/2005/08/addressing"/>
4543 (PSTS-084) <sp:Header Name="ReplyTo"
4544 (PSTS-085) Namespace="http://www.w3.org/2005/08/addressing"/>
4545 (PSTS-086) <sp:Header Name="MessageID"

```

```

4546 (PSTS-087)      Namespace="http://www.w3.org/2005/08/addressing"/>
4547 (PSTS-088)      <sp:Header Name="RelatesTo"
4548 (PSTS-089)      Namespace="http://www.w3.org/2005/08/addressing"/>
4549 (PSTS-090)      <sp:Header Name="Action"
4550 (PSTS-091)      Namespace="http://www.w3.org/2005/08/addressing"/>
4551 (PSTS-092)      </sp:SignedParts>
4552 (PSTS-093)      <sp:EncryptedParts>
4553 (PSTS-094)      <sp:Body/>
4554 (PSTS-095)      </sp:EncryptedParts>
4555 (PSTS-096)      </wsp:All>
4556 (PSTS-097)      </wsp:ExactlyOne>
4557 (PSTS-098)      </wsp:Policy>

```

4558 Above is the STS Policy that the Client will encounter when obtaining the IssuedToken required by the  
4559 Service Policy. This policy is quite similar in detail to the Service Policy and therefore only the differences  
4560 that are noteworthy will be discussed in the details below.

4561 Similar to the Service Policy, the STS endpoint Policy contains 4 assertions to be complied with:  
4562 sp:SymmetricBinding (PSTS-010)-(PSTS-036), sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047),  
4563 sp:Wss11 (PSTS-048)-(PSTS-056), and sp:Trust13 (PSTS-057)-(PSTS-063).

4564 Lines (PSTS-010)-(PSTS-036) contain the **sp:SymmetricBinding assertion**, where the main difference  
4565 from the previous policy is that here the sp:ProtectionToken is an sp:X509Token that will be used to  
4566 encrypt the ephemeral client-generated key (K1) that will be used for Client-STS communication. Derived  
4567 keys will not be required in this communication.

4568 Lines (PSTS-037)-(PSTS-047) contain the **sp:EndorsingSupportingTokens assertion**, which in this  
4569 case contains an sp:X509Token assertion (PSTS-039)-(PSTS-045) that requires the Client to include  
4570 (PSTS-040) its X509 certificate, which must be used to sign the message signature. The STS uses this  
4571 mechanism to authenticate the Client.

4572 The **sp:Wss11 assertion** (PSTS-048)-(PSTS-056), **sp:Trust13 assertion** (PSTS-057)-(PSTS-063) and  
4573 the **operation input and output policies** (PSTS-067)-(PSTS-098) are the same as those described for  
4574 the Service policy above (P067)-(P0116).

4575

4576 Below are included sample messages that comply with the above policies. The messages are presented  
4577 in the same order that they would be used in a real scenario and therefore the Client-STS request  
4578 (MSTS-001)-(MSTS-0231) and response (RSTS-001)-(RSTS-0263) are presented first, which are then  
4579 followed by the Client-Service request (M001-M229) and response (R001)-(R153).

4580 Here is an example Client request to the STS:

```

4581 (MSTS-001)      <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
4582 (MSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
4583 (MSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
4584 (MSTS-004)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4585 (MSTS-005)      wssecurity-secext-1.0.xsd"
4586 (MSTS-005)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4587 (MSTS-005)      wssecurity-utility-1.0.xsd" >
4588 (MSTS-006)      <s:Header>
4589 (MSTS-007)      <a:Action s:mustUnderstand="1" u:Id="_3">
4590 (MSTS-008)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
4591 (MSTS-009)      </a:Action>
4592 (MSTS-010)      <a:MessageID u:Id="_4">
4593 (MSTS-011)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4594 (MSTS-012)      </a:MessageID>
4595 (MSTS-013)      <a:ReplyTo u:Id="_5">
4596 (MSTS-014)      <a:Address>
4597 (MSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4598 (MSTS-016)      </a:Address>
4599 (MSTS-017)      </a:ReplyTo>
4600 (MSTS-018)      <a:To s:mustUnderstand="1" u:Id="_6">
4601 (MSTS-019)      http://server.example.com/STS/Scenarios5-6
4602 (MSTS-020)      </a:To>
4603 (MSTS-021)      <o:Security s:mustUnderstand="1">

```

```

4604 (MSTS-022) <u:Timestamp
4605 (MSTS-023)   u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4606 (MSTS-024)   <u:Created>2005-10-25T00:47:36.144Z</u:Created>
4607 (MSTS-025)   <u:Expires>2005-10-25T00:52:36.144Z</u:Expires>
4608 (MSTS-026) </u:Timestamp>
4609 (MSTS-027) <e:EncryptedKey
4610 (MSTS-028)   Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-9">
4611 (MSTS-029)   <e:EncryptionMethod Algorithm=
4612 (MSTS-030)     "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
4613 (MSTS-031)   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
4614 (MSTS-032)     <o:SecurityTokenReference>
4615 (MSTS-033)       <o:KeyIdentifier ValueType=
4616 (MSTS-034)       "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4617 (MSTS-035)         1.1.xsd#ThumbprintSHA1">
4618 (MSTS-036)         W+rgYBmLmVEG//scD7Vo8Kq5G7I=
4619 (MSTS-037)       </o:KeyIdentifier>
4620 (MSTS-038)     </o:SecurityTokenReference>
4621 (MSTS-039)   </KeyInfo>
4622 (MSTS-040)   <e:CipherData>
4623 (MSTS-041)     <e:CipherValue>
4624 (MSTS-042)       <!--base64 encoded cipher-->
4625 (MSTS-043)     </e:CipherValue>
4626 (MSTS-044)   </e:CipherData>
4627 (MSTS-045)   <e:ReferenceList>
4628 (MSTS-046)     <e:DataReference URI="#_2"/>
4629 (MSTS-047)   </e:ReferenceList>
4630 (MSTS-048) </e:EncryptedKey>
4631 (MSTS-049) <o:BinarySecurityToken
4632 (MSTS-050)   u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"
4633 (MSTS-051)   ValueType=
4634 (MSTS-052)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4635 (MSTS-053)   profile-1.0#X509v3"
4636 (MSTS-054)   EncodingType=
4637 (MSTS-055)   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
4638 (MSTS-056)   message-security-1.0#Base64Binary">
4639 (MSTS-057)
4640 (MSTS-058) MIIDDDCCafSgAwIBAgIQM6YEf7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4w
4641 (MSTS-059) DAYDVQKDAVPQVNJUzEeMBwGAlUEAwVt0FTSVMgSW50ZXJvcCBUZXN0IENBMB4XDTA1
4642 (MSTS-060) MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVowQjEOMAwGAlUECgwFT0FTSVMxIDAeBgNV
4643 (MSTS-061) BAsMF09BU01TIEIudGVyb3AgVGZzdCBDZXJ0MQ4wDAYDVQDDAVBbG1jZTCBnzANBggkq
4644 (MSTS-062) hkiG9w0BAQEFAAOBjQAwYkCgYEAoqi99By1VYo0aHrkKCNT4DkIgL/SgahbeKdGhrb
4645 (MSTS-063) u3K2XG7arfD9tqIBIKMfrx4Gp90NJa85AV1yiNsEyyvq+mUnMpNcKnLXLOjkTmMcqDYbb
4646 (MSTS-064) kehJlXPnaWLzve+mW0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz+EzsCAwEA
4647 (MSTS-065) AaOBkzCBkDAJBgnVHRMEAjAAMDGAlUdHwQsMCowKKImhiRodHRWoi8vaW50ZXJvcC5i
4648 (MSTS-066) YnRlc3QubmV0L2Nybc9jYs5jcmwwDgYDVR0PAQH/BAQDAgSwMB0GAlUdDgQWBQK410T
4649 (MSTS-067) UHZ1QV3V2QtllNDm+PoxiDafBgNVHSMEGDAWgBTAnSj8wes1oR3WqqqgHBpNwkkPdZAN
4650 (MSTS-068) BgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW+6yrLXyU1P2xJbEkohXHI5OWwKWleOb9h1kh
4651 (MSTS-069) WntUalfcFOJAgUyH30TtpHldzx1+vK2LPzhoUFKYHE1IyQvokBN2JjFO64BqkCKnZhl
4652 (MSTS-070) dLRPxBghkTdxQgdf5rCK/wh3xVsZCNTfuMnmlAM61OAg8QduDah3WFZpEA0s2nwQaCNQ
4653 (MSTS-071) TNmjJC8tav1CBR6+E5FAMwPXP7pJxn9Fw9OXRYqBRA4v2y7YpbGkG2GI9UvOHW6SGvf4
4654 (MSTS-072) FRStHMMO35YbpikGsLix3vAsXWwi4rWfVOYzQK00FPNi9RMCUdSH06m9uLWckiCxjos0
4655 (MSTS-073) FQODZE9l4ATGy9s9hNVwryOJTW==
4656 (MSTS-074) </o:BinarySecurityToken>
4657 (MSTS-075) <Signature Id="_0" xmlns="http://www.w3.org/2000/09/xmldsig#">
4658 (MSTS-076)   <SignedInfo>
4659 (MSTS-077)     <CanonicalizationMethod Algorithm=
4660 (MSTS-078)       "http://www.w3.org/2001/10/xml-exc-c14n#">
4661 (MSTS-079)     <SignatureMethod Algorithm=
4662 (MSTS-080)       "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
4663 (MSTS-081)     <Reference URI="#_1">
4664 (MSTS-082)       <Transforms>
4665 (MSTS-083)         <Transform Algorithm=
4666 (MSTS-084)           "http://www.w3.org/2001/10/xml-exc-c14n#">
4667 (MSTS-085)         </Transforms>

```

```

4668 (MSTS-083) <DigestMethod Algorithm=
4669 (MSTS-084) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4670 (MSTS-085) <DigestValue>VX1fCPwCzVsSc1hZf0BSbCgW2hM=</DigestValue>
4671 (MSTS-086) </Reference>
4672 (MSTS-087) <Reference URI="#_3">
4673 (MSTS-088) <Transforms>
4674 (MSTS-089) <Transform Algorithm=
4675 (MSTS-090) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4676 (MSTS-091) </Transforms>
4677 (MSTS-092) <DigestMethod Algorithm=
4678 (MSTS-093) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4679 (MSTS-094) <DigestValue>FwiFAUuqNDo9SDkk5A28Mg7Pa8Q=</DigestValue>
4680 (MSTS-095) </Reference>
4681 (MSTS-096) <Reference URI="#_4">
4682 (MSTS-097) <Transforms>
4683 (MSTS-098) <Transform Algorithm=
4684 (MSTS-099) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4685 (MSTS-100) </Transforms>
4686 (MSTS-101) <DigestMethod Algorithm=
4687 (MSTS-102) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4688 (MSTS-103) <DigestValue>oM59PsOTpMrDdOcwXYQzjVU10xw=</DigestValue>
4689 (MSTS-104) </Reference>
4690 (MSTS-105) <Reference URI="#_5">
4691 (MSTS-106) <Transforms>
4692 (MSTS-107) <Transform Algorithm=
4693 (MSTS-108) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4694 (MSTS-109) </Transforms>
4695 (MSTS-110) <DigestMethod Algorithm=
4696 (MSTS-111) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4697 (MSTS-112) <DigestValue>KIK3vk1FN1QmMdQkplq2azfzrzg=</DigestValue>
4698 (MSTS-113) </Reference>
4699 (MSTS-114) <Reference URI="#_6">
4700 (MSTS-115) <Transforms>
4701 (MSTS-116) <Transform Algorithm=
4702 (MSTS-117) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4703 (MSTS-118) </Transforms>
4704 (MSTS-119) <DigestMethod Algorithm=
4705 (MSTS-120) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4706 (MSTS-121) <DigestValue>RJEE3hrcyCD6PzFJo6fyut6biVg=</DigestValue>
4707 (MSTS-122) </Reference>
4708 (MSTS-123) <Reference
4709 (MSTS-124) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4710 (MSTS-125) <Transforms>
4711 (MSTS-126) <Transform Algorithm=
4712 (MSTS-127) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4713 (MSTS-128) </Transforms>
4714 (MSTS-129) <DigestMethod Algorithm=
4715 (MSTS-130) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4716 (MSTS-131) <DigestValue>zQdN5XpejfqXn0Wko0m51ZYiasE=</DigestValue>
4717 (MSTS-132) </Reference>
4718 (MSTS-133) </SignedInfo>
4719 (MSTS-134) <SignatureValue
4720 (MSTS-135) >iHGJ+xV2VZTjM1Rc7AQJrwLY/aM=</SignatureValue>
4721 (MSTS-136) <KeyInfo>
4722 (MSTS-137) <o:SecurityTokenReference>
4723 (MSTS-138) <o:Reference
4724 (MSTS-139) ValueType=
4725 (MSTS-140) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4726 (MSTS-141) 1.1.xsd#EncryptedKey"
4727 (MSTS-141) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-9"/>
4728 (MSTS-142) </o:SecurityTokenReference>
4729 (MSTS-143) </KeyInfo>
4730 (MSTS-144) </Signature>
4731 (MSTS-145) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

```

