



# WS-SecurityPolicy Examples Version 1.0

## Committee Specification 02

4 November 2010

### Specification URIs:

#### This Version:

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

#### Previous 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>

#### 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>).

---

## 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.

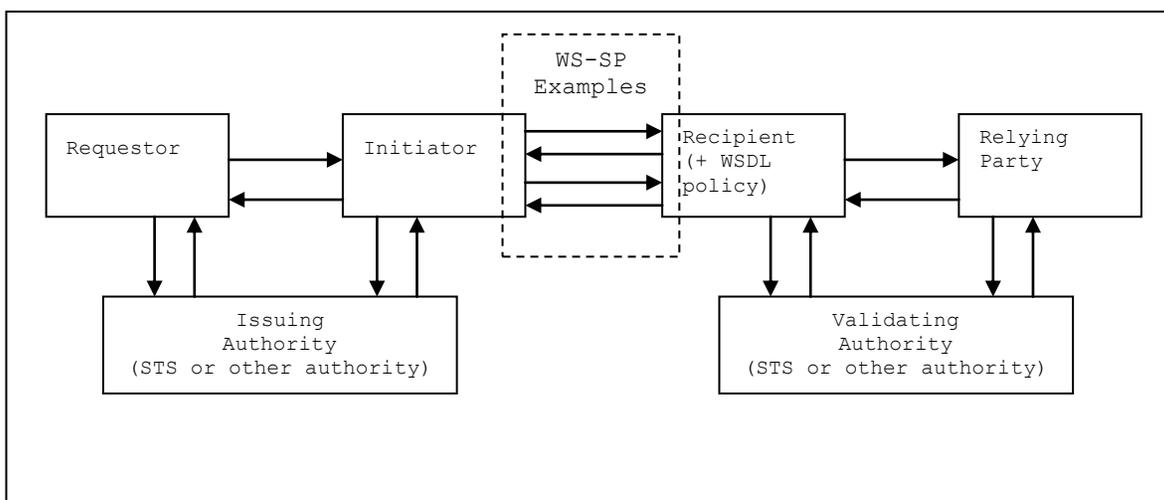
47 The reason for providing this context is to help interested readers understand that while the technology  
48 may be securing the integrity and confidentiality of the messages, there are additional questions about  
49 transactions such as who is liable for any commitments resulting from the transaction 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 • **Requestor:** the person or entity requesting the service and who will be supplying credentials  
121 issued by the IssuingAuthority and will be validated by a ValidatingAuthority. The Requestor is the  
122 logical entity that supplies the credentials that ultimately get passed to the Recipient that will be  
123 trusted by the RelyingParty if they are validated by the ValidatingAuthority. (Note: the logistics of  
124 supplying the trusted credential is distinct from the logistics of packaging up the credentials within  
125 a WS-Security header in a SOAP message and transmitting that message to the Recipient. The  
126 latter logistics are covered by the Initiator, described below. The Requestor and Initiator may be  
127 combined into a single physical entity and this is a common occurrence, however they do not  
128 have to be and it is also a common occurrence that they are separate physical entities. The latter  
129 case is typified by the Requestor being a user at a browser that may be prompted for credentials  
130 by the Initiator on a server using HTTP to challenge the Requestor for a username and  
131 password.)

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

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

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

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

## 182 1.1.2 Concepts

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

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

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

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

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

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

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

### 223 1.1.2.1 X509 Certificates

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

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

## 238 1.2 Terminology

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

## 242 1.3 Normative References

- 243 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
244 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 245 [WSS10-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.0 (WS-  
246 Security 2004)”, March 2004.  
247 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)  
248 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 249 [WSS11-SOAPMSG] OASIS Standard, “Web Services Security: SOAP Message Security 1.1”,  
250 OASIS Standard incorporating Approved Errata, 01 November 2006.  
251 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.pdf)  
252 [SOAPMessageSecurity.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.pdf)
- 253 [WSS10-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.0”,  
254 March 2004.  
255 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)  
256 [profile-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
- 257 [WSS11-USERNAME] OASIS Standard, “Web Services Security: UsernameToken Profile 1.1”,  
258 February 2006.  
259 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf>
- 260 [WSS10-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token  
261 Profile 1.0”, March 2004.  
262 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)  
263 [1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)
- 264 [WSS11-X509-PROFILE] OASIS Standard, “Web Services Security: X.509 Certificate Token  
265 Profile 1.1”, OASIS Standard incorporating Approved Errata, 01 November 2006.  
266 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-x509TokenProfile.pdf)  
267 [x509TokenProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-x509TokenProfile.pdf)
- 268 [WSS10-SAML11-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.0”,  
269 December 2004.  
270 <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>
- 271 [WSS11-SAML1120-PROFILE] OASIS Standard, “Web Services Security: SAML Token Profile 1.1”,  
272 OASIS Standard Incorporating Approved Errata, 1 November 2006.  
273 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SAMLSAMLTokenProfile.pdf)  
274 [SAMLSAMLTokenProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SAMLSAMLTokenProfile.pdf)
- 275 [WSS11-LIBERTY-SAML20-PROFILE]  
276 [http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-](http://www.projectliberty.org/liberty/content/download/894/6258/file/liberty-idwsf-security-mechanisms-saml-profile-v2.0.pdf)  
277 [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)
- 278 [WSS-TC-ALL-TOKEN-PROFILES] OASIS WS-Security TC links to all other WSS 1.0 and WSS 1.1  
279 token profiles  
280 <http://www.oasis-open.org/specs/index.php#wssv1.0>  
281 <http://www.oasis-open.org/specs/index.php#wssv1.1>  
282 and other documents (interop) in the TC repository:  
283 [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss)

284 [WS-SECURITYPOLICY] OASIS Standard, "WS-SecurityPolicy 1.2", July 2007.  
 285 [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)  
 286 [spec-os.doc](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.doc)

287 [WS-SECURECONVERSATION] OASIS Standard, "WS-SecureConversation 1.3", March 2007.  
 288 [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)  
 289 [secureconversation-1.3-os.doc](http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.doc)

290 [WS-TRUST] OASIS Standard, "WS-Trust 1.3", March 2007.  
 291 <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.doc>

292 [WS-POLICY] <http://www.w3.org/TR/ws-policy>

293 [WS-POLICYATTACHMENT] <http://www.w3.org/TR/ws-policy-attach>

294 [WSDL] <http://www.w3.org/TR/wsdl>

295 [SSL] <http://www.ietf.org/rfc/rfc2246.txt>

296 [SAML11-CORE] OASIS Standard, "Assertions and Protocol for the OASIS Security Assertion  
 297 Markup Language (SAML) V1.1", September 2003.  
 298 [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)  
 299 [1.1.pdf](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)