```

4732 (MSTS-0146) <SignedInfo>
4733 (MSTS-0147)   <CanonicalizationMethod Algorithm=
4734 (MSTS-0148)     "http://www.w3.org/2001/10/xml-exc-c14n#" />
4735 (MSTS-0149)   <SignatureMethod Algorithm=
4736 (MSTS-0150)     "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4737 (MSTS-0151)   <Reference URI="#_0">
4738 (MSTS-0152)     <Transforms>
4739 (MSTS-0153)       <Transform Algorithm=
4740 (MSTS-0154)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4741 (MSTS-0155)     </Transforms>
4742 (MSTS-0156)     <DigestMethod Algorithm=
4743 (MSTS-0157)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4744 (MSTS-0158)     <DigestValue>UZKtShk8q6iu9WR5uQZp04iAitg=</DigestValue>
4745 (MSTS-0159)     </Reference>
4746 (MSTS-0160)   </SignedInfo>
4747 (MSTS-0161)   <SignatureValue>
4748 (MSTS-0162)   Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbmCtZFuQz9tnW35Lmst6vI
4749 (MSTS-0163)   RefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLfAo14=
4750 (MSTS-0164)   </SignatureValue>
4751 (MSTS-0165)   <KeyInfo>
4752 (MSTS-0166)     <o:SecurityTokenReference>
4753 (MSTS-0167)       <o:Reference
4754 (MSTS-0168)         ValueType=
4755 (MSTS-0169)       "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4756 (MSTS-0170)       profile-1.0#X509v3"
4757 (MSTS-0171)         URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-6" />
4758 (MSTS-0172)       </o:SecurityTokenReference>
4759 (MSTS-0173)     </KeyInfo>
4760 (MSTS-0174)   </Signature>
4761 (MSTS-0175)   </o:Security>
4762 (MSTS-0176)   </s:Header>
4763 (MSTS-0177)   <s:Body u:Id="_1">
4764 (MSTS-0178)     <e:EncryptedData
4765 (MSTS-0179)       Id="_2"
4766 (MSTS-0180)       Type="http://www.w3.org/2001/04/xmlenc#Content">
4767 (MSTS-0181)       <e:EncryptionMethod Algorithm=
4768 (MSTS-0182)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
4769 (MSTS-0183)     <e:CipherData>
4770 (MSTS-0184)       <e:CipherValue>
4771 (MSTS-0185)         <!-- base64 encoded octets with encrypted RST request-->
4772 (MSTS-0186)         <!-- Unencrypted form: -->
4773 (MSTS-0187)         <!--
4774 (MSTS-0188)       <t:RequestSecurityToken>
4775 (MSTS-0189)         <t:RequestType>
4776 (MSTS-0190)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
4777 (MSTS-0191)         </t:RequestType>
4778 (MSTS-0192)       <wsp:AppliesTo
4779 (MSTS-0193)         xmlns:wsp="http://www.w3.org/ns/ws-policy">
4780 (MSTS-0194)         <a:EndpointReference
4781 (MSTS-0195)           xmlns:a="http://www.w3.org/2005/08/addressing">
4782 (MSTS-0196)             <a:Address
4783 (MSTS-0197)               >http://server.example.com/Scenarios5</a:Address>
4784 (MSTS-0198)           </a:EndpointReference>
4785 (MSTS-0199)         </wsp:AppliesTo>
4786 (MSTS-0200)     <t:Entropy>
4787 (MSTS-0201)       <t:BinarySecret
4788 (MSTS-0202)         u:Id="uuid-4acf589c-0076-4a83-8b66-5f29341514b7-3"
4789 (MSTS-0203)         Type=
4790 (MSTS-0204)           "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
4791 (MSTS-0205)       >Uv38QLxDQM9gLoDZ6OwYDiFk094nmwu3Wmay7EdKmhW=</t:BinarySecret>
4792 (MSTS-0206)     </t:Entropy>
4793 (MSTS-0207)     <t:KeyType>
4794 (MSTS-0208)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey
4795 (MSTS-0209)     </t:KeyType>

```

```

4796 (MSTS-0209) <t:KeySize>256</t:KeySize>
4797 (MSTS-0210) <t:ComputedKeyAlgorithm>
4798 (MSTS-0211) http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
4799 (MSTS-0212) </t:ComputedKeyAlgorithm>
4800 (MSTS-0213) <t:SecondaryParameters>
4801 (MSTS-0214) <t:TokenType>
4802 (MSTS-0215) >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4803 1.1#SAMLV1.1</t:TokenType>
4804 (MSTS-0216) <t:KeyType>
4805 (MSTS-0217) >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:Ke
4806 yType>
4807 (MSTS-0218) <t:KeySize>256</t:KeySize>
4808 (MSTS-0219) <t:CanonicalizationAlgorithm>
4809 (MSTS-0220) >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorith
4810 m>
4811 (MSTS-0221) <t:EncryptionAlgorithm>
4812 >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4813 (MSTS-0222) <t:EncryptWith>
4814 >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4815 (MSTS-0223) <t:SignWith>
4816 >http://www.w3.org/2000/09/xmlsig#hmac-sha1</t:SignWith>
4817 (MSTS-0224) </t:SecondaryParameters>
4818 (MSTS-0225) </t:RequestSecurityToken>
4819 (MSTS-0226) -->
4820 (MSTS-0227) </e:CipherValue>
4821 (MSTS-0228) </e:CipherData>
4822 (MSTS-0229) </e:EncryptedData>
4823 (MSTS-0230) </s:Body>
4824 (MSTS-0231) </s:Envelope>

```

4825 The message above is a request by the Client to the STS in compliance with the STS policy (PSTS-001)-  
4826 (PSTS-098). Salient features of this message appropriate to the compliance are described below.

4827 Lines (MSTS-027)-(MSTS-047) contain the **e:EncryptedKey element**, which contains the encrypted  
4828 client-generated ephemeral key (K1) (MSTS-039)-(MSTS-043). K1 is encrypted using the STS certificate  
4829 (X509T2) which is specified by its Thumbprint identifier in the WS-Security o:SecurityTokenReference  
4830 (MSTS-032)-(MSTS-037). The e:ReferenceList in the e:EncryptedKey (MSTS-044)-(MSTS-046) indicates  
4831 that the encryption key K1 is used to directly encrypt the message Body which is referenced by the  
4832 e:DataReference URI="#\_2" (MSTS-045).

4833 Lines (MSTS-048)-(MSTS-071) contain a **WS-Security o:BinarySecurityToken**, which contains the  
4834 Client's public X509 certificate (X509T1) that is required for Client authentication to the STS by the  
4835 sp:EndorsingToken in the STS policy (PSTS-037)-(PSTS-047).

4836 Lines (MSTS-072)-(MSTS-0144) contain the **WS-SP message signature** in an XML Digital Signature  
4837 (dsig) element (namespace = ...xmlsig (MSTS-072)). The elements covered by the dsig Signature are  
4838 identified in the dsig Reference elements:

- 4839 ➤ dsig Reference (MSTS-078) covers the s:Body (Id="\_1" (MSTS-0176))
- 4840 ➤ dsig Reference (MSTS-087) covers the a:Action (Id="\_3" (MSTS-007))
- 4841 ➤ dsig Reference (MSTS-096) covers the a:MessageID (Id="\_4" (MSTS-010))
- 4842 ➤ dsig Reference (MSTS-0105) covers the a:ReplyTo (Id="\_5" (MSTS-013))
- 4843 ➤ dsig Reference (MSTS-0114) covers the a:To (Id="\_6" (MSTS-018))
- 4844 ➤ dsig Reference (MSTS-0123) covers the u:Timestamp (Id="uuid ... 3849-10" (MSTS-023))

4845 all as required by the STS Input and Output Policy sp:SignedParts (PSTS-076)-(PSTS-092), which covers  
4846 the first 5 elements above (note: if an element is not present that is identified in the policy (such as  
4847 FaultTo or RelatesTo) it obviously is not required to be signed, but if present must be signed). The  
4848 u:Timestamp is required to be signed by its presence in the STS endpoint policy (PSTS-033).

4849 Lines (MSTS-0136)-(MSTS-0143) of the **WS-SP message signature contain the dsig KeyInfo**, which  
4850 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing  
4851 validation key (i.e. the key which can be used to verify this dsig:Signature), which in this case is contained

4852 in the e:EncryptedKey element (Id="uuid ... 3849-9" (MSTS-027)) that was described above. Note: at this  
4853 point to verify the Signature, the STS will decrypt the e:EncryptedKey using the private key from the STS  
4854 certificate, X509T2, which will produce the client-generated ephemeral signing and encryption key, K1,  
4855 which, in turn, may be used to validate the message signature. Note also: at this point the STS only  
4856 knows whether the data covered by the message signature is valid or not, but the STS does not yet know  
4857 the identity of the entity that actually sent the data. That is covered next:

4858 Lines (MSTS-0145)-(MSTS-0173) contain the **WS-SP endorsing signature**, also in an XML Digital  
4859 Signature element. The endorsing signature only covers one element, the message signature element,  
4860 which is identified by the dsig Reference element:

4861 ➤ dsig Reference (MSTS-0151) covers the dsig Signature (Id="\_0" (MSTS-072))  
4862 as required by the STS endpoint policy sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047).

4863 Lines (MSTS-0165)-(MSTS-0174) of the **WS-SP endorsing signature contain the dsig KeyInfo**, which  
4864 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing  
4865 validation key, which in this case is in the o:BinarySecurityToken element (Id="uuid ... 3849-6"  
4866 (MSTS-048)) that was also described above. Note: now the STS finally has the credentials necessary to  
4867 authenticate the Client. The STS will generally have an identity database of trusted clients and their  
4868 associated X509 certificates. If a client successfully produces a signature that can be validated by the  
4869 associated certificate in the STS database, which would have to match the certificate in the  
4870 o:BinarySecurityToken element, then the client is presumed to be authenticated to the level of security  
4871 provided by this mechanism (strong single factor authentication).

4872 Lines (MSTS-0176)-(MSTS-0230) contain the **SOAP message s:Body element**, which contains its  
4873 payload in an e:EncryptedData element (MSTS-0177)-(MSTS-0229). This e:EncryptedData element was  
4874 identified above by its Id="\_2" in the e:ReferenceList (MSTS-044) of the e:EncryptedKey that was  
4875 processed by the STS above to obtain the client ephemeral key (K1), with which this e:EncryptedData  
4876 can be decrypted. Generally, when the e:EncryptedKey is processed, this e:EncryptedData would have  
4877 been decrypted at that time and made available for processing in the event of successful Client  
4878 authentication as described above. Because the contents of this payload are relevant to the rest of this  
4879 example, the contents of the payload will be briefly described.

4880 Lines (MSTS-0187)-(MSTS-0225) contain the **decrypted contents of the SOAP message s:Body**  
4881 **element**, which contain a WS-Trust t:RequestSecurityToken element. The t:RequestType (MSTS-0189)  
4882 is "...Issue", which means this is a request to issue a security token, which is the main service of the STS.  
4883 The WS-Policy wsp:AppliesTo element (MSTS-0191)-(MSTS-0198) contains the a:Address of the service,  
4884 to which the Client is requesting access, a service which the STS is presumably responsible for  
4885 authenticating and equipping validated clients to enable their access to. In this case the service is  
4886 identified by the URL <http://server.example.com/Scenarios5>, which was part of the WS-SX Interop  
4887 [[WSSX-WSTR-WSSC-INTEROP](#)].

4888 Lines (MSTS-0199)-(MSTS-0205) contain the Client entropy required by the Service policy (P079), which  
4889 is entropy data provided by the Client to aid in production of the ephemeral key, K2, that will be used for  
4890 Client-Service communication. Lines (MSTS-0206)-(MSTS-0212) contain additional parameters used in  
4891 the WS-Trust interface to the STS that will not be described here.

4892 Lines (MSTS-0213)-(MSTS-0224) contain the WS-Trust t:SecondaryParameters, which contains the  
4893 contents of the Service policy's RequestSecurityTokenTemplate (P045)-(P059), which the Service  
4894 requires that the Client pass to the STS as described above. The t:SecondaryParameters contains the  
4895 information the Service has requested about the SAML 1.1 IssuedToken that the STS is being requested  
4896 to create and deliver to the Client in order the enable the Client to access the Service successfully.

4897 Assuming everything above has executed successfully, the STS will then issue the SAML 1.1  
4898 IssuedToken and return it to the Client in a WS-Trust t:RequestSecurityTokenResponse that is described  
4899 next.

4900

4901 Here is an example STS response to the above Client request:

```
4902 (RSTS-001) <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope  
4903 (RSTS-002) xmlns:a=http://www.w3.org/2005/08/addressing  
4904 (RSTS-003) xmlns:e=http://www.w3.org/2001/04/xmlenc#
```

```

4905 (RSTS-004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
4906 secext-1.1.xsd"
4907 (RSTS-005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4908 wssecurity-secext-1.0.xsd"
4909 (RSTS-006)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4910 wssecurity-utility-1.0.xsd" >
4911 (RSTS-007)      <s:Header>
4912 (RSTS-008)      <a:Action s:mustUnderstand="1" u:Id="_4">
4913 (RSTS-009)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
4914 (RSTS-010)      </a:Action>
4915 (RSTS-011)      <a:RelatesTo u:Id="_5">
4916 (RSTS-012)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4917 (RSTS-013)      </a:RelatesTo>
4918 (RSTS-014)      <a:To s:mustUnderstand="1" u:Id="_6">
4919 (RSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4920 (RSTS-016)      </a:To>
4921 (RSTS-017)      <o:Security s:mustUnderstand="1">
4922 (RSTS-018)      <u:Timestamp
4923 (RSTS-019)      u:Id="uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4924 (RSTS-020)      <u:Created>2005-10-25T00:47:38.718Z</u:Created>
4925 (RSTS-021)      <u:Expires>2005-10-25T00:52:38.718Z</u:Expires>
4926 (RSTS-022)      </u:Timestamp>
4927 (RSTS-023)      <e:ReferenceList>
4928 (RSTS-024)      <e:DataReference URI="#_3"/>
4929 (RSTS-025)      </e:ReferenceList>
4930 (RSTS-026)      <k:SignatureConfirmation u:Id="_0"
4931 (RSTS-027)      Value="iHGJ+xV2VZTjM1Rc7AQJrWLY/aM="/>
4932 (RSTS-028)      <k:SignatureConfirmation u:Id="_1"
4933 (RSTS-029)      Value=
4934 (RSTS-030)      "Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWlcmG3xGLZYcc8tbmCtZFuQz9tnW35
4935 (RSTS-031)      Lmst6vRefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLfAo14="/>
4936 (RSTS-032)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
4937 (RSTS-033)      <SignedInfo>
4938 (RSTS-034)      <CanonicalizationMethod Algorithm=
4939 (RSTS-035)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4940 (RSTS-036)      <SignatureMethod Algorithm=
4941 (RSTS-037)      "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
4942 (RSTS-038)      <Reference URI="#_2">
4943 (RSTS-039)      <Transforms>
4944 (RSTS-040)      <Transform Algorithm=
4945 (RSTS-041)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4946 (RSTS-042)      </Transforms>
4947 (RSTS-043)      <DigestMethod Algorithm=
4948 (RSTS-044)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4949 (RSTS-045)      <DigestValue>kKx5bpLLlyucgXQ6exv/PbjSf1A=</DigestValue>
4950 (RSTS-046)      </Reference>
4951 (RSTS-047)      <Reference URI="#_4">
4952 (RSTS-048)      <Transforms>
4953 (RSTS-049)      <Transform Algorithm=
4954 (RSTS-050)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4955 (RSTS-051)      </Transforms>
4956 (RSTS-052)      <DigestMethod Algorithm=
4957 (RSTS-053)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4958 (RSTS-054)      <DigestValue>LB+VGn4fP2z45jg0Mdzyo8yTAWQ=</DigestValue>
4959 (RSTS-055)      </Reference>
4960 (RSTS-056)      <Reference URI="#_5">
4961 (RSTS-057)      <Transforms>
4962 (RSTS-058)      <Transform Algorithm=
4963 (RSTS-059)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4964 (RSTS-060)      </Transforms>
4965 (RSTS-061)      <DigestMethod Algorithm=
4966 (RSTS-062)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4967 (RSTS-063)      <DigestValue>izHLxm6V4Lc3Pss9Y6VRv3I5RPw=</DigestValue>
4968 (RSTS-064)      </Reference>

```

```

4969 (RSTS-065) <Reference URI="#_6">
4970 (RSTS-066) <Transforms>
4971 (RSTS-067) <Transform Algorithm=
4972 (RSTS-068) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4973 </Transforms>
4974 (RSTS-069) <DigestMethod Algorithm=
4975 (RSTS-070) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4976 (RSTS-071) <DigestValue>6LS4X08vC/GMGay2vwmD8fL7J2U=</DigestValue>
4977 (RSTS-072) </Reference>
4978 (RSTS-073) <Reference
4979 (RSTS-074) URI="#uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4980 (RSTS-075) <Transforms>
4981 (RSTS-076) <Transform Algorithm=
4982 (RSTS-077) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4983 (RSTS-078) </Transforms>
4984 (RSTS-079) <DigestMethod Algorithm=
4985 (RSTS-080) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4986 (RSTS-081) <DigestValue>uXGSpCBfbT1fLNBLdGMgy6DGDio=</DigestValue>
4987 (RSTS-082) </Reference>
4988 (RSTS-083) <Reference URI="#_0">
4989 (RSTS-084) <Transforms>
4990 (RSTS-085) <Transform Algorithm=
4991 (RSTS-086) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4992 (RSTS-087) </Transforms>
4993 (RSTS-088) <DigestMethod Algorithm=
4994 (RSTS-089) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4995 (RSTS-090) <DigestValue>z86w+GrzqRZF56ciuz6ogzVXAUA=</DigestValue>
4996 (RSTS-091) </Reference>
4997 (RSTS-092) <Reference URI="#_1">
4998 (RSTS-093) <Transforms>
4999 (RSTS-094) <Transform Algorithm=
5000 (RSTS-095) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5001 (RSTS-096) </Transforms>
5002 (RSTS-097) <DigestMethod Algorithm=
5003 (RSTS-098) "http://www.w3.org/2000/09/xmldsig#sha1"/>
5004 (RSTS-099) <DigestValue>Z9Tfd20a5aGngsyOPEvQuE0urvQ=</DigestValue>
5005 (RSTS-0100) </Reference>
5006 (RSTS-0101) </SignedInfo>
5007 (RSTS-0102) <SignatureValue>Q7HhboPUaZyXqUKgG7NCYlhMTXI=</SignatureValue>
5008 (RSTS-0103) <KeyInfo>
5009 (RSTS-0104) <o:SecurityTokenReference
5010 (RSTS-0105) k:TokenType="http://docs.oasis-open.org/wss/
5011 (RSTS-0106) oasis-wss-soap-message-security-1.1#EncryptedKey"
5012 (RSTS-0107) xmlns:k="http://docs.oasis-open.org/wss/
5013 (RSTS-0108) oasis-wss-wssecurity-secext-1.1.xsd">
5014 (RSTS-0109) <o:KeyIdentifier ValueType=
5015 (RSTS-0110) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5016 (RSTS-0111) 1.1.xsd#EncryptedKeySHA1">
5017 (RSTS-0112) CixQW5yEb3mw6XYqD8Ysvrf8cwI=
5018 (RSTS-0113) </o:KeyIdentifier>
5019 (RSTS-0114) </o:SecurityTokenReference>
5020 (RSTS-0115) </KeyInfo>
5021 (RSTS-0116) </Signature>
5022 (RSTS-0117) </o:Security>
5023 (RSTS-0118) </s:Header>
5024 (RSTS-0119) <s:Body u:Id="_2">
5025 (RSTS-0120) <e:EncryptedData Id="_3"
5026 (RSTS-0121) Type="http://www.w3.org/2001/04/xmlenc#Content">
5027 (RSTS-0122) <e:EncryptionMethod Algorithm=
5028 (RSTS-0123) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5029 (RSTS-0124) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5030 (RSTS-0125) <o:SecurityTokenReference
5031 (RSTS-0126) k:TokenType="http://docs.oasis-open.org/wss/
5032 (RSTS-0127) oasis-wss-soap-message-security-1.1#EncryptedKey"

```

```

5033 (RSTS-0127)          xmlns:k="http://docs.oasis-open.org/wss/
5034 (RSTS-0128)          oasis-wss-wssecurity-secext-1.1.xsd">
5035 (RSTS-0129)          <o:KeyIdentifier ValueType=
5036 (RSTS-0130)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5037 (RSTS-0131)          1.1.xsd#EncryptedKeySHA1">
5038 (RSTS-0131)          CixQW5yEb3mw6XYqD8Ysvrf8cwI=
5039 (RSTS-0132)          </o:KeyIdentifier>
5040 (RSTS-0133)          </o:SecurityTokenReference>
5041 (RSTS-0134)          </KeyInfo>
5042 (RSTS-0135)          <e:CipherData>
5043 (RSTS-0136)          <e:CipherValue>
5044 (RSTS-0137)          <!--base64 encoded octets of encrypted RSTR-->
5045 (RSTS-0138)          <!--
5046 (RSTS-0139)          <t:RequestSecurityTokenResponseCollection>
5047 (RSTS-0140)          <t:RequestSecurityTokenResponse>
5048 (RSTS-0141)          <t:TokenType>
5049 (RSTS-0142)          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5050 (RSTS-0143)          1.1#SAMLV1.1
5051 (RSTS-0143)          </t:TokenType>
5052 (RSTS-0144)          <t:KeySize>256</t:KeySize>
5053 (RSTS-0145)          <t:RequestedAttachedReference>
5054 (RSTS-0146)          <o:SecurityTokenReference>
5055 (RSTS-0147)          <o:KeyIdentifier ValueType=
5056 (RSTS-0148)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5057 (RSTS-0148)          1.0#SAMLAssertionID">
5058 (RSTS-0149)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5059 (RSTS-0150)          </o:KeyIdentifier>
5060 (RSTS-0151)          </o:SecurityTokenReference>
5061 (RSTS-0152)          </t:RequestedAttachedReference>
5062 (RSTS-0153)          <t:RequestedUnattachedReference>
5063 (RSTS-0154)          <o:SecurityTokenReference>
5064 (RSTS-0155)          <o:KeyIdentifier ValueType=
5065 (RSTS-0156)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5066 (RSTS-0156)          1.0#SAMLAssertionID">
5067 (RSTS-0157)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5068 (RSTS-0158)          </o:KeyIdentifier>
5069 (RSTS-0159)          </o:SecurityTokenReference>
5070 (RSTS-0160)          </t:RequestedUnattachedReference>
5071 (RSTS-0161)          <t:Lifetime>
5072 (RSTS-0162)          <u:Created>2005-10-24T20:19:26.526Z</u:Created>
5073 (RSTS-0163)          <u:Expires>2005-10-25T06:24:26.526Z</u:Expires>
5074 (RSTS-0164)          </t:Lifetime>
5075 (RSTS-0165)          <t:RequestedSecurityToken>
5076 (RSTS-0166)          <saml:Assertion MajorVersion="1" MinorVersion="1"
5077 (RSTS-0167)          AssertionID="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16"
5078 (RSTS-0168)          Issuer="Test STS" IssueInstant="2005-10-24T20:24:26.526Z"
5079 (RSTS-0169)          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
5080 (RSTS-0170)          <saml:Conditions NotBefore="2005-10-24T20:19:26.526Z"
5081 (RSTS-0171)          NotOnOrAfter="2005-10-25T06:24:26.526Z">
5082 (RSTS-0172)          <saml:AudienceRestrictionCondition>
5083 (RSTS-0173)          <saml:Audience
5084 (RSTS-0174)          >http://server.example.com/Scenarios5</saml:Audience>
5085 (RSTS-0175)          </saml:AudienceRestrictionCondition>
5086 (RSTS-0176)          </saml:Conditions>
5087 (RSTS-0177)          <saml:Advice>
5088 (RSTS-0178)          </saml:Advice>
5089 (RSTS-0179)          <saml:AttributeStatement>
5090 (RSTS-0180)          <saml:Subject>
5091 (RSTS-0181)          <saml:SubjectConfirmation>
5092 (RSTS-0182)          <saml:ConfirmationMethod>
5093 (RSTS-0183)          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
5094 (RSTS-0184)          </saml:ConfirmationMethod>
5095 (RSTS-0185)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5096 (RSTS-0186)          <e:EncryptedKey

```

```

5097 (RSTS-0187)          xmlns:e="http://www.w3.org/2001/04/xmlenc#">
5098 (RSTS-0188)          <e:EncryptionMethod Algorithm=
5099 (RSTS-0189)          "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
5100 (RSTS-0190)          </e:EncryptionMethod>
5101 (RSTS-0191)          <KeyInfo>
5102 (RSTS-0192)          <o:SecurityTokenReference xmlns:o=
5103 (RSTS-0193)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5104 (RSTS-0194)          secext-1.0.xsd">
5105 (RSTS-0194)          <o:KeyIdentifier ValueType=
5106 (RSTS-0195)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5107 (RSTS-0196)          1.1.xsd#ThumbprintSHA1">
5108 (RSTS-0196)          NQM0IBvuplAtETQvk+6gn8C13wE=
5109 (RSTS-0197)          </o:KeyIdentifier>
5110 (RSTS-0198)          </o:SecurityTokenReference>
5111 (RSTS-0199)          </KeyInfo>
5112 (RSTS-0200)          <e:CipherData>
5113 (RSTS-0201)          <e:CipherValue>
5114 (RSTS-0202)          EEcYjwNoYcJ+20xTYE5e/fixl5KOgzgrfaYAxkDFv/VXiuKfl084h8PmogTfM+azcgAf
5115 (RSTS-0203)          mArVQvOyKWXRb5vmXYfvHLlhZTbXacy+nowSUNnEjp37VDbI3RJ5k6tBHF+ow0NM/P6G
5116 (RSTS-0204)          PNZ9ZqJi11GDgWJkFsJzNZXNbbMgwuFu3cA=</e:CipherValue>
5117 (RSTS-0205)          </e:CipherData>
5118 (RSTS-0206)          </e:EncryptedKey>
5119 (RSTS-0207)          </KeyInfo>
5120 (RSTS-0208)          </saml:SubjectConfirmation>
5121 (RSTS-0209)          </saml:Subject>
5122 (RSTS-0210)          </saml:AttributeStatement>
5123 (RSTS-0211)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5124 (RSTS-0212)          <SignedInfo>
5125 (RSTS-0213)          <CanonicalizationMethod Algorithm=
5126 (RSTS-0214)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5127 (RSTS-0215)          </CanonicalizationMethod>
5128 (RSTS-0216)          <SignatureMethod Algorithm=
5129 (RSTS-0217)          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
5130 (RSTS-0218)          </SignatureMethod>
5131 (RSTS-0219)          <Reference
5132 (RSTS-0220)          URI="#uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16">
5133 (RSTS-0221)          <Transforms>
5134 (RSTS-0222)          <Transform Algorithm=
5135 (RSTS-0223)          "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
5136 (RSTS-0224)          </Transform>
5137 (RSTS-0225)          <Transform Algorithm=
5138 (RSTS-0226)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5139 (RSTS-0227)          </Transform>
5140 (RSTS-0228)          </Transforms>
5141 (RSTS-0229)          <DigestMethod Algorithm=
5142 (RSTS-0230)          "http://www.w3.org/2000/09/xmldsig#sha1">
5143 (RSTS-0231)          </DigestMethod>
5144 (RSTS-0232)          <DigestValue
5145 (RSTS-0233)          >7nHBrFPsm+LEFAoV4NoQPoEl5Lk=</DigestValue>
5146 (RSTS-0234)          </Reference>
5147 (RSTS-0235)          </SignedInfo>
5148 (RSTS-0236)          <SignatureValue
5149 (RSTS-0237)          >TugV4pTIwCH87bLD4jiMgVGtkbRBt1tRlHXJArL34A/YfA4AnGBLXB4pJdUsUxMUTbQ
5150 (RSTS-0238)          14PoGgEsdLNg8C77peARELGP1/Tqw7T3u5zBYHxCHCiV2FWBBfeOmwJmgoaBf8XZJ4Al
5151 (RSTS-0239)          yqPq61P61jrQjZJafpHuYpAZnZQSVsiJaBPQ=</SignatureValue>
5152 (RSTS-0240)          <KeyInfo>
5153 (RSTS-0241)          <o:SecurityTokenReference xmlns:o=
5154 (RSTS-0242)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5155 (RSTS-0243)          secext-1.0.xsd">
5156 (RSTS-0243)          <o:KeyIdentifier ValueType=
5157 (RSTS-0244)          http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5158 (RSTS-0244)          1.1.xsd#ThumbprintSHA1
5159 (RSTS-0245)          >W+rgYBmLmVEG//scD7Vo8Kq5G7I=</o:KeyIdentifier>
5160 (RSTS-0246)          </o:SecurityTokenReference>

```