300 [SAML20-CORE] OASIS Standard, "Assertions and Protocols for the OASIS Security Assertion  
 301 Markup Language (SAML) V2.0", March 2005.  
 302 <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>

303 [XML-DSIG] <http://www.w3.org/TR/xmlsig-core/>

304 [XML-ENCR] <http://www.w3.org/TR/xmlenc-core/>

## 305 1.4 Non-Normative References

306 WSS10-INTEROP-01: OASIS Working Draft 06, "Web Services Security: Interop 1 Scenarios",  
 307 Jun 2003.  
 308 [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)  
 309 [06-merged-changes.pdf](http://www.oasis-open.org/committees/download.php/11374/wss-interop1-draft-06-merged-changes.pdf)

310

311 WSS10-INTEROP-02: OASIS Working Draft 06, "Web Services Security: Interop 2 Scenarios",  
 312 Oct 2003.  
 313 [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)  
 314 [06-merged.doc](http://www.oasis-open.org/committees/download.php/11375/wss-interop2-draft-06-merged.doc)

315 WSS11-INTEROP-01: OASIS Working Draft 02, "Web Services Security 1.1: Interop Scenario",  
 316 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)  
 317 [interop-draft-01.doc](http://www.oasis-open.org/committees/download.php/12997/wss-11-interop-draft-01.doc)

318 WSS10-KERBEROS-INTEROP: OASIS Working Draft 02, "Web Services Security: Kerberos Token  
 319 Profile Interop Scenarios", Jan 2005.  
 320 [http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-](http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-interop.doc)  
 321 [interop.doc](http://www.oasis-open.org/committees/download.php/11720/wss-kerberos-interop.doc)

322 WSS10-SAML11-INTEROP: OASIS Working Draft 12, "Web Services Security: SAML Interop 1  
 323 Scenarios", July 2004.  
 324 [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)  
 325 [draft-12.doc](http://www.oasis-open.org/committees/download.php/7702/wss-saml-interop1-draft-12.doc)

326 WSS11-SAML1120-INTEROP: OASIS Working Draft 4, "Web Services Security: SAML 2.0 Interop  
 327 Scenarios", January 2005.  
 328 [http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)  
 329 [draft-v4.doc](http://www.oasis-open.org/committees/download.php/16556/wss-saml2-interop-draft-v4.doc)

330 WSSX-PRE-INTEROP: OASIS Contribution Version 1.0d, "WS-SecureConversation and WS-Trust  
 331 Interop Scenarios", September 29, 2004.  
 332 [http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-](http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html)  
 333 [examples.html](http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html)

- 334 WSSX-WSTR-WSSC-INTEROP: OASIS White Paper Version ED-10, “WS-SX Interop Scenarios - Phase  
335 2 Scenarios for demonstration of WS-SX TC specifications”, October 31, 2006.  
336 [http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-  
337 10.doc](http://www.oasis-open.org/committees/download.php/20954/ws-sx-interop-ed-10.doc)
- 338 WS-SECURE-INTEROP: OASIS White Paper Version ED-01, “Examples of Secure Web Service  
339 Message Exchange”, July 11, 2008.  
340 [http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-  
341 message-examples.doc](http://www.oasis-open.org/committees/download.php/28803/ws-sx-secure-message-examples.doc)
- 342 WSI-SCM-SAMPLEAPPL: <http://java.sun.com/webservices/docs/1.4/wsi-sampleapp/index.html> (login  
343 required)

## 344 **1.5 Specifications**

## 345 **1.6 Interops and Sample Messages**

---

## 346 2 Scenarios

### 347 2.1 UsernameToken

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

#### 350 2.1.1 UsernameToken – no security binding

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

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

##### 357 2.1.1.1 UsernameToken with plain text password

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

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

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

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

369

```
370 (P001) <wsp:Policy>  
371 (P002) <sp:SupportingTokens>  
372 (P003) <wsp:Policy>  
373 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-  
374 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient"  
375 (P005) </wsp:Policy>  
376 (P006) </sp:SupportingTokens>  
377 (P007) </wsp:Policy>
```

378

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

380

```
381 (M001) <?xml version="1.0" encoding="utf-8" ?>  
382 (M002) <soap:Envelope xmlns:soap="...">  
383 (M003) <soap:Header>  
384 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">  
385 (M005) <wsse:UsernameToken>  
386 (M006) <wsse:Username>Chris</wsse:Username>  
387 (M007) <wsse:Password Type="http://docs.oasis-  
388 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-  
389 1.0#PasswordText">srhC</wsse:Password>  
390 (M008) <wsse:Nonce EncodingType="...#Base64Binary"  
391 (M009) >pN...=</wsse:Nonce>
```

```

392 (M010) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
393 (M011) </wsse:UsernameToken>
394 (M012) </wsse:Security>
395 (M013) </soap:Header>
396 (M014) <soap:Body>
397 (M015) <Ping xmlns="http://xmlsoap.org/Ping">
398 (M016) <text>EchoString</text>
399 (M017) </Ping>
400 (M018) </soap:Body>
401 (M019) </soap:Envelope>

```

402

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

### 409 2.1.1.2 UsernameToken without password

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

```

416
417 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
418 (P002) <sp:SupportingTokens>
419 (P003) <wsp:Policy>
420 (P004) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
421 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
422 (P005) <wsp:Policy>
423 (P006) <sp:NoPassword/>
424 (P007) </wsp:Policy>
425 (P008) </sp:UsernameToken>
426 (P009) </wsp:Policy>
427 (P010) </sp:SupportingTokens>
428 (P011) </wsp:Policy>

```

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

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

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

```

433
434 (M001) <?xml version="1.0" encoding="utf-8" ?>
435 (M002) <soap:Envelope xmlns:soap="...">
436 (M003) <soap:Header>
437 (M004) <wsse:Security soap:mustUnderstand="1"
438 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
439 wssecurity-secext-1.0.xsd">
440 (M005) <wsse:UsernameToken>
441 (M006) <wsse:Username>Chris</wsse:Username>
442 (M007) </wsse:UsernameToken>
443 (M008) </wsse:Security>
444 (M009) </soap:Header>
445 (M010) <soap:Body>
446 (M011) <Ping xmlns="http://xmlsoap.org/Ping">

```

```

447 (M012)      <text>EchoString</text>
448 (M013)      </Ping>
449 (M014)      </soap:Body>
450 (M015)      </soap:Envelope>