```

5161 (RSTS-0247) </KeyInfo>
5162 (RSTS-0248) </Signature>
5163 (RSTS-0249) </saml:Assertion>
5164 (RSTS-0250) </t:RequestedSecurityToken>
5165 (RSTS-0251) <t:RequestedProofToken>
5166 (RSTS-0252) <t:BinarySecret
5167 (RSTS-0253) u:Id="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-14"
5168 (RSTS-0254) >zT8LWAUwUrIVKA/rkCr0kx1EmKAehcB6TGWJuAgucBM=</t:BinarySecret>
5169 (RSTS-0255) </t:RequestedProofToken>
5170 (RSTS-0256) </t:RequestSecurityTokenResponse>
5171 (RSTS-0257) </t:RequestSecurityTokenResponseCollection>
5172 (RSTS-0258) -->
5173 (RSTS-0259) </e:CipherValue>
5174 (RSTS-0260) </e:CipherData>
5175 (RSTS-0261) </e:EncryptedData>
5176 (RSTS-0262) </s:Body>
5177 (RSTS-0263) </s:Envelope>

```

5178 From here on the description will be less detailed except where new concepts that have not been covered  
5179 previously in this example. For instance, tracing the dsig References to their associated elements and  
5180 policy requirements will not be detailed since it follows the same patterns that have just been described in  
5181 the message above.

5182 The message above is a response from the STS to the Client that contains the requested security tokens  
5183 that the Client needs to access the Service.

5184 Lines (RSTS-023)-(RSTS-025) contain a **standalone e:ReferenceList** that has an e:DataReference  
5185 (Id="\_3") that points to the e:EncryptedData element in the SOAP s:Body (RSTS-0119). This  
5186 e:DataReference will be used later in the processing of this message.

5187 Lines (RSTS-026)-(RSTS-031) contain **2 WS-Security 1.1 k:SignatureConfirmation elements** that  
5188 indicate to the Client that the STS has processed the data in the Client request and in particular, has  
5189 processed the information covered by the 2 dsig Signatures in that request, that are identified by their  
5190 respective dsig SignatureValue elements (see lines (MSTS-0134)-(MSTS-0135) and (MSTS-0161)-  
5191 (MSTS-0163) in the Client request to the STS above to compare SignatureValues).

5192 Lines (RSTS-032)-(RSTS-0115) contain the **WS-SP message signature** for this message.

5193 Lines (RSTS-0103)-(RSTS-0114) contain the **WS-SP message signature dsig KeyInfo element**, which  
5194 contains a WS-Security 1.1 o:SecurityTokenReference, which contains an o:KeyIdentifier of ValueType  
5195 "... EncryptedKeySHA1". This is a WS-Security 1.1 construct [[WS\\_SECURITY\\_11](#)] used to reference a  
5196 security token that is not included in the message, which in this case is the Client-generated ephemeral  
5197 key, K1, that the Client used to prepare the request and delivered to the STS in the e:EncryptedKey  
5198 element in that request (MSTS-027) that was described in detail above. The contents of the  
5199 o:KeyIdentifier (RSTS-0111) consist of the SHA1 of K1. This should be sufficient information to let the  
5200 Client know that the STS used its X509 certificate, X509T2, to decrypt K1 from the e:EncryptedKey in the  
5201 Client request, which will enable the Client to trust the contents of this STS response.

5202 Lines (RSTS-0119)-(RSTS-0261) contain the **e:EncryptedData**, which is also provided in decrypted form  
5203 for instructive purposes. As mentioned above, this element is referred to by the standalone  
5204 e:ReferenceList element in the WS-Security header (RSTS-023) and as a result can be not tied to any  
5205 particular encryption key as described in [[WS\\_SECURITY\\_11](#)], and since referenced will be processed by  
5206 WS-Security.

5207 Lines (RSTS-0123)-(RSTS-0134) contain the **e:EncryptedData dsig KeyInfo**, which refers to the same  
5208 ephemeral key, K1, as described above for the message signature for this message.

5209 Lines (RSTS-0140)-(RSTS-0256) contain the **decrypted WS-Trust t:RequestSecurityTokenResponse**  
5210 from the STS. Briefly, lines (RSTS-0145)-(RSTS-0160) contain WS-Security o:SecurityTokenReference  
5211 elements that refer to the SAML 1.1 Assertion that is provided below. These are convenience elements  
5212 for the Client to use in subsequent message preparation where the SAML Assertion is used.

5213 Lines (RSTS-0165)-(RSTS-0250) contain **the actual t:RequestedSecurityToken** that has been the main  
5214 object of discussion up until this point, which is the SAML 1.1 Assertion, saml:Assertion (RSTS-0166)-  
5215 (RSTS-0249), that is the sp:IssuedToken provided by the STS for the Client to use to access the Service.

5216 Lines (RSTS-0170)-(RSTS-0176) of the `saml:Assertion` contain the **saml:Conditions** that specify that this  
5217 token is intended only for the Service, identified by the URL in the **saml:Audience** element (RSTS-0174).

5218 Lines (RSTS-0181)-(RSTS-0208) contain the **all-important saml:SubjectConfirmation element**, which  
5219 contains the **STS-generated encrypted ephemeral key, K3**, that will be used by the Client and Service  
5220 to communicate securely. There are **2 copies of this key, K3**, in this `t:RequestSecurityTokenResponse`.  
5221 This copy is for the Service. The Client's copy is described below. In any event, the  
5222 `saml:SubjectConfirmation` element contains a `dsig:KeyInfo` (RSTS-0185)-(RSTS-0207), which contains an  
5223 `e:EncryptedKey` element (RSTS-0186)-(RSTS-0206), which, in turn, contains a second `dsig:KeyInfo`,  
5224 which contains a `WS-Security o:SecurityTokenReference` element that contains an `o:KeyIdentifier` (RSTS-  
5225 0194)-(RSTS-0197) that identifies the key, X509T3, the Service's public X509 certificate, that can be  
5226 used by the Service to decrypt the ultimate object here, which is the ephemeral key, K3, contained in the  
5227 `e:CipherData` (RSTS-0200)-(RSTS-0205). The Service's public X509 certificate is identified by its `WS-`  
5228 `Security 1.1 ThumbprintSHA1` (RSTS-0196). Therefore, when the Service receives this `saml:Assertion`, it  
5229 has the ability to obtain the ephemeral key, K3, contained in the `saml:SubjectConfirmation`, with which it  
5230 can securely communicate with the Client, based on the assurances provided by the STS.

5231 Lines (RSTS-0211)-(RSTS-0248) contain the **saml:Assertion dsig Signature** that is contained in and  
5232 covers the `saml:Assertion` via the `saml:AssertionID`, the value of which appears on lines (RSTS-0220) and  
5233 (RSTS-0167).

5234 Lines (RSTS-0240)-(RSTS-0247) contain the **saml:Assertion dsig Signature KeyInfo element**, which  
5235 contains the `ThumbprintSHA1` of the **STS public key, X509T2**, which is the same certificate that the  
5236 Client referenced in the `e:EncryptedKey` when it made initial contact with the STS above (MSTS-035).  
5237 This completes the initial discussion of the characteristics of the IssuedToken `saml:Assertion` that will be  
5238 used in the remainder of this example.

5239 Lines (RSTS-0251)-(RSTS-0255) of the `t:RequestSecurityTokenResponse` contains the  
5240 **t:RequestedProofToken**, which contains the **Client copy of the STS-generated ephemeral key, K3**,  
5241 which will be used in the Client-Service communication to authenticate the client to the Service.

5242 At this point we have completed the setup portion of this example that has enabled secure trusted  
5243 communication between the Client and Service as governed by an STS. The exact strength of the  
5244 security protecting this example would be the subject of an official security analysis that is beyond the  
5245 scope of this document, however, the intent has been to provide sufficient detail that all parties concerned  
5246 with such an analysis would have sufficient context to understand and evaluate such an analysis.

5247

5248 Here is an example of a Client request to the Service using the tokens from the STS response:

5249

```
5250 (M001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope  
5251 (M002)   xmlns:a=http://www.w3.org/2005/08/addressing  
5252 (M003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#  
5253 (M004)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
5254 (M005)   wssecurity-secext-1.0.xsd"  
5255 (M006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-  
5256 (M007)   secureconversation/200512"  
5257 (M008)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-  
5258 (M009)   wssecurity-utility-1.0.xsd" >  
5259 (M010)   <s:Header>  
5260 (M011)     <a:Action s:mustUnderstand="1" u:Id="_5">  
5261 (M012)       http://example.org/Ping  
5262 (M013)     </a:Action>  
5263 (M014)     <a:MessageID u:Id="_6">  
5264 (M015)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423  
5265 (M016)     </a:MessageID>  
5266 (M017)     <a:ReplyTo u:Id="_7">  
5267 (M018)       <a:Address>  
5268 (M019)         http://www.w3.org/2005/08/addressing/anonymous  
5269 (M020)       </a:Address>  
5270 (M021)     </a:ReplyTo>  
5271 (M022)     <a:To s:mustUnderstand="1" u:Id="_8">  
5272 (M023)       http://server.example.com/Scenarios5
```

```

5273 (M021) </a:To>
5274 (M022) <o:Security s:mustUnderstand="1">
5275 (M023) <u:Timestamp
5276 (M024) <u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5277 (M025) <u:Created>2005-10-25T00:47:38.222Z</u:Created>
5278 (M026) <u:Expires>2005-10-25T00:52:38.222Z</u:Expires>
5279 (M027) </u:Timestamp>
5280 (M028) <e:EncryptedKey
5281 (M029) <Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-4">
5282 (M030) <e:EncryptionMethod Algorithm=
5283 (M031) http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5284 (M032) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5285 (M033) <o:SecurityTokenReference>
5286 (M034) <o:KeyIdentifier ValueType=
5287 (M035) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5288 1.1.xsd#ThumbprintSHA1">
5289 (M036) NQM0IBvuplAtETQvk+6gn8C13wE=
5290 (M037) </o:KeyIdentifier>
5291 (M038) </o:SecurityTokenReference>
5292 (M039) </KeyInfo>
5293 (M040) <e:CipherData>
5294 (M041) <e:CipherValue>
5295 (M042) <!-- base64 encoded octets of encrypted key K2 -->
5296 (M043) </e:CipherValue>
5297 (M044) </e:CipherData>
5298 (M045) </e:EncryptedKey>
5299 (M046) <sc:DerivedKeyToken u:Id="_0" >
5300 (M047) <o:SecurityTokenReference>
5301 (M048) <o:Reference
5302 (M049) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5303 (M050) </o:SecurityTokenReference>
5304 (M051) <sc:Offset>0</sc:Offset>
5305 (M052) <sc:Length>24</sc:Length>
5306 (M053) <sc:Nonce>7hI6U16OHavfYgpquHWuQ==</sc:Nonce>
5307 (M054) </sc:DerivedKeyToken>
5308 (M055) <sc:DerivedKeyToken u:Id="_2">
5309 (M056) <o:SecurityTokenReference>
5310 (M057) <o:Reference
5311 (M058) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5312 (M059) </o:SecurityTokenReference>
5313 (M060) <sc:Nonce>OEu+WEEUxPFRQK7SCFAnEQ==</sc:Nonce>
5314 (M061) </sc:DerivedKeyToken>
5315 (M062) <e:ReferenceList>
5316 (M063) <e:DataReference URI="#_4"/>
5317 (M064) </e:ReferenceList>
5318 (M065) <!-- encrypted SAML assertion -->
5319 (M066) <e:EncryptedData Id="_3"
5320 (M067) Type="http://www.w3.org/2001/04/xmlenc#Element">
5321 (M068) <e:EncryptionMethod Algorithm=
5322 (M069) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5323 (M070) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5324 (M071) <!-- encrypted Key K2 -->
5325 (M072) <e:EncryptedKey>
5326 (M073) <e:EncryptionMethod Algorithm=
5327 (M074) "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5328 (M075) <KeyInfo>
5329 (M076) <o:SecurityTokenReference>
5330 (M077) <o:KeyIdentifier ValueType=
5331 (M078) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5332 1.1.xsd#ThumbprintSHA1">
5333 (M079) NQM0IBvuplAtETQvk+6gn8C13wE=
5334 (M080) </o:KeyIdentifier>
5335 (M081) </o:SecurityTokenReference>
5336 (M082) </KeyInfo>

```