```

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

### 453 2.1.1.3 UsernameToken with timestamp, nonce and password hash

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

```

459
460 (P001)      <wsp:Policy xmlns:wsp="..." xmlns:sp="...">
461 (P002)      <sp:SupportingTokens>
462 (P003)      <wsp:Policy>
463 (P004)      <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
464 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
465 (P005)      <wsp:Policy>
466 (P006)      <sp:HashPassword/>
467 (P007)      </wsp:Policy>
468 (P008)      </sp:UsernameToken>
469 (P009)      </wsp:Policy>
470 (P010)     </sp:SupportingTokens>
471 (P011)     </wsp:Policy>

```

472  
473 An example of a message that conforms to the above stated policy is as follows.

```

474
475 (M001)     <?xml version="1.0" encoding="utf-8" ?>
476 (M002)     <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
477 (M003)     <soap:Header>
478 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="...">
479 (M005)     <wsse:UsernameToken
480           wsu:Id="uuid-7cee5976-0111-e9c1-e34b-af1e85fa3866">
481 (M006)     <wsse:Username>Chris</wsse:Username>
482 (M007)     <wsse:Password Type="http://docs.oasis-
483 open.org/wss/2004/01/oasis-200401-wss-username-token-profile-
484 1.0#PasswordDigest"
485           >weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==</wsse:Password>
486 (M008)     <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
487 (M009)     <wsu:Created>2007-05-01T01:15:30Z</wsu:Created>
488 (M010)     </wsse:UsernameToken>
489 (M011)     </wsse:Security>
490 (M012)     </soap:Header>
491 (M013)     <soap:Body wsu:Id="uuid-7cee4264-0111-e0cb-8329-af1e85fa3866">
492 (M014)     <Ping xmlns="http://xmlsoap.org/Ping">
493 (M015)     <text>EchoString</text>
494 (M016)     </Ping>
495 (M017)     </soap:Body>
496 (M018)     </soap:Envelope>

```

497  
498 This message is very similar to the one in section 2.1.1.1. A UsernameToken starts on line (M005) to  
499 satisfy the UsernameToken assertion. However, in this example the Password element on line (M007) is  
500 of type PasswordDigest to satisfy the HashPassword assertion on line (P006). The Nonce (M008) and

501 Created timestamp (M009) are also included as dictated by the HashPassword assertion. The Nonce and  
502 timestamp values are included in the password digest on line (M007).

## 503 2.1.2 Use of SSL Transport Binding

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

510

```
511 <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
512   <sp:TransportBinding>  
513     <wsp:Policy>  
514       <sp:TransportToken>  
515         <wsp:Policy>  
516           <sp:HttpsToken />  
517         </wsp:Policy>  
518       </sp:TransportToken>  
519       <sp:AlgorithmSuite>  
520         <wsp:Policy>  
521           <sp:Basic256 />  
522         </wsp:Policy>  
523       </sp:AlgorithmSuite>  
524       <sp:Layout>  
525         <wsp:Policy>  
526           <sp:Strict />  
527         </wsp:Policy>  
528       </sp:Layout>  
529       <sp:IncludeTimestamp />  
530     </wsp:Policy>  
531   </sp:TransportBinding>  
532 </wsp:Policy>
```

### 533 2.1.2.1 UsernameToken as supporting token

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

537

```
538 (P001) <wsp:Policy xmlns:wsp="..." xmlns:sp="...">  
539 (P002)   <sp:TransportBinding>  
540 (P003)     <wsp:Policy>  
541 (P004)       <sp:TransportToken>  
542 (P005)         <wsp:Policy>  
543 (P006)           <sp:HttpsToken/>  
544 (P007)         </wsp:Policy>  
545 (P008)       </sp:TransportToken>  
546 (P009)       <sp:AlgorithmSuite>  
547 (P010)         <wsp:Policy>  
548 (P011)           <sp:Basic256/>  
549 (P012)         </wsp:Policy>  
550 (P013)       </sp:AlgorithmSuite>  
551 (P014)       <sp:Layout>  
552 (P015)         <wsp:Policy>  
553 (P016)           <sp:Strict/>  
554 (P017)         </wsp:Policy>  
555 (P018)       </sp:Layout>  
556 (P019)       <sp:IncludeTimestamp/>  
557 (P020)     </wsp:Policy>
```

```

558 (P021) </sp:TransportBinding>
559 (P022) <sp:SupportingTokens>
560 (P023) <wsp:Policy>
561 (P024) <sp:UsernameToken/>
562 (P025) </wsp:Policy>
563 (P026) </sp:SupportingTokens>
564 (P027) </wsp:Policy>

```

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

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

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

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

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

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

```

579 (M001) <?xml version="1.0" encoding="utf-8" ?>
580 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsu="...">
581 (M003) <soap:Header>
582 (M004) <wsse:Security soap:mustUnderstand="1"
583 < xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
584 wss-wssecurity-secext-1.0.xsd">
585 (M005) <wsu:Timestamp wsu:Id="uuid-8066364f-0111-f371-47bf-ba986d2d7dc4">
586 (M006) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
587 (M007) </wsu:Timestamp>
588 (M008) <wsse:UsernameToken
589 < wsu:Id="uuid-8066368d-0111-e744-f37b-ba986d2d7dc4">
590 (M009) <wsse:Username>Chris</wsse:Username>
591 (M010) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
592 200401-wss-username-token-profile-1.0#PasswordText">sirhC</wsse:Password>
593 (M011) </wsse:UsernameToken>
594 (M012) </wsse:Security>
595 (M013) </soap:Header>
596 (M014) <soap:Body wsu:Id="uuid-8066363f-0111-ffdc-de48-ba986d2d7dc4">
597 (M015) <Ping xmlns="http://xmlsoap.org/Ping">
598 (M016) <text>EchoString</text>
599 (M017) </Ping>
600 (M018) </soap:Body>
601 (M019) </soap:Envelope>