```

5337 (M083) <e:CipherData>
5338 (M084) <e:CipherValue>
5339 (M085)
5340 cb7+JW2idPNSarK9quqCe9PQwmW2hoUghuyKRe+I9zOts6HaMcg73LqCWuK/jtdpvNl6
5341 (M086) GT/ZDYfcAJ7NLyMGxSiwi4DULtOShqS6OTYBIKqUKiA+zXNl2koVsy7amcUhPMIT6/fo
5342 (M087) hH+6MZDA4t6jomcyhlCiW8d9IAzSWFkfg2k=
5343 (M088) </e:CipherValue>
5344 (M089) </e:CipherData>
5345 (M090) </e:EncryptedKey>
5346 (M091) </KeyInfo>
5347 (M092) <e:CipherData>
5348 (M093) <e:CipherValue>
5349 (M094) <!-- base64 encoded octets from SAML assertion encrypted
5350 (M095) with the encrypted key K2 above -->
5351 (M096) <!-- SAML assertion element is identical as received in
5352 (M097) RSTR , unencrypted form is omitted for brevity -->
5353 (M098) <!--.....-->
5354 (M099) </e:CipherValue>
5355 (M0100) </e:CipherData>
5356 (M0101) </e:EncryptedData>
5357 (M0102)
5358 (M0103) <sc:DerivedKeyToken u:Id="_9">
5359 (M0104) <o:SecurityTokenReference>
5360 (M0105) <o:KeyIdentifier ValueType=
5361 (M0106) "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5362 1.0#SAMLAssertionID">
5363 (M0107) <b>uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16</b>
5364 (M0108) </o:KeyIdentifier>
5365 (M0109) </o:SecurityTokenReference>
5366 (M0110) <sc:Offset>0</sc:Offset>
5367 (M0111) <sc:Length>24</sc:Length>
5368 (M0112) <sc:Nonce>pgnS/VDSzJn6SFz+Vy23JA==</sc:Nonce>
5369 (M0113) </sc:DerivedKeyToken>
5370 (M0114) <Signature Id="_1" xmlns="http://www.w3.org/2000/09/xmldsig#">
5371 (M0115) <SignedInfo>
5372 (M0116) <CanonicalizationMethod Algorithm=
5373 (M0117) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5374 (M0118) <SignatureMethod Algorithm=
5375 (M0119) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5376 (M0120) <Reference URI="#_3">
5377 (M0121) <Transforms>
5378 (M0122) <Transform Algorithm=
5379 (M0123) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5380 (M0124) </Transforms>
5381 (M0125) <DigestMethod Algorithm=
5382 (M0126) "http://www.w3.org/2000/09/xmldsig#sha1" />
5383 (M0127) <DigestValue>eQdQVGRkVI1YfKJBw7vOYCOeLQw=</DigestValue>
5384 (M0128) </Reference>
5385 (M0129) <Reference URI="#_5">
5386 (M0130) <Transforms>
5387 (M0131) <Transform Algorithm=
5388 (M0132) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5389 (M0133) </Transforms>
5390 (M0134) <DigestMethod Algorithm=
5391 (M0135) "http://www.w3.org/2000/09/xmldsig#sha1" />
5392 (M0136) <DigestValue>xxyKpp5RZ2TebKca2IGOafIgcxk=</DigestValue>
5393 (M0137) </Reference>
5394 (M0138) <Reference URI="#_6">
5395 (M0139) <Transforms>
5396 (M0140) <Transform Algorithm=
5397 (M0141) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5398 (M0142) </Transforms>
5399 (M0143) <DigestMethod Algorithm=
5400 (M0144) "http://www.w3.org/2000/09/xmldsig#sha1" />

```

```

5401 (M0145) <DigestValue>WyGDDyYbL/hQZJfE3Yx2aK3RkK8=</DigestValue>
5402 (M0146) </Reference>
5403 (M0147) <Reference URI="#_7">
5404 (M0148) <Transforms>
5405 (M0149) <Transform Algorithm=
5406 (M0150) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5407 (M0151) </Transforms>
5408 (M0152) <DigestMethod Algorithm=
5409 (M0153) "http://www.w3.org/2000/09/xmldsig#sha1"/>
5410 (M0154) <DigestValue>AEOH0t2KYR8mivgqUGDrgMtxgEQ=</DigestValue>
5411 (M0155) </Reference>
5412 (M0156) <Reference URI="#_8">
5413 (M0157) <Transforms>
5414 (M0158) <Transform Algorithm=
5415 (M0159) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5416 (M0160) </Transforms>
5417 (M0161) <DigestMethod Algorithm=
5418 (M0162) "http://www.w3.org/2000/09/xmldsig#sha1"/>
5419 (M0163) <DigestValue>y8n6Dxd3DbD6TR6d6H/oVWsV4yE=</DigestValue>
5420 (M0164) </Reference>
5421 (M0165) <Reference
5422 (M0166) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5423 (M0167) <Transforms>
5424 (M0168) <Transform Algorithm=
5425 (M0169) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5426 (M0170) </Transforms>
5427 (M0171) <DigestMethod Algorithm=
5428 (M0172) "http://www.w3.org/2000/09/xmldsig#sha1"/>
5429 (M0173) <DigestValue>/Cc+bGkkeQ6j1VXZx8PGgmF6MjI=</DigestValue>
5430 (M0174) </Reference>
5431 (M0175) </SignedInfo>
5432 (M0176) <!--base64 encoded signature value -->
5433 (M0177) <SignatureValue>EyKUHUuffPUPE/ZjaFrMJJ5KLKY=</SignatureValue>
5434 (M0178) <KeyInfo>
5435 (M0179) <o:SecurityTokenReference>
5436 (M0180) <o:Reference URI="#_0"/>
5437 (M0181) </o:SecurityTokenReference>
5438 (M0182) </KeyInfo>
5439 (M0183) </Signature>
5440 (M0184) <!-- signature over the primary signature above
5441 (M0185) using the key derived from the proof-key, K3,
5442 (M0186) associated with SAML assertion -->
5443 (M0187) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5444 (M0188) <SignedInfo>
5445 (M0189) <CanonicalizationMethod Algorithm=
5446 (M0190) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5447 (M0191) <SignatureMethod Algorithm=
5448 (M0192) "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
5449 (M0193) <Reference URI="#_1">
5450 (M0194) <Transforms>
5451 (M0195) <Transform Algorithm=
5452 (M0196) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
5453 (M0197) </Transforms>
5454 (M0198) <DigestMethod Algorithm=
5455 (M0199) "http://www.w3.org/2000/09/xmldsig#sha1"/>
5456 (M0200) <DigestValue>TMSmLlgeUn8cxyb60Ye5Q2nUuxY=</DigestValue>
5457 (M0201) </Reference>
5458 (M0202) </SignedInfo>
5459 (M0203) <SignatureValue>Fh4NyOpAi+NqVFiHBgHwyvzah9I=</SignatureValue>
5460 (M0204) <KeyInfo>
5461 (M0205) <o:SecurityTokenReference>
5462 (M0206) <o:Reference URI="#_9"/>
5463 (M0207) </o:SecurityTokenReference>
5464 (M0208) </KeyInfo>

```

```

5465 (M0209)         </Signature>
5466 (M0210)         </o:Security>
5467 (M0211)         </s:Header>
5468 (M0212)         <s:Body u:Id="_3">
5469 (M0213)         <e:EncryptedData Id="_4"
5470 (M0214)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5471 (M0215)         <e:EncryptionMethod Algorithm=
5472 (M0216)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5473 (M0217)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5474 (M0218)         <o:SecurityTokenReference >
5475 (M0219)         <o:Reference URI="#_2"/>
5476 (M0220)         </o:SecurityTokenReference>
5477 (M0221)         </KeyInfo>
5478 (M0222)         <e:CipherData>
5479 (M0223)         <e:CipherValue>
5480 (M0224)         <!-- base64 encoded octets of encrypted body content-->
5481 (M0225)         </e:CipherValue>
5482 (M0226)         </e:CipherData>
5483 (M0227)         </e:EncryptedData>
5484 (M0228)         </s:Body>
5485 (M0229)         </s:Envelope>

```

5486 The message above is a request from the Client to the Service using the tokens provided by the STS  
5487 above.

5488 Lines (M022)-(M0210) contain the **WS-Security o:Security header** for this request.

5489 Lines (M028)-(M045) contain an **e:EncryptedKey** that contains the **ephemeral key, K2**, for the Client-  
5490 Service communication. The service must use its X509T3 (M034)-(M037) to decrypt this Client-generated  
5491 ephemeral key, K2.

5492 Lines (M046)-(M054) contain a **WS-SecureConversation sc:DerivedKeyToken**, that contains the  
5493 information required to **generate the signing key, DKT1(K2)** [**WS\_SECURE\_CONV**], including an  
5494 sc:Nonce (M053), and a WS-Security SecurityTokenReference that contains a direct reference (M049) to  
5495 the e:EncryptedKey, K2 (M042). This derived key, DKT1(K2), is used to sign the message in the  
5496 message signature element (M0114)-(M0183).

5497 Lines (M055)-(M054) provide similar **sc:DerivedKeyToken** constructs the **generate the encrypting key**  
5498 **DKT2(K2)**. This derived key, DKT2(K2), is used to encrypt the message body, resulting in the  
5499 EncryptedData element in the s:Body, on lines (M0213)-(M0227).

5500 Lines (M062)-(M064) provide an **e:ReferenceList to reference the e:EncryptedData in the s:Body**  
5501 (M0213), which contains the Client request for the Service. The s:Body payload data in this Client request  
5502 is not of interest to the security properties of this example and will not be shown in decrypted form.

5503 Lines (M066)-(M0101) contain an **e:EncryptedData element that contains the saml:Assertion**  
5504 described in the STS response above. This e:EncryptedData contains a dsig KeyInfo, which, in turn,  
5505 contains an e:EncryptedKey element (M072)-(M090), which contains the Client-generated ephemeral  
5506 key, K2, which was used by the Client to directly encrypt the saml:Assertion. The encryption key, K2, may  
5507 be decrypted, again using the Service public X509 certificate, again identified by its ThumbprintSHA1,  
5508 (M077)-(M080).

5509 Lines (M0103)-(M0113) contain a **third sc:DerivedKeyToken** that contains the information necessary for  
5510 the Service to **generate the endorsing signing key, DKT3(K2)**.

5511 Lines (M0114)-(M0183) contains the **message signature**. It is **signed using** the key identified in the dsig  
5512 KeyInfo (M0178)-(M0182), which contains a reference to the **derived signing key, DKT1(K2)**, which may  
5513 be constructed by the service using the sc:DerivedKeyToken (M046) described above.

5514 Lines (M0187)-(M0209) contain the **message endorsing signature**. It is **signed using the client proof**  
5515 **key, K3**, that was generated by the STS. Note that the Service should compare this key, K3, with the one  
5516 in the saml:SubjectConfirmation in the decrypted saml:Assertion to verify that the Client is using the same  
5517 proof key, K3, that is contained in the saml:Assertion that authenticates this request.

5518 Lines (M0213)-(M0227) contain the **SOAP s:Body message payload, e:EncryptedData**, which may be  
5519 decrypted using the sc:DerivedKeyToken (M055) to generate the decryption key DKT2(K2).

5520

5521 Here is an example response from the Service to the Client:

5522

```
5523 (R001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5524 (R002)   xmlns:a=http://www.w3.org/2005/08/addressing
5525 (R003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5526 (R004)   xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5527         1.1.xsd"
5528 (R005)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5529         wssecurity-secext-1.0.xsd"
5530 (R006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5531         secureconversation/200512"
5532 (R007)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5533         wssecurity-utility-1.0.xsd">
5534 (R008)   <s:Header>
5535 (R009)     <a:Action s:mustUnderstand="1" u:Id="_6">
5536 (R010)       http://example.org/PingResponse
5537 (R011)     </a:Action>
5538 (R012)     <a:RelatesTo u:Id="_7">
5539 (R013)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5540 (R014)     </a:RelatesTo>
5541 (R015)     <a:To s:mustUnderstand="1" u:Id="_8">
5542 (R016)       http://www.w3.org/2005/08/addressing/anonymous
5543 (R017)     </a:To>
5544 (R018)     <o:Security s:mustUnderstand="1">
5545 (R019)       <u:Timestamp u:Id="uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5546 (R020)         <u:Created>2005-10-25T00:47:38.921Z</u:Created>
5547 (R021)         <u:Expires>2005-10-25T00:52:38.921Z</u:Expires>
5548 (R022)       </u:Timestamp>
5549 (R023)       <sc:DerivedKeyToken u:Id="_0" >
5550 (R024)         <o:SecurityTokenReference
5551 (R025)           k:TokenType="http://docs.oasis-open.org/wss/
5552 (R026)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5553 (R027)           xmlns:k="http://docs.oasis-open.org/wss/
5554 (R028)             oasis-wss-wssecurity-secext-1.1.xsd">
5555 (R029)             <o:KeyIdentifier ValueType=
5556 (R030)             "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5557             1.1.xsd#EncryptedKeySHA1"
5558 (R031)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5559 (R032)           </o:SecurityTokenReference>
5560 (R033)           <sc:Offset>0</sc:Offset>
5561 (R034)           <sc:Length>24</sc:Length>
5562 (R035)           <sc:Nonce>KFjylGb73BubLul0ZGgx+w==</sc:Nonce>
5563 (R036)         </sc:DerivedKeyToken>
5564 (R037)       <sc:DerivedKeyToken u:Id="_3">
5565 (R038)         <o:SecurityTokenReference
5566 (R039)           k:TokenType="http://docs.oasis-open.org/wss/
5567 (R040)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5568 (R041)           xmlns:k="http://docs.oasis-open.org/wss/
5569 (R042)             oasis-wss-wssecurity-secext-1.1.xsd">
5570 (R043)             <o:KeyIdentifier ValueType=
5571 (R044)             "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5572             1.1.xsd#EncryptedKeySHA1"
5573 (R045)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5574 (R046)           </o:SecurityTokenReference>
5575 (R047)           <sc:Nonce>omyh+Eg6XIa8q3V5IkHiXg==</sc:Nonce>
5576 (R048)         </sc:DerivedKeyToken>
5577 (R049)       <e:ReferenceList>
5578 (R050)         <e:DataReference URI="#_5"/>
5579 (R051)       </e:ReferenceList>
5580 (R052)       <k:SignatureConfirmation u:Id="_1"
5581 (R053)         Value="EyKUHUuffPUPE/ZjaFrMJJ5KLKY="/>
5582 (R054)       <k:SignatureConfirmation u:Id="_2"
```

```

5583 (R055) Value="Fh4NyOpAi+NqVFiHBgHWyvzah9I="/>
5584 (R056) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5585 (R057) <SignedInfo>
5586 (R058) <CanonicalizationMethod Algorithm=
5587 (R059) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5588 (R060) <SignatureMethod Algorithm=
5589 (R061) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5590 (R062) <Reference URI="#_4">
5591 (R063) <Transforms>
5592 (R064) <Transform Algorithm=
5593 (R065) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5594 (R066) </Transforms>
5595 (R067) <DigestMethod Algorithm=
5596 (R068) "http://www.w3.org/2000/09/xmldsig#sha1" />
5597 (R069) <DigestValue>y/oItF5TcTOFan7SavhZTTTv48M=</DigestValue>
5598 (R070) </Reference>
5599 (R071) <Reference URI="#_6">
5600 (R072) <Transforms>
5601 (R073) <Transform Algorithm=
5602 (R074) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5603 (R075) </Transforms>
5604 (R076) <DigestMethod Algorithm=
5605 (R077) "http://www.w3.org/2000/09/xmldsig#sha1" />
5606 (R078) <DigestValue>X4UIaLWnaAWTriw4UJ/SFDgm090=</DigestValue>
5607 (R079) </Reference>
5608 (R080) <Reference URI="#_7">
5609 (R081) <Transforms>
5610 (R082) <Transform Algorithm=
5611 (R083) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5612 (R084) </Transforms>
5613 (R085) <DigestMethod Algorithm=
5614 (R086) "http://www.w3.org/2000/09/xmldsig#sha1" />
5615 (R087) <DigestValue>vqy8/D4CDCaI1nnd4wl1Qjyp+qM=</DigestValue>
5616 (R088) </Reference>
5617 (R089) <Reference URI="#_8">
5618 (R090) <Transforms>
5619 (R091) <Transform Algorithm=
5620 (R092) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5621 (R093) </Transforms>
5622 (R094) <DigestMethod Algorithm=
5623 (R095) "http://www.w3.org/2000/09/xmldsig#sha1" />
5624 (R096) <DigestValue>H1lvLAr5g8pbZ6jfZ+2WNYiNjiM=</DigestValue>
5625 (R097) </Reference>
5626 (R098) <Reference
5627 (R099) URI="#uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5628 (R100) <Transforms>
5629 (R101) <Transform Algorithm=
5630 (R102) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5631 (R103) </Transforms>
5632 (R104) <DigestMethod Algorithm=
5633 (R105) "http://www.w3.org/2000/09/xmldsig#sha1" />
5634 (R106) <DigestValue>dr0g6hycoc884i+BD8FYCJGbbbE=</DigestValue>
5635 (R107) </Reference>
5636 (R108) <Reference URI="#_1">
5637 (R109) <Transforms>
5638 (R110) <Transform Algorithm=
5639 (R111) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5640 (R112) </Transforms>
5641 (R113) <DigestMethod Algorithm=
5642 (R114) "http://www.w3.org/2000/09/xmldsig#sha1" />
5643 (R115) <DigestValue>Rv3N7VNfAqpn0khr3F/qQZmE/l4=</DigestValue>
5644 (R116) </Reference>
5645 (R117) <Reference URI="#_2">
5646 (R118) <Transforms>

```

```

5647 (R0119)         <Transform Algorithm=
5648 (R0120)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
5649 (R0121)         </Transforms>
5650 (R0122)         <DigestMethod Algorithm=
5651 (R0123)         "http://www.w3.org/2000/09/xmldsig#sha1" />
5652 (R0124)         <DigestValue>X2pxEnYPM8cMLrbhNqPgs8xk+a4=</DigestValue>
5653 (R0125)         </Reference>
5654 (R0126)         </SignedInfo>
5655 (R0127)         <SignatureValue>I2jQuDTWWQiNJy/ziyg8AFYO/z4=</SignatureValue>
5656 (R0128)         <KeyInfo>
5657 (R0129)         <o:SecurityTokenReference>
5658 (R0130)         <o:Reference URI="#_0" />
5659 (R0131)         </o:SecurityTokenReference>
5660 (R0132)         </KeyInfo>
5661 (R0133)         </Signature>
5662 (R0134)         </o:Security>
5663 (R0135)         </s:Header>
5664 (R0136)         <s:Body u:Id="_4">
5665 (R0137)         <e:EncryptedData Id="_5"
5666 (R0138)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5667 (R0139)         <e:EncryptionMethod Algorithm=
5668 (R0140)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
5669 (R0141)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5670 (R0142)         <o:SecurityTokenReference>
5671 (R0143)         <o:Reference URI="#_3" />
5672 (R0144)         </o:SecurityTokenReference>
5673 (R0145)         </KeyInfo>
5674 (R0146)         <e:CipherData>
5675 (R0147)         <e:CipherValue>
5676 (R0148)         <!--base64 encoded octets of encrypted body content-->
5677 (R0149)         </e:CipherValue>
5678 (R0150)         </e:CipherData>
5679 (R0151)         </e:EncryptedData>
5680 (R0152)         </s:Body>
5681 (R0153)         </s:Envelope>

```

5682 The message above is a response from the Service to the Client using the tokens based on the  
5683 saml:Assertion IssuedToken from the STS.

5684 Lines (R018)-(R0134) contain the WS-Security o:Security header in the SOAP s:Header.

5685 Lines (R023)-(R036) contain an **sc:DerivedKeyToken** that may be used to **construct the derived**  
5686 **signing key DKT4(K2)**, which uses the Client-generated ephemeral key, K2, that the Service received in  
5687 the Client request (M028)-(M045) above, and is now used in the response to the client, similar to the way  
5688 the STS used K1 to respond to the Client, except that in this case the Service will use derived keys  
5689 DKT4(K2) and DKT5(K2) in addition to K2 in the response. This sc:DerivedKeyToken contains a WS-  
5690 Security o:SecurityTokenReference (R024)-(R032) that uses the WS-Security 1.1 mechanism  
5691 EncryptedKeySHA1 to identify the Client-generated ephemeral key, K2, as the key to use to derive  
5692 DKT4(K2) along with the sc:Nonce provided (R035) (and the label as described in  
5693 [\[WS\\_SECURE\\_CONV\]](#)).

5694 Lines (R037)-(R048) contain a **second sc:DerivedKeyToken** that may be used by the Client to  
5695 **construct the derived encryption key, DKT5(K2)**. The same EncryptedKeySHA1 mechanism is used  
5696 by the Client to construct DKT5(K2) as described for DKT4(K2).

5697 Lines (R049)-(R051) contain an **e:ReferenceList** containing an e:DataReference to the EncryptedData  
5698 element in the s:Body (R0136).

5699 Lines (R052)-(R055) contain 2 WS-Security 1.1 k:SignatureConfirmation elements confirming the dsig  
5700 Signatures that were in the client request above (M0203) on the endorsing signature and (M0177) on the  
5701 message signature of the Client request. The dsig SignatureValues compare respectively to assure the  
5702 Client that the data from the Client request was processed and is what is being referred to in this  
5703 response.

5704 Lines (R056)-(R0134) contain the message signature, which is signed as referenced in the dsig KeyInfo  
5705 (R0128)-(R0132) by DKT4(K2), which may be constructed as described above using the  
5706 sc:DerivedKeyToken element (R023)-(R036).

5707 Lines (R0137)-(R0151) contain the Service response payload to the Client which may be decrypted using  
5708 DKT5(K2) using the sc:DerivedKeyToken element (R037)-(R048).

## 5709 2.4 Secure Conversation Scenarios

### 5710 2.4.1 (WSS 1.0) Secure Conversation bootstrapped by Mutual 5711 Authentication with X.509 Certificates

5712 This scenario corresponds to the situation where both parties have an X.509 certificate (and public/private  
5713 key pair). Because of the volume of messages from each source, the Requestor/Initiator uses WS-  
5714 SecureConversation to establish a new session key and uses the session key for integrity and/or  
5715 confidentiality protection. This improves performance, by using less expensive symmetric key operations  
5716 and improves security by reducing the exposure of the long term secret.

5717 The recipient models this scenario by using the symmetric binding that includes a  
5718 SecureConversationToken assertion to describe the token type accepted by this endpoint. This token  
5719 assertion further contains a bootstrap policy to indicate the security binding that is used by requestors that  
5720 want a security context token issued by this service. The bootstrap policy affects the Request Security  
5721 Token Request (RST) and Request Security Token Request Response (RSTR) messages sent between  
5722 the Initiator and the Recipient to establish the security context. It is modeled by use of an asymmetric  
5723 binding assertion for this scenario because both parties mutually authenticate each other using their  
5724 X.509 certificates.

5725 The message level policies cover a different scope of the web service definition than the security binding  
5726 level policy and so appear as separate policies and are attached at Message Policy Subject. These are  
5727 shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint  
5728 subject (WSS10SecureConversation\_policy) and two (WSS10SecureConversation\_input\_policy,  
5729 WSS10SecureConversation\_output\_policy) with message subjects to achieve this use case.

5730

```
5731 (P001) <wsp:Policy wsu:Id="WSS10SecureConversation_policy">  
5732 (P002)   <wsp:ExactlyOne>  
5733 (P003)     <wsp:All>  
5734 (P004)       <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-  
5735 securitypolicy/200702">  
5736 (P005)         <wsp:Policy>  
5737 (P006)           <sp:ProtectionToken>  
5738 (P007)             <wsp:Policy>  
5739 (P008)               <sp:SecureConversationToken  
5740 sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken  
5741 /AlwaysToRecipient">  
5742 (P009)                 <wsp:Policy>  
5743 (P010)                   <sp:RequireDerivedKeys/>  
5744 (P011)                   <sp:BootstrapPolicy>  
5745 (P012)                   <wsp:Policy>  
5746 (P013)                   <wsp:ExactlyOne>  
5747 (P014)                   <wsp:All>  
5748 (P015)                   <sp:AsymmetricBinding  
5749 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">  
5750 (P016)                   <wsp:Policy>  
5751 (P017)                   <sp:InitiatorToken>  
5752 (P018)                   <wsp:Policy>  
5753 (P019)                   <sp:X509Token  
5754 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/  
5755 AlwaysToRecipient">  
5756 (P020)                   <wsp:Policy>  
5757 (P021)                   <sp:WssX509V3Token10/>  
5758 (P022)                   </wsp:Policy>
```

```

5759         </sp:X509Token>
5760     </wsp:Policy>
5761 </sp:InitiatorToken>
5762 <sp:RecipientToken>
5763     <wsp:Policy>
5764         <sp:X509Token
5765 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
5766 AlwaysToInitiator">
5767             <wsp:Policy>
5768                 <sp:WssX509V3Token10/>
5769             </wsp:Policy>
5770         </sp:X509Token>
5771     </wsp:Policy>
5772 </sp:RecipientToken>
5773 <sp:AlgorithmSuite>
5774     <wsp:Policy>
5775         <sp:TripleDesRsa15/>
5776     </wsp:Policy>
5777 </sp:AlgorithmSuite>
5778 <sp:Layout>
5779     <wsp:Policy>
5780         <sp:Strict/>
5781     </wsp:Policy>
5782 </sp:Layout>
5783 <sp:IncludeTimestamp/>
5784 <sp:OnlySignEntireHeadersAndBody/>
5785 </wsp:Policy>
5786 </sp:AsymmetricBinding>
5787 <sp:Wss10>
5788     <wsp:Policy>
5789         <sp:MustSupportRefKeyIdentifier/>
5790     </wsp:Policy>
5791 </sp:Wss10>
5792 <sp:SignedParts>
5793 <sp:Body/>
5794 <sp:Header Name="Action"
5795     Namespace="http://www.w3.org/2005/08/addressing"/>
5796 </sp:SignedParts>
5797 <sp:EncryptedParts>
5798 <sp:Body/>
5799 </sp:EncryptedParts>
5800 </wsp:All>
5801 </wsp:ExactlyOne>
5802 </wsp:Policy>
5803 </sp:BootstrapPolicy>
5804 </wsp:Policy>
5805 </sp:SecureConversationToken>
5806 </wsp:Policy>
5807 </sp:ProtectionToken>
5808 <sp:AlgorithmSuite>
5809     <wsp:Policy>
5810         <sp:Basic256/>
5811     </wsp:Policy>
5812 </sp:AlgorithmSuite>
5813 <sp:Layout>
5814     <wsp:Policy>
5815         <sp:Strict/>
5816     </wsp:Policy>
5817 </sp:Layout>
5818 <sp:IncludeTimestamp/>
5819 <sp:OnlySignEntireHeadersAndBody/>
5820 </wsp:Policy>
5821 </sp:SymmetricBinding>
5822 <sp:Trust13>

```