```

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

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

605

### 606 2.1.3 (WSS 1.0) UsernameToken with Mutual X.509v3 Authentication, Sign, 607 Encrypt

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

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

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

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

621 The policy is as follows:

```
622 (P001) <wsp:Policy wsu:Id="wss10_up_cert_policy" >  
623 (P002)   <wsp:ExactlyOne>  
624 (P003)     <wsp>All>  
625 (P004)       <sp:AsymmetricBinding>  
626 (P005)         <wsp:Policy>  
627 (P006)           <sp:InitiatorToken>  
628 (P007)             <wsp:Policy>  
629 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-  
630 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
631 (P009)                 <wsp:Policy>  
632 (P010)                   <sp:WssX509V3Token10/>  
633 (P011)                     </wsp:Policy>  
634 (P012)                   </sp:X509Token>  
635 (P013)                   </wsp:Policy>  
636 (P014)                 </sp:InitiatorToken>  
637 (P015)                 <sp:RecipientToken>  
638 (P016)                   <wsp:Policy>  
639 (P017)                     <sp:X509Token sp:IncludeToken="http://docs.oasis-  
640 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
641 (P018)                       <wsp:Policy>  
642 (P019)                         <sp:WssX509V3Token10/>  
643 (P020)                           </wsp:Policy>  
644 (P021)                         </sp:X509Token>  
645 (P022)                         </wsp:Policy>  
646 (P023)                       </sp:RecipientToken>  
647 (P024)                       <sp:AlgorithmSuite>  
648 (P025)                         <wsp:Policy>  
649 (P026)                           <sp:Basic256/>  
650 (P027)                           </wsp:Policy>  
651 (P028)                       </sp:AlgorithmSuite>  
652 (P029)                       <sp:Layout>  
653 (P030)                         <wsp:Policy>  
654 (P031)                           <sp:Strict/>  
655 (P032)                           </wsp:Policy>  
656 (P033)                         </sp:Layout>  
657 (P034)                         <sp:IncludeTimestamp/>  
658 (P035)                         <sp:OnlySignEntireHeadersAndBody/>  
659 (P036)                         </wsp:Policy>  
660 (P037)                       </sp:AsymmetricBinding>  
661 (P038)                       <sp:Wss10>  
662 (P039)                         <wsp:Policy>  
663 (P040)                           <sp:MustSupportRefKeyIdentifier/>
```

```

664      (P041)      </wsp:Policy>
665      (P042)      </sp:Wss10>
666      (P043)      <sp:SignedEncryptedSupportingTokens>
667      (P044)      <wsp:Policy>
668      (P045)      <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
669      open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
670      (P046)      <wsp:Policy>
671      (P047)      <sp:WssUsernameToken10/>
672      (P048)      </wsp:Policy>
673      (P049)      </sp:UsernameToken>
674      (P050)      </wsp:Policy>
675      (P051)      </sp:SignedEncryptedSupportingTokens>
676      (P052)      </wsp:All>
677      (P053)      </wsp:ExactlyOne>
678      </wsp:Policy>
679
680      (P054) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
681      (P055) <wsp:ExactlyOne>
682      (P056) <wsp:All>
683      (P057) <sp:SignedParts>
684      (P058) <sp:Body/>
685      (P059) </sp:SignedParts>
686      (P060) <sp:EncryptedParts>
687      (P061) <sp:Body/>
688      (P062) </sp:EncryptedParts>
689      (P063) </wsp:All>
690      (P064) </wsp:ExactlyOne>
691      (P065) </wsp:Policy>
692
693      (P066) <wsp:Policy wsu:Id="WSS10UsernameForCertificate_output_policy">
694      (P067) <wsp:ExactlyOne>
695      (P068) <wsp:All>
696      (P069) <sp:SignedParts>
697      (P070) <sp:Body/>
698      (P071) </sp:SignedParts>
699      (P072) <sp:EncryptedParts>
700      (P073) <sp:Body/>
701      (P074) </sp:EncryptedParts>
702      (P075) </wsp:All>
703      (P076) </wsp:ExactlyOne>
704      (P077) </wsp:Policy>

```

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

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

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

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

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

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

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

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

```
729 (M001) <?xml version="1.0" encoding="utf-8" ?>
730 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
731 (M003)   <soap:Header>
732 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
733 (M005)       <xenc:ReferenceList>
734 (M006)         <xenc:DataReference URI="#encUT"/>
735 (M007)         <xenc:DataReference URI="#encBody"/>
736 (M008)       </xenc:ReferenceList>
737 (M009)       <wsu:Timestamp wsu:Id="T0">
738 (M010)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
739 (M011)       </wsu:Timestamp>
740 (M012)       <wsse:BinarySecurityToken wsu:Id="binaryToken" ValueType="...#X509v3"
741 EncodingType="...#Base64Binary">
742 (M013)         MIIEZzCCA9CgAwIBAgIQEmtJZc0...
743 (M014)       </wsse:BinarySecurityToken>
744 (M015)       <xenc:EncryptedData wsu:Id="encUT">
745 (M016)         <ds:KeyInfo>
746 (M017)           <wsse:SecurityTokenReference>
747 (M018)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
748 ValueType="...#X509SubjectKeyIdentifier">
749 (M019)               MIGfMa0GCSq...
750 (M020)             </wsse:KeyIdentifier>
751 (M021)           </wsse:SecurityTokenReference>
752 (M022)         </ds:KeyInfo>
753 (M023)         <xenc:CipherData>
754 (M024)           <xenc:CipherValue>...</xenc:CipherValue>
755 (M025)         </xenc:CipherData>
756 (M026)       </xenc:EncryptedData>
757 (M027)       <ds:Signature>
758 (M028)         <ds:SignedInfo>...
759 (M029)           <ds:Reference URI="#T0">...</ds:Reference>
760 (M030)           <ds:Reference URI="#usernameToken">...</ds:Reference>
761 (M031)           <ds:Reference URI="#body">...</ds:Reference>
762 (M032)         </ds:SignedInfo>
763 (M033)       <ds:SignatureValue>HFLP...</ds:SignatureValue>
764 (M034)       <ds:KeyInfo>
765 (M035)         <wsse:SecurityTokenReference>
766 (M036)           <wsse:Reference URI="#binaryToken"/>
767 (M037)         </wsse:SecurityTokenReference>
768 (M038)       </ds:KeyInfo>
769 (M039)       </ds:Signature>
770 (M040)     </wsse:Security>
771 (M041)   </soap:Header>
772 (M042)   <soap:Body wsu:Id="body">
773 (M043)     <xenc:EncryptedData wsu:Id="encBody">
774 (M044)       <ds:KeyInfo>
775 (M045)         <wsse:SecurityTokenReference>
776 (M046)           <wsse:KeyIdentifier EncodingType="...#Base64Binary"
777 ValueType="...#X509SubjectKeyIdentifier">
778 (M047)             MIGfMa0GCSq...
779 (M048)           </wsse:KeyIdentifier>
780 (M049)         </wsse:SecurityTokenReference>
781 (M050)       </ds:KeyInfo>
782 (M051)       <xenc:CipherData>
783 (M052)         <xenc:CipherValue>...</xenc:CipherValue>
```

```
784 (M053) </xenc:CipherData>
785 (M054) </xenc:EncryptedData>
786 (M055) </soap:Body>
787 (M056) </soap:Envelope>
```

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

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

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

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

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

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

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

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

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

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

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

### 818 **2.1.3.1 (WSS 1.0) Encrypted UsernameToken with X.509v3**

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

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

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

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

829 The policy is as follows:

```
830 (P001) <wsp:Policy wsu:Id="wss10_encrypted_unt_policy" >
831 (P002) <sp:AsymmetricBinding>
832 (P003) <wsp:Policy>
833 (P004) <sp:InitiatorEncryptionToken>
834 (P005) <wsp:Policy>
```

```

835 (P006) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
836 sx/ws-securitypolicy/200702/IncludeToken/Never">
837 (P007) <wsp:Policy>
838 (P008) <sp:WssX509V3Token10/>
839 (P009) </wsp:Policy>
840 (P010) </sp:X509Token>
841 (P011) </wsp:Policy>
842 (P012) </sp:InitiatorEncryptionToken>
843 (P013) <sp:RecipientSignatureToken>
844 (P014) <wsp:Policy>
845 (P015) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
846 sx/ws-securitypolicy/200702/IncludeToken/Never">
847 (P016) <wsp:Policy>
848 (P017) <sp:WssX509V3Token10/>
849 (P018) </wsp:Policy>
850 (P019) </sp:X509Token>
851 (P020) </wsp:Policy>
852 (P021) </sp:RecipientSignatureToken>
853 (P022) <sp:AlgorithmSuite>
854 (P023) <wsp:Policy>
855 (P024) <sp:Basic256/>
856 (P025) </wsp:Policy>
857 (P026) </sp:AlgorithmSuite>
858 (P027) <sp:Layout>
859 (P028) <wsp:Policy>
860 (P029) <sp:Lax/>
861 (P030) </wsp:Policy>
862 (P031) </sp:Layout>
863 (P032) <sp:IncludeTimestamp/>
864 (P033) <sp:OnlySignEntireHeadersAndBody/>
865 (P034) </wsp:Policy>
866 (P035) </sp:AsymmetricBinding>
867 (P036) <sp:Wss10>
868 (P037) <wsp:Policy>
869 (P038) <sp:MustSupportRefKeyIdentifier/>
870 (P039) <sp:MustSupportRefIssuerSerial/>
871 (P040) </wsp:Policy>
872 (P041) </sp:Wss10>
873 (P042) <sp:EncryptedSupportingTokens>
874 (P043) <wsp:Policy>
875 (P044) <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-
876 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
877 (P045) <wsp:Policy>
878 (P046) <sp:HashPassword/>
879 (P047) <sp:WssUsernameToken10/>
880 (P048) </wsp:Policy>
881 (P049) </sp:UsernameToken>
882 (P050) </wsp:Policy>
883 (P051) </sp:EncryptedSupportingTokens>
884 (P052) </wsp:Policy>
885
886 (P053) <wsp:Policy wsu:Id="WSS10UsernameForCertificates_input_policy">
887 (P054) <wsp:ExactlyOne>
888 (P055) <wsp>All>
889 (P056) <sp:EncryptedParts>
890 (P057) <sp:Body/>
891 (P058) </sp:EncryptedParts>
892 (P059) </wsp>All>
893 (P060) </wsp:ExactlyOne>
894 (P061) </wsp:Policy>
895
896 (P062) <wsp:Policy wsu:Id="WSS10UsernameForCertificate output_policy">
897 (P063) <wsp:ExactlyOne>
898 (P064) <wsp>All>

```

```

899 (P065) <sp:SignedParts>
900 (P066) <sp:Body/>
901 (P067) </sp:SignedParts>
902 (P068) </wsp:All>
903 (P069) </wsp:ExactlyOne>
904 (P070) </wsp:Policy>

```

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

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

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

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

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

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

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

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

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

930

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

```

932 (M001) <?xml version="1.0" encoding="utf-8" ?>
933 (M002) <soapenv:Envelope xmlns:soapenv="..." xmlns:xenc="..." . . . >
934 (M003) <soapenv:Header>
935 (M004) <wsse:Security xmlns:wsse="..." xmlns:wspu="..." xmlns:ds="..."
936 soapenv:mustUnderstand="1" >
937 (M005) <xenc:EncryptedKey>
938 (M006) <xenc:EncryptionMethod Algorithm=". . .#rsa-oaep-mgf1p">
939 (M007) <ds:DigestMethod Algorithm=". . .#sha1" />
940 (M008) </xenc:EncryptionMethod>
941 (M009) <ds:KeyInfo>
942 (M010) <wsse:SecurityTokenReference >
943 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
944 (M012) Value="...#X509SubjectKeyIdentifier">CuJ. .
945 .=</wsse:KeyIdentifier>
946 (M013) </wsse:SecurityTokenReference>
947 (M014) </ds:KeyInfo>
948 (M015) <xenc:CipherData>
949 (M016) <xenc:CipherValue>dbj...=</xenc:CipherValue>
950 (M017) </xenc:CipherData>
951 (M018) <xenc:ReferenceList>
952 (M019) <xenc:DataReference URI="#encBody"/>
953 (M020) <xenc:DataReference URI="#encUnt"/>
954 (M021) </xenc:ReferenceList>

```

```

955 (M022) </xenc:EncryptedKey>
956 (M023) <xenc:EncryptedData Id="encUnt" Type="...#Element"
957 MimeType="text/xml" Encoding="UTF-8" >
958 (M024) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
959 (M025) <xenc:CipherData>
960 (M026) <xenc:CipherValue>Kzf...=</xenc:CipherValue>
961 (M027) </xenc:CipherData>
962 (M028) </xenc:EncryptedData>
963 (M029) <wsu:Timestamp >
964 (M030) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
965 (M031) </wsu:Timestamp>
966 (M032) </wsse:Security>
967 (M033) </soapenv:Header>
968 (M034) <soapenv:Body >
969 (M035) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
970 Encoding="UTF-8" >
971 (M036) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
972 (M037) <xenc:CipherData>
973 (M038) <xenc:CipherValue>a/9...B</xenc:CipherValue>
974 (M039) </xenc:CipherData>
975 (M040) </xenc:EncryptedData>
976 (M041) </soapenv:Body>
977 (M042) </soapenv:Envelope>

```

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

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

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

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

```

991 (M043) <wsse:UsernameToken wsu:Id="usernameToken">
992 (M044) <wsse:Username>Chris</wsse:Username>
993 (M045) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
994 wss-username-token-profile-1.0#PasswordDigest">oY...=</wsse:Password>
995 (M046) <wsse:Nonce EncodingType="...#Base64Binary">pN...=</wsse:Nonce>
996 (M047) <wsu:Created>2007-03-28T18:42:03Z</wsu:Created>
997 (M048) </wsse:UsernameToken>

```

998 Line (M046) contains the Nonce element and Line (M047) contains a timestamp. It is recommended that  
999 these two elements should also be included in the PasswordText case for better security  
1000 [\[WSS10-USERNAME\]](#).