```

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

```

5866 Lines (P004) – (P082) indicate that the service uses the symmetric security binding to protect messages,  
5867 using a Security Context Token (Lines (P008) – (P066)) to sign messages in both directions. The actual  
5868 basis for the signatures should be keys derived from that Security Context Token as stated by the  
5869 RequireDerivedKey assertion in Line (P010). Messages must include a Timestamp element (Line (P079))  
5870 and the signature value must be calculated over the entire body and header elements (Line (P080)).

5871 Lines (P083) – (P088) contain the Trust13 assertion which defines the requirements for the WS-Trust  
5872 related message exchange as part of the SC bootstrap. Line (P085) indicates that the Initiator (Client) has  
5873 to provide entropy to be used as key material for the requested proof token. Line (P086) requires the  
5874 same from the recipient (Server) which results in computed key from both client and server entropy  
5875 values.

5876 Lines (P011) – (P065) contain the bootstrap policy for the initial creation of the security context between  
5877 the communication parties before it is being used. It contains an AsymmetricBinding assertion (Lines  
5878 (P015) – (P048)) which indicates that the initiator's (WS-Security 1.0 X.509 Certificate) token (Lines  
5879 (P017) – (P025)) used by the recipient to verify the signature in the RST must be included in the message  
5880 (P019). The exchange of entropy and key material in the body of the RST and RSTR messages is  
5881 protected by the EncryptedParts assertion of the bootstrap policy in lines (P058) – (P061).

5882 According to the MustSupportKeyRefIdentifier assertions in Line (P051), an X.509 Key Identifier must be  
5883 used to identify the token. Lines (P054) – (P057) require that the body and the WS-Addressing Action  
5884 header is signed on each message (RST, RSTR) within the bootstrap process.

5885 An example of an RST message sent from the initiator to the recipient according to the bootstrap policy  
5886 defined by this policy is as follows:

```
5887 (M001) <?xml version="1.0" encoding="utf-8" ?>
5888 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
5889         xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="...">
5890 (M003)   <soap:Header>
5891 (M004)     <wsa:Action wsu:Id="action">
5892 (M005)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
5893 (M006)     </wsa:Action>
5894 (M007)     <wsse:Security>
5895 (M008)       <wsu:Timestamp wsu:Id="timestamp">
5896 (M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5897 (M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5898 (M011)       </wsu:Timestamp>
5899 (M012)       <wsse:BinarySecurityToken wsu:Id="clientToken"
5900             Value="..."#X509v3" EncodingType="..."#Base64Binary">
5901 (M013)         MIICZDCCAc2gAwIBAgIRALSOLzt7...
5902 (M014)       </wsse:BinarySecurityToken>
5903 (M015)       <ds:Signature xmlns:ds="...">
5904 (M016)         <ds:SignedInfo>
5905 (M017)           <ds:CanonicalizationMethod
5906             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5907 (M018)           <ds:SignatureMethod
5908             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
5909 (M019)           <ds:Reference URI="#action">
5910 (M020)             <ds:Transforms>
5911 (M021)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
5912                   exc-c14n#" />
5913 (M022)             </ds:Transforms>
5914 (M023)             <ds:DigestMethod
5915               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
5916 (M024)             <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
5917 (M025)           </ds:Reference>
5918 (M026)           <ds:Reference URI="#timestamp">...</ds:Reference>
5919 (M027)           <ds:Reference URI="#body">...</ds:Reference>
5920 (M028)         </ds:SignedInfo>
5921 (M029)         <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
5922 (M030)         <ds:KeyInfo>
5923 (M031)           <wsse:SecurityTokenReference>
5924 (M032)             <wsse:Reference URI="#clientToken"
5925                   Value="..."#X509v3" />
5926 (M033)           </wsse:SecurityTokenReference>
5927 (M034)         </ds:KeyInfo>
5928 (M035)       </ds:Signature>
5929 (M036)     <xenc:EncryptedKey>
5930 (M037)       <xenc:EncryptionMethod Algorithm="..."#rsa-1_5" />
5931 (M038)       <ds:KeyInfo>
5932 (M039)         <wsse:SecurityTokenReference>
5933 (M040)           <wsse:KeyIdentifier
5934 (M041)             Value="..."#X509v3SubjectKeyIdentifier">AtETQ...
5935 (M042)           </wsse:KeyIdentifier>
5936 (M043)         </wsse:SecurityTokenReference>
5937 (M044)       </ds:KeyInfo>
5938 (M045)       <xenc:CipherData>
5939 (M046)         <xenc:CipherValue>
5940 (M047)           <!-- encrypted key -->
5941 (M048)         </xenc:CipherValue>
5942 (M049)       </xenc:CipherData>
5943 (M050)       <xenc:ReferenceList>
5944 (M051)         <xenc:DataReference URI="#request" />
```

```

5945 (M052) </xenc:ReferenceList>
5946 (M053) </xenc:EncryptedKey>
5947 (M054) </wsse:Security>
5948 (M055) </soap:Header>
5949 (M056) <soap:Body wsu:Id="body">
5950 (M057) <xenc:EncryptedData Id="request" Type="...#Content">
5951 (M058) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
5952 (M059) <xenc:CipherData>
5953 (M060) <xenc:CipherValue>
5954 (M061) <!-- encrypted RST request -->
5955 (M062) <!-- ### Begin unencrypted RST request
5956 (M063) <wst:RequestSecurityToken>
5957 (M064) <wst:TokenType>
5958 (M065) http://docs.oasis-open.org/ws-sx/ws-secureconversation/
5959 200512/sct
5960 (M066) </wst:TokenType>
5961 (M067) <wst:RequestType>
5962 (M068) http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
5963 (M069) </wst:RequestType>
5964 (M070) <wsp:AppliesTo
5965 xmlns:wsp="http://www.w3.org/ns/ws-policy">
5966 (M071) <wsp:EndpointReference>
5967 (M072) <wsp:Address>
5968 (M073) http://acme.com/ping/
5969 (M074) </wsp:Address>
5970 (M075) </wsp:EndpointReference>
5971 (M076) </wsp:AppliesTo>
5972 (M077) <wst:Entropy>
5973 (M078) <wst:BinarySecret>cr9b0cb1wCEyY9ehaGK33e41h9s=
5974 (M079) </wst:BinarySecret>
5975 (M080) </wst:Entropy>
5976 (M081) </wst:RequestSecurityToken>
5977 (M082) ### End unencrypted RST request -->
5978 (M083) </xenc:CipherValue>
5979 (M084) </xenc:CipherData>
5980 (M085) </xenc:EncryptedData>
5981 (M086) </soap:Body>
5982 (M087) </soap:Envelope>

```

- 5983 Line (M005) indicates to the recipient that the SCT Binding of WS-Trust is used.
- 5984 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 5985 Lines (M012) – (M014) contain the initiator’s X.509 token as required by the InitiatorToken assertion’s value (“.../AlwaysToRecipient”).
- 5986
- 5987 Lines (M015) – (M035) hold the message signature.
- 5988 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).
- 5989
- 5990
- 5991
- 5992 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 5993
- 5994 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.
- 5995
- 5996 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.
- 5997
- 5998 Lines (M056) – (M086) hold the Security Token Reference pointing to the initiators X.509 certificate.
- 5999 Lines (M038) – (M058) contain the SOAP Body of the message with the encrypted RST element. Lines 6000 (M062) – (M082) show the unencrypted content of the RST request. It specifies the Token Type (Lines 6001 (M064) – (M066)) and the Request Type (Lines (M067) – (M069)). According to the Trust13 assertion, it 6002 also includes entropy provided by the initiator as indicated by Lines (M077) – (M080).

6003 An example of an RSTR message sent from the recipient to the initiator according to the bootstrap policy  
 6004 defined by this policy is as follows:

```

6005 (M001) <?xml version="1.0" encoding="UTF-8"?>
6006 (M002) <soap:Envelope xmlns:soap="..." xmlns:wst="..." xmlns:wsc="..."
6007 (M003)   xmlns:xenc="...">
6008 (M004)   <soap:Header>
6009 (M005)     <wsa:Action wsu:Id="action">
6010 (M006)       http://docs.oasis-open.org/ws-sx/ws-
6011 trust/200512/RSTRC/IssueFinal
6012 (M007)     </wsa:Action>
6013 (M008)     <wsse:Security>
6014 (M009)       <wsu:Timestamp wsu:Id="timestamp">
6015 (M010)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6016 (M011)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6017 (M012)       </wsu:Timestamp>
6018 (M013)       <wsse:BinarySecurityToken wsu:Id="serviceToken"
6019 (M014)         ValueType="...#X509v3" EncodingType="...#Base64Binary">
6020 (M015)         MIIASDPIOASDsaöAgIRALSOLzt7...
6021 (M016)       </wsse:BinarySecurityToken>
6022 (M017)       <ds:Signature xmlns:ds="...">
6023 (M018)         <ds:SignedInfo>
6024 (M019)           <ds:CanonicalizationMethod
6025 (M020)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6026 (M021)           <ds:SignatureMethod
6027 (M022)             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6028 (M023)           <ds:Reference URI="#action">
6029 (M024)             <ds:Transforms>
6030 (M025)               <ds:Transform
6031 (M026)                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6032 (M027)             </ds:Transforms>
6033 (M028)           <ds:DigestMethod
6034 (M029)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6035 (M030)           <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6036 (M031)         </ds:Reference>
6037 (M032)         <ds:Reference URI="#timestamp">...</ds:Reference>
6038 (M033)         <ds:Reference URI="#body">...</ds:Reference>
6039 (M034)       </ds:SignedInfo>
6040 (M035)       <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6041 (M036)       <ds:KeyInfo>
6042 (M037)         <wsse:SecurityTokenReference>
6043 (M038)           <wsse:Reference URI="#myToken" ValueType="...#X509v3"/>
6044 (M039)         </wsse:SecurityTokenReference>
6045 (M040)       </ds:KeyInfo>
6046 (M041)     </ds:Signature>
6047 (M042)     <xenc:EncryptedKey>
6048 (M043)       <xenc:EncryptionMethod Algorithm="...#rsa-1_5"/>
6049 (M044)       <ds:KeyInfo>
6050 (M045)         <wsse:SecurityTokenReference>
6051 (M046)           <wsse:KeyIdentifier
6052 (M047)             ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
6053 (M048)           </wsse:KeyIdentifier>
6054 (M049)         </wsse:SecurityTokenReference>
6055 (M050)       </ds:KeyInfo>
6056 (M051)       <xenc:CipherData>
6057 (M052)         <xenc:CipherValue>
6058 (M053)           <!-- encrypted key -->
6059 (M054)         </xenc:CipherValue>
6060 (M055)       </xenc:CipherData>
6061 (M056)       <xenc:ReferenceList>
6062 (M057)         <xenc:DataReference URI="#response"/>
6063 (M058)       </xenc:ReferenceList>
6064 (M059)     </xenc:EncryptedKey>
6065 (M060)   </wsse:Security>
6066 (M061) </soap:Header>

```

```

6067      (M056)      <soap:Body wsu:Id="body">
6068      (M057)      <xenc:EncryptedData Id="response" Type="...#Content">
6069      (M058)      <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
6070      (M059)      <xenc:CipherData>
6071      (M060)      <xenc:CipherValue>
6072      (M061)      <!-- encrypted RSTR -->
6073      (M062)      <!-- ### Begin unencrypted RSTR
6074      (M063)      <wst:RequestSecurityTokenResponseCollection>
6075      (M064)      <wst:RequestSecurityTokenResponse>
6076      (M065)      <wst:RequestedSecurityToken>
6077      (M066)      <wsc:SecurityContextToken>
6078      (M067)      <wsc:Identifier>uuid:...</wsc:Identifier>
6079      (M068)      </wsc:SecurityContextToken>
6080      (M069)      </wst:RequestedSecurityToken>
6081      (M070)      <wst:RequestedProofToken>
6082      (M071)      <xenc:EncryptedKey Id="ek049ea390c90011dbba4e00304852867e">
6083      (M072)      <xenc:EncryptionMethod
6084      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
6085      (M073)      <ds:KeyInfo>
6086      (M074)      <wsse:SecurityTokenReference>
6087      (M075)      <wsse:KeyIdentifier
6088      Value="...#X509SubjectKeyIdentifier"
6089      EncodingType="...#Base64Binary">
6090      (M076)      Wjw2gDCBye6NJAh0lPCyUldvTN8=
6091      (M077)      </wsse:KeyIdentifier>
6092      (M078)      </wsse:SecurityTokenReference>
6093      (M079)      </ds:KeyInfo>
6094      (M080)      <xenc:CipherData>
6095      (M081)      <xenc:CipherValue>hhx9TcaVL6XwBN1...</xenc:CipherValue>
6096      (M082)      </xenc:CipherData>
6097      (M083)      </xenc:EncryptedKey>
6098      (M084)      </wst:RequestedProofToken>
6099      (M085)      <wsp:AppliesTo
6100      xmlns:wsp="http://www.w3.org/ns/ws-policy">
6101      (M086)      <wsp:EndpointReference>
6102      (M087)      <wsp:Address>
6103      (M088)      http://acme.com/ping/
6104      (M089)      </wsp:Address>
6105      (M090)      </wsp:EndpointReference>
6106      (M091)      </wsp:AppliesTo>
6107      (M092)      <wst:Entropy>
6108      (M093)      <wst:BinarySecret>asdhjwkjqwe123498SDFALasd=
6109      (M094)      </wst:BinarySecret>
6110      (M095)      </wst:Entropy>
6111      (M096)      </wst:RequestSecurityTokenResponse>
6112      (M097)      </wst:RequestSecurityTokenResponseCollection>
6113      (M098)      ### End unencrypted RSTR -->
6114      (M099)      </xenc:CipherValue>
6115      (M100)      </xenc:CipherData>
6116      (M101)      </xenc:EncryptedData>
6117      (M102)      </soap:Body>
6118      (M103)      </soap:Envelope>

```

- 6119 Line (M005) indicates to the initiator that this is the final response to the RST in an RSTRC.
- 6120 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 6121 Lines (M012) – (M014) contain the recipient's X.509 token as required by the RecipientToken assertion's value (".../AlwaysToInitiator").
- 6122
- 6123 Lines (M015) – (M035) hold the message signature.
- 6124 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.
- 6125
- 6126 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 6127

6128 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp  
6129 assertion definition.

6130 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts  
6131 assertion.

6132 Lines (M031) – (M033) hold the Security Token Reference pointing to the requestor's X.509 certificate.

6133 Lines (M056) – (M102) contain the SOAP Body of the message with the encrypted RSTR collection  
6134 element. Commented out is the unencrypted form of the RSTR collection in Lines (M063) – (M097), which  
6135 includes one RSTR (Lines (M064) – (M096)). Lines (M065) – (M069) contain the new Security Context  
6136 Token. The accompanying RequestedProofToken in Lines (M070) – (M084) include the encrypted secret  
6137 key (Lines (M071) – (M083)) of the security context. The key is encrypted using the initiator's public key  
6138 that was already used to verify its signature in the incoming request.

6139 According to the Trust13 assertion, the response includes entropy provided by the recipient as indicated  
6140 by Lines (M092) – (M095).

6141 An Example of an SCT-secured application message sent from the initiator to the recipient according to  
6142 the message input policy (WSS10SecureConversation\_input\_policy) is as follows:

```

6143 (M001)    <?xml version="1.0" encoding="UTF-8"?>
6144 (M002)    <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
6145           xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="..." xmlns:wsm="...">
6146 (M003)    <soap:Header>
6147 (M004)    <wsa:To wsu:Id="to">http://acme.com/ping/</wsa:To>
6148 (M005)    <wsa:MessageID wsu:Id="msgid">
6149 (M006)    http://acme.com/guid/12318731edh-CA47-1067-B31D-10662DA
6150 (M007)    </wsa:MessageID>
6151 (M008)    <wsa:Action wsu:Id="action">urn:Ping</wsa:Action>
6152 (M009)    <wsse:Security>
6153 (M010)    <wsu:Timestamp wsu:Id="timestamp">
6154 (M011)    <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6155 (M012)    <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6156 (M013)    </wsu:Timestamp>
6157 (M014)    <wsc:SecurityContextToken wsu:Id="SCT">
6158 (M015)    <wsc:Identifier>uuid:91f50600-60cc-11da-8e67-000000000000
6159 (M016)    </wsc:Identifier>
6160 (M017)    </wsc:SecurityContextToken>
6161 (M018)    <wsc:DerivedKeyToken wsu:Id="DKT">
6162 (M019)    <wsse:SecurityTokenReference>
6163 (M020)    <wsse:Reference URI="#SCT" ValueType="http://docs.oasis-
6164           open.org/ws-sx/ws-secureconversation/200512/sct"/>
6165 (M021)    </wsse:SecurityTokenReference>
6166 (M022)    <wsc:Offset>0</wsc:Offset>
6167 (M023)    <wsc:Length>24</wsc:Length>
6168 (M024)    <wsc:Label>
6169           WSSecure ConversationWSSecure Conversation
6170 (M025)    </wsc:Label>
6171 (M026)    <wsc:Nonce>ylN04kFBJesy2U2SQL6ezI3SCak=</wsc:Nonce>
6172 (M027)    </wsc:DerivedKeyToken>
6173 (M028)    <ds:Signature xmlns:ds="...">
6174 (M029)    <ds:SignedInfo>
6175 (M030)    <ds:CanonicalizationMethod
6176           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6177 (M031)    <ds:SignatureMethod
6178           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6179 (M032)    <ds:Reference URI="#msgid">
6180 (M033)    <ds:Transforms>
6181 (M034)    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
6182           exc-c14n#"/>
6183 (M035)    </ds:Transforms>
6184 (M036)    <ds:DigestMethod
6185           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6186 (M037)    <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6187 (M038)    </ds:Reference>

```

```

6188 (M039) <ds:Reference URI="#to">...</ds:Reference>
6189 (M040) <ds:Reference URI="#action">...</ds:Reference>
6190 (M041) <ds:Reference URI="#timestamp">...</ds:Reference>
6191 (M042) <ds:Reference URI="#body">...</ds:Reference>
6192 (M043) </ds:SignedInfo>
6193 (M044) <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6194 (M045) <ds:KeyInfo>
6195 (M046) <wsse:SecurityTokenReference>
6196 (M047) <wsse:Reference URI="#DKT" ValueType="http://docs.oasis-
open.org/ws-sx/ws-secureconversation/200512/dk"/>
6197 (M048) </wsse:SecurityTokenReference>
6198 (M049) </ds:KeyInfo>
6199 (M050) </ds:Signature>
6200 (M051) </wsse:Security>
6201 (M052) </soap:Header>
6202 (M053) <soap:Body wsu:Id="body">
6203 (M054) <pns:Ping xmlns:pns="http://tempuri.org/">
6204 (M055) <pns:Text>abc</pns:Text>
6205 (M056) </pns:Ping>
6206 (M057) </soap:Body>
6207 (M058) </soap:Envelope>

```

- 6209
- 6210 Lines (M004) – (M008) contain the WS-Addressing headers according to the UsingAddressing assertion.
- 6211 Lines (M010) – (M013) hold the Timestamp as specified by the IncludeTimestamp assertion within the
- 6212 SymmetricBinding assertion.
- 6213 Lines (M014) – (M017) contain the Security Context Token which has been issued by the recipient in the
- 6214 previous message. Based on this token, the initiator included a Derived Key Token (Lines (M018) –
- 6215 (M027)) as indicated by the RequireDerivedKey assertion in the policy.
- 6216 Lines (M028) – (M050) hold the message signature that uses the Derived Key Token (Lines (M046) –
- 6217 (M048)) to sign the WS-Addressing headers (Lines (M032) – (M040)), the timestamp (Line (M041)) and
- 6218 the body (Line (M042)) of the message, according to the SignedParts assertion of the message input
- 6219 policy (WSS10SecureConversation\_input\_policy).

---

## 6220 **3 Conformance**

6221 This document contains non-normative examples of usage of WS-SecurityPolicy and other related  
6222 specifications.

6223 Therefore **there are no conformance statements that apply to this document.**

6224 Refer to the referenced specifications [[References](#)], which will individually contain conformance  
6225 requirements for WS-SecurityPolicy and related specifications.

6226

6227

---

## A. Acknowledgements

6228 The following individuals have participated in the creation of this specification and are gratefully  
6229 acknowledged:

6230 **Original Contributors:**

6231 Ashok Malhotra, Oracle  
6232 Prateek Mishra, Oracle  
6233 Ramana Turlapati, Oracle

6234 **Participants:**

6235 Don Adams, Tibco  
6236 Moushmi Banerjee, Oracle  
6237 Symon Chang, Oracle  
6238 Henry Chung, IBM  
6239 Paul Cotton, Microsoft  
6240 Marc Goodner, Microsoft  
6241 Martin Gudgin, Microsoft  
6242 Tony Gullotta, SOA  
6243 Jiandong Guo, Sun  
6244 Frederick Hirsch, Nokia  
6245 Chris Kaler, Microsoft  
6246 Rich Levinson, Oracle  
6247 Chunlong Liang, IBM  
6248 Hal Lockhart, Oracle  
6249 Mike Lyons, Layer 7  
6250 Ashok Malhotra, Oracle  
6251 Carlo Milono, Tibco  
6252 Prateek Mishra, Oracle  
6253 Anthony Nadalin, Microsoft  
6254 Martin Raeppe, SAP AG  
6255 Bruce Rich, IBM  
6256 Ramana Turlapati, Oracle  
6257 Greg Whitehead, Hewlett-Packard  
6258 Ching-Yun (C.Y.) Chao, IBM  
6259

6260 **TC Members during the development of this specification:**

6261 Don Adams, Tibco Software Inc.  
6262 Jan Alexander, Microsoft Corporation  
6263 Steve Anderson, BMC Software  
6264 Donal Arundel, IONA Technologies  
6265 Howard Bae, Oracle Corporation  
6266 Abbie Barbir, Nortel Networks Limited  
6267 Charlton Barreto, Adobe Systems  
6268 Michael Botha, Software AG, Inc.  
6269 Toufic Boubez, Layer 7 Technologies Inc.  
6270 Norman Brickman, Mitre Corporation  
6271 Melissa Brumfield, Booz Allen Hamilton  
6272 Geoff Bullen, Microsoft Corporation  
6273 Lloyd Burch, Novell  
6274 Scott Cantor, Internet2  
6275 Greg Carpenter, Microsoft Corporation  
6276 Steve Carter, Novell  
6277 Symon Chang, Oracle Corporation  
6278 Martin Chapman, Oracle Corporation  
Ching-Yun (C.Y.) Chao, IBM

6279 Kate Cherry, Lockheed Martin  
6280 Henry (Hyenvui) Chung, IBM  
6281 Luc Clement, Systinet Corp.  
6282 Paul Cotton, Microsoft Corporation  
6283 Glen Daniels, Sonic Software Corp.  
6284 Peter Davis, Neustar, Inc.  
6285 Duane DeCouteau, Veterans Health Administration  
6286 Martijn de Boer, SAP AG  
6287 Werner Dittmann, Siemens AG  
6288 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
6289 Fred Dushin, IONA Technologies  
6290 Petr Dvorak, Systinet Corp.  
6291 Colleen Evans, Microsoft Corporation  
6292 Ruchith Fernando, WSO2  
6293 Mark Fussell, Microsoft Corporation  
6294 Vijay Gajjala, Microsoft Corporation  
6295 Marc Goodner, Microsoft Corporation  
6296 Hans Granqvist, VeriSign  
6297 Martin Gudgin, Microsoft Corporation  
6298 Tony Gullotta, SOA Software Inc.  
6299 Jiandong Guo, Sun Microsystems  
6300 Phillip Hallam-Baker, VeriSign  
6301 Patrick Harding, Ping Identity Corporation  
6302 Heather Hinton, IBM  
6303 Frederick Hirsch, Nokia Corporation  
6304 Jeff Hodges, Neustar, Inc.  
6305 Will Hopkins, Oracle Corporation  
6306 Alex Hristov, Otecia Incorporated  
6307 John Hughes, PA Consulting  
6308 Diane Jordan, IBM  
6309 Venugopal K, Sun Microsystems  
6310 Chris Kaler, Microsoft Corporation  
6311 Dana Kaufman, Forum Systems, Inc.  
6312 Paul Knight, Nortel Networks Limited  
6313 Ramanathan Krishnamurthy, IONA Technologies  
6314 Christopher Kurt, Microsoft Corporation  
6315 Kelvin Lawrence, IBM  
6316 Hubert Le Van Gong, Sun Microsystems  
6317 Jong Lee, Oracle Corporation  
6318 Rich Levinson, Oracle Corporation  
6319 Tommy Lindberg, Dajeil Ltd.  
6320 Mark Little, JBoss Inc.  
6321 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.  
6322 Eve Maler, Sun Microsystems  
6323 Ashok Malhotra, Oracle Corporation  
6324 Anand Mani, CrimsonLogic Pte Ltd  
6325 Jonathan Marsh, Microsoft Corporation  
6326 Robin Martherus, Oracle Corporation  
6327 Miko Matsumura, Infravio, Inc.  
6328 Gary McAfee, IBM  
6329 Michael McIntosh, IBM  
6330 John Merrells, Sxip Networks SRL  
6331 Jeff Mischkinsky, Oracle Corporation  
6332 Prateek Mishra, Oracle Corporation  
6333 Bob Morgan, Internet2  
6334 Vamsi Motukuru, Oracle Corporation  
6335 Raajmohan Na, EDS

6336 Anthony Nadalin, IBM  
6337 Andrew Nash, Reactivity, Inc.  
6338 Eric Newcomer, IONA Technologies  
6339 Duane Nickull, Adobe Systems  
6340 Toshihiro Nishimura, Fujitsu Limited  
6341 Rob Philpott, RSA Security  
6342 Denis Pilipchuk, Oracle Corporation.  
6343 Darren Platt, Ping Identity Corporation  
6344 Martin Raeppele, SAP AG  
6345 Nick Ragouzis, Enosis Group LLC  
6346 Prakash Reddy, CA  
6347 Alain Regnier, Ricoh Company, Ltd.  
6348 Irving Reid, Hewlett-Packard  
6349 Bruce Rich, IBM  
6350 Tom Rutt, Fujitsu Limited  
6351 Maneesh Sahu, Actional Corporation  
6352 Frank Siebenlist, Argonne National Laboratory  
6353 Joe Smith, Apani Networks  
6354 Davanum Srinivas, WSO2  
6355 David Staggs, Veterans Health Administration  
6356 Yakov Sverdlov, CA  
6357 Gene Thurston, AmberPoint  
6358 Victor Valle, IBM  
6359 Asir Vedamuthu, Microsoft Corporation  
6360 Greg Whitehead, Hewlett-Packard  
6361 Ron Williams, IBM  
6362 Corinna Witt, Oracle Corporation  
6363 Kyle Young, Microsoft Corporation  
6364

6365

## B. Revision History

6366

[optional; should not be included in OASIS Standards]

6367

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

6368