## 1001 2.1.4 (WSS 1.1), User Name with Certificates, Sign, Encrypt

1002 This scenario is based on the “Examples of Secure Web Service Message Exchange Document”  
1003 [\[WS-SECURE-INTEROP\]](#).

1004 The use case here is the following: the Initiator generates a symmetric key; the symmetric key is  
1005 encrypted using the Recipient’s certificate and placed in an encrypted key element. The UsernameToken  
1006 identifying the Requestor and message body are signed using the symmetric key. The body and  
1007 UsernameToken are also encrypted. The Authority for this request is generally the Subject of the  
1008 Initiator’s X509 certificate.

1009 We can use the symmetric security binding [WSSP] with X509token as the protection token to illustrate  
1010 this case. If derived keys are to be used, then the derived keys property of X509Token should be set.

1011 The policy is as follows:

```
1012 (P001) <wsp:Policy wsu:Id="WSS11UsernameWithCertificates_policy">
1013 (P002)   <wsp:ExactlyOne>
1014 (P003)   <wsp:All>
1015 (P004)     <sp:SymmetricBinding>
1016 (P005)       <wsp:Policy>
1017 (P006)         <sp:ProtectionToken>
1018 (P007)           <wsp:Policy>
1019 (P008)             <sp:X509Token sp:IncludeToken="http://docs.oasis-
1020 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
1021 (P009)               <wsp:Policy>
1022 (P010)                 <sp:RequireThumbprintReference/>
1023 (P011)                 <sp:WssX509V3Token11/>
1024 (P012)               </wsp:Policy>
1025 (P013)             </sp:X509Token>
1026 (P014)           </wsp:Policy>
1027 (P015)         </sp:ProtectionToken>
1028 (P016)       <sp:AlgorithmSuite>
1029 (P017)         <wsp:Policy>
1030 (P018)           <sp:Basic256/>
1031 (P019)         </wsp:Policy>
1032 (P020)       </sp:AlgorithmSuite>
1033 (P021)     <sp:Layout>
1034 (P022)       <wsp:Policy>
1035 (P023)         <sp:Strict/>
1036 (P024)       </wsp:Policy>
1037 (P025)     </sp:Layout>
1038 (P026)   <sp:IncludeTimestamp/>
1039 (P027)   <sp:OnlySignEntireHeadersAndBody/>
1040 (P028) </wsp:Policy>
1041 (P029) </sp:SymmetricBinding>
1042 (P030) <sp:SignedEncryptedSupportingTokens>
1043 (P031) <wsp:Policy>
1044 (P032)   <sp:UsernameToken sp:IncludeToken="http://docs.oasis-
1045 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1046 (P033)     <wsp:Policy>
1047 (P034)       <sp:WssUsernameToken11/>
1048 (P035)     </wsp:Policy>
1049 (P036)   </sp:UsernameToken>
1050 (P037) </wsp:Policy>
1051 (P038) </sp:SignedEncryptedSupportingTokens>
1052 (P039) <sp:Wss11>
1053 (P040)   <wsp:Policy>
1054 (P041)     <sp:MustSupportRefKeyIdentifier/>
1055 (P042)     <sp:MustSupportRefIssuerSerial/>
1056 (P043)     <sp:MustSupportRefThumbprint/>
1057 (P044)     <sp:MustSupportRefEncryptedKey/>
1058 (P045)   </wsp:Policy>
1059 (P046) </sp:Wss11>
1060 (P047) </wsp:All>
1061 (P048) </wsp:ExactlyOne>
1062 (P049) </wsp:Policy>
1063
1064 (P050) <wsp:Policy wsu:Id="UsernameForCertificates_input_policy">
1065 (P051)   <wsp:ExactlyOne>
1066 (P052)   <wsp:All>
1067 (P053)     <sp:SignedParts>
1068 (P054)       <sp:Body/>
1069 (P055)     </sp:SignedParts>
1070 (P056)     <sp:EncryptedParts>
1071 (P057)       <sp:Body/>
```

```

1072 (P058)         </sp:EncryptedParts>
1073 (P059)         </wsp:All>
1074 (P060)         </wsp:ExactlyOne>
1075 (P061)         </wsp:Policy>
1076
1077 (P062)         <wsp:Policy wsu:Id="UsernameForCertificate_output_policy">
1078 (P063)         <wsp:ExactlyOne>
1079 (P064)         <wsp:All>
1080 (P065)         <sp:SignedParts>
1081 (P066)         <sp:Body/>
1082 (P067)         </sp:SignedParts>
1083 (P068)         <sp:EncryptedParts>
1084 (P069)         <sp:Body/>
1085 (P070)         </sp:EncryptedParts>
1086 (P071)         </wsp:All>
1087 (P072)         </wsp:ExactlyOne>
1088 (P073)         </wsp:Policy>

```

1089 Lines (P004) – (P297) contain a SymmetricBinding assertion which indicates the use of one token to both  
1090 sign and encrypt a message.

1091 Lines (P006) – (P015) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P012)  
1092 indicate that the protection token must be an X.509 token that must never be included in any messages in  
1093 the message exchange. Line (P010) dictates the X.509 token must be an X.509v3 security token as  
1094 described in the WS-Security 1.1 X.509 Token Profile. Line (P011) dicates a thumbprint reference must  
1095 be used to identify the token in any message.

1096 Line (P026) requires the inclusion of a timestamp.

1097 Lines (P030) – (P038) contain a SignedEncryptedSupportingTokens assertion which identifies the  
1098 inclusion of an additional token which must be included in the message signature and encrypted. Lines  
1099 (P032) – (P036) indicate that the supporting token must be a UsernameToken and must be included in all  
1100 messages to the recipient. Line (P034) dictates the UsernameToken must conform to the WS-Security 1.1  
1101 UsernameToken Profile.

1102 Lines (P040) – (P046) contain some WS-Security 1.1 related interoperability requirements, specifically  
1103 support for key identifier, issuer serial number, thumbprint, and encrypted key references.

1104 Lines (P050) – (P061) contain a policy that is attached to the input message. Lines (P053) – (P055)  
1105 require that the body of the input message must be signed. Lines (P056) – (P058) require the body of the  
1106 input message must be encrypted.

1107 Lines (P062) – (P073) contain a policy that is attached to the output message. Lines (P065) – (P067)  
1108 require that the body of the output message must be signed. Lines (P068) – (P070) require the body of  
1109 the output message must be encrypted.

1110 An example of an input message that conforms to the above stated policy is as follows.

```

1111 (M001) <?xml version="1.0" encoding="utf-8" ?>
1112 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1113 (M003) <soap:Header>
1114 (M004) <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..."
1115 xmlns:wsu="...">
1116 (M005) <xenc:EncryptedKey wsu:Id="EK">
1117 (M006) <xenc:EncryptionMethod
1118 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
1119 (M007) <ds:KeyInfo>
1120 (M008) <wsse:SecurityTokenReference>
1121 (M009) <wsse:KeyIdentifier ValueType="http://docs.oasis-
1122 open.org/wss/oasis-wss-soap-message-security-1.1#ThumbPrintSHA1">
1123 (M010) LKiQ/CmFrJDJqCLFcjlhIsmZ/+0=
1124 (M011) </wsse:KeyIdentifier>
1125 (M012) </wsse:SecurityTokenReference>
1126 (M013) </ds:KeyInfo>
1127 (M014) </xenc:EncryptedKey>
1128 (M015) <xenc:ReferenceList>
1129 (M016) <xenc:DataReference URI="#encUT"/>

```

```

1130 (M017) <xenc:DataReference URI="#encBody"/>
1131 (M018) </xenc:ReferenceList>
1132 (M019) <wsu:Timestamp wsu:Id="T0">
1133 (M020) <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1134 (M021) </wsu:Timestamp>
1135 (M022) <xenc:EncryptedData wsu:Id="encUT">
1136 (M023) <xenc:EncryptionMethod
1137 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1138 (M024) <ds:KeyInfo>
1139 (M025) <wsse:SecurityTokenReference>
1140 (M026) <wsse:Reference URI="#EK"/>
1141 (M027) </wsse:SecurityTokenReference>
1142 (M028) </ds:KeyInfo>
1143 (M029) <xenc:CipherData>
1144 (M030) <xenc:CipherValue>...</xenc:CipherValue>
1145 (M031) </xenc:CipherData>
1146 (M032) </xenc:EncryptedData>
1147 (M033) <ds:Signature>
1148 (M034) <ds:SignedInfo>...
1149 (M035) <ds:Reference URI="#T0">...</ds:Reference>
1150 (M036) <ds:Reference URI="#usernameToken">...</ds:Reference>
1151 (M037) <ds:Reference URI="#body">...</ds:Reference>
1152 (M038) </ds:SignedInfo>
1153 (M039) <ds:SignatureValue>HFLP...</ds:SignatureValue>
1154 (M040) <ds:KeyInfo>
1155 (M041) <wsse:SecurityTokenReference>
1156 (M042) <wsse:Reference URI="#EK"/>
1157 (M043) </wsse:SecurityTokenReference>
1158 (M044) </ds:KeyInfo>
1159 (M045) </ds:Signature>
1160 (M046) </wsse:Security>
1161 (M047) </soap:Header>
1162 (M048) <soap:Body wsu:Id="body">
1163 (M049) <xenc:EncryptedData wsu:Id="encBody">
1164 (M050) <xenc:EncryptionMethod
1165 Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
1166 (M051) <ds:KeyInfo>
1167 (M052) <wsse:SecurityTokenReference>
1168 (M053) <wsse:Reference URI="#EK"/>
1169 (M054) </wsse:SecurityTokenReference>
1170 (M055) </ds:KeyInfo>
1171 (M056) <xenc:CipherData>
1172 (M057) <xenc:CipherValue>...</xenc:CipherValue>
1173 (M058) </xenc:CipherData>
1174 (M059) </xenc:EncryptedData>
1175 (M060) </soap:Body>
1176 (M061) </soap:Envelope>

```

1177 Lines (M005) – (M014) contain the encrypted symmetric key as required by the use of the  
1178 SymmetricBinding assertion starting on line (P004) with an X509Token ProtectionToken assertion. Line  
1179 (M006) references the KwRsaOaep Asymmetric Key Wrap algorithm dictated by the Basic 256 Algorithm  
1180 Suite assertion on line (P018). Lines (M009) – (M011) hold a KeyIdentifier of the protection token used to  
1181 encrypt the symmetric key as required by the SymmetricBinding assertion. Because the ProtectionToken  
1182 assertion disallowed the token from being inserted into the message and instead required a thumbprint  
1183 reference, a thumbprint reference is included to identify the token.

1184 Line (M016) is an encryption data reference that references the encrypted supporting UsernameToken on  
1185 lines (M022) – (M032). The encryption was required by the SignedEncryptedSupportingTokens assertion  
1186 on line (P038). Line (M023) references the Aes256 Encryption algorithm dictated by the Basic 256  
1187 Algorithm Suite assertion on line (P018). The encrypted symmetric key is used to encrypt the  
1188 UsernameToken as referenced on line (M026).

1189 Line (M017) is an encryption data reference that references the encrypted body of the message on lines  
1190 (M049) – (M059). The encryption was required by the EncryptedParts assertion of the input message

1191 policy. The encrypted symmetric key is used to encrypt the UsernameToken as referenced on line  
 1192 (M053).

1193 Lines (M019) – (M021) contain a timestamp for the message as required by the IncludeTimestamp  
 1194 assertion.

1195 Lines (M033) – (M045) contain the message signature.

1196 Line (M035) indicates the message timestamp is included in the signature as required by the  
 1197 IncludeTimestamp assertion definition.

1198 Line (M036) indicates the supporting UsernameToken is included in the signature as required by the  
 1199 SignedSupportingTokens assertion. Because the token was encrypted its content prior to encryption is  
 1200 included below to better illustrate the reference.

```

1201 (M062) <wsse:UsernameToken wsu:Id="usernameToken">
1202 (M063) <wsse:Username>Chris</wsse:Username>
1203 (M064) <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
1204 200401-wss-username-token-profile-1.0#PasswordText ">sirhC</wsse:Password>
1205 (M065) </wsse:UsernameToken>
  
```

1206 Line (M037) indicates the message body is included in the signature as required by the SignedParts  
 1207 assertion of the input message policy.

1208 Line (M042) references the encrypted symmetric key for signing as dictated by the SymmetricBinding  
 1209 assertion.

## 1210 2.2 X.509 Token Authentication Scenario Assertions

### 1211 2.2.1 (WSS1.0) X.509 Certificates, Sign, Encrypt

1212 This use-case corresponds to the situation where both parties have X.509v3 certificates (and public-  
 1213 private key pairs). The requestor identifies itself to the service. The message exchange is integrity  
 1214 protected and encrypted.

1215 This modeled by use of an asymmetric security binding assertion.

1216 The message level policies in this and subsequent sections cover a different scope of the web service  
 1217 definition than the security binding level policy and so appear as separate policies and are attached at  
 1218 WSDL Message Policy Subject. These are shown below as input and output policies. Thus, we need a  
 1219 set of coordinated policies one with endpoint subject and two with message subjects to achieve this use  
 1220 case.

1221 The policies are as follows:

```

1222 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1223 (P002) <wsp:ExactlyOne>
1224 (P003) <wsp>All>
1225 (P004) <sp:AsymmetricBinding>
1226 (P005) <wsp:Policy>
1227 (P006) <sp:InitiatorToken>
1228 (P007) <wsp:Policy>
1229 (P008) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1230 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1231 (P009) <wsp:Policy>
1232 (P010) <sp:WssX509V3Token10/>
1233 (P011) </wsp:Policy>
1234 (P012) </sp:X509Token>
1235 (P013) </wsp:Policy>
1236 (P014) </sp:InitiatorToken>
1237 (P015) <sp:RecipientToken>
1238 (P016) <wsp:Policy>
1239 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1240 sx/ws-securitypolicy/200702/IncludeToken/Never">
1241 (P018) <wsp:Policy>
1242 (P019) <sp:WssX509V3Token10/>
1243 (P020) </wsp:Policy>
  
```

```

1244 (P021)         </sp:X509Token>
1245 (P022)         </wsp:Policy>
1246 (P023)         </sp:RecipientToken>
1247 (P024)         <sp:AlgorithmSuite>
1248 (P025)         <wsp:Policy>
1249 (P026)         <sp:Basic256/>
1250 (P027)         </wsp:Policy>
1251 (P028)         </sp:AlgorithmSuite>
1252 (P029)         <sp:Layout>
1253 (P030)         <wsp:Policy>
1254 (P031)         <sp:Strict/>
1255 (P032)         </wsp:Policy>
1256 (P033)         </sp:Layout>
1257 (P034)         <sp:IncludeTimestamp/>
1258 (P035)         <sp:OnlySignEntireHeadersAndBody/>
1259 (P036)         </wsp:Policy>
1260 (P037)         </sp:AsymmetricBinding>
1261 (P038)         <sp:Wss10>
1262 (P039)         <wsp:Policy>
1263 (P040)         <sp:MustSupportRefKeyIdentifier/>
1264 (P041)         </wsp:Policy>
1265 (P042)         </sp:Wss10>
1266 (P043)         </wsp:All>
1267 (P044)         </wsp:ExactlyOne>
1268 (P045) </wsp:Policy>
1269
1270 (P046) <wsp:Policy wsu:Id="WSS10Anonymous_with_Certificates_input_policy">
1271 (P047) <wsp:ExactlyOne>
1272 (P048) <wsp:All>
1273 (P049) <sp:SignedParts>
1274 (P050) <sp:Body/>
1275 (P051) </sp:SignedParts>
1276 (P052) <sp:EncryptedParts>
1277 (P053) <sp:Body/>
1278 (P054) </sp:EncryptedParts>
1279 (P055) </wsp:All>
1280 (P056) </wsp:ExactlyOne>
1281 (P057) </wsp:Policy>
1282
1283 (P058) <wsp:Policy wsu:Id="WSS10anonymous_with_certs_output_policy">
1284 (P059) <wsp:ExactlyOne>
1285 (P060) <wsp:All>
1286 (P061) <sp:SignedParts>
1287 (P062) <sp:Body/>
1288 (P063) </sp:SignedParts>
1289 (P064) <sp:EncryptedParts>
1290 (P065) <sp:Body/>
1291 (P066) </sp:EncryptedParts>
1292 (P067) </wsp:All>
1293 (P068) </wsp:ExactlyOne>
1294 (P069) </wsp:Policy>

```

1295 Lines (P004) – (P037) contain the AsymmetricBinding assertion which indicates that the initiator's token  
1296 must be used for the message signature and the recipient's token must be used for message encryption.

1297 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)  
1298 indicate that the initiator token must be an X.509 token that must be included with all messages sent to  
1299 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the  
1300 WS-Security 1.0 X.509 Token Profile.

1301 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)  
1302 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token  
1303 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to  
1304 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the  
1305 token in any messages where the token is used.

1306 Line (P034) requires the inclusion of a timestamp.

1307 Lines (P046) – (P057) contain a policy that is attached to the input message. Lines (P049) – (P051)

1308 require that the body of the input message must be signed. Lines (P052) – (P054) require the body of the

1309 input message must be encrypted.

1310 Lines (P058) – (P069) contain a policy that is attached to the output message. Lines (P061) – (P063)

1311 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of

1312 the output message must be encrypted.

1313 An example of an input message that conforms to the above stated policy is as follows.

```

1314 (M001) <?xml version="1.0" encoding="utf-8" ?>
1315 (M002) <soap:Envelope xmlns:soap="..." xmlns:xenc="..." xmlns:ds="...">
1316 (M003)   <soap:Header>
1317 (M004)     <wsse:Security soap:mustUnderstand="1" xmlns:wsse="..." xmlns:wsu="...">
1318 (M005)       <xenc:EncryptedKey >
1319 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1320 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1321 (M008)         </xenc:EncryptionMethod>
1322 (M009)         <ds:KeyInfo>
1323 (M010)           <wsse:SecurityTokenReference >
1324 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1325 (M012)             Value="...#X509SubjectKeyIdentifier">
1326 (M013)               MIGfMa0GCSq...
1327 (M014)             </wsse:KeyIdentifier>
1328 (M015)           </ds:KeyInfo>
1329 (M016)         <xenc:CipherData>
1330 (M017)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1331 (M018)         </xenc:CipherData>
1332 (M019)         <xenc:ReferenceList>
1333 (M020)           <xenc:DataReference URI="#encBody"/>
1334 (M021)         </xenc:ReferenceList>
1335 (M022)       </xenc:EncryptedKey>
1336 (M023)       <wsu:Timestamp wsu:Id="T0">
1337 (M024)         <wsu:Created>2001-09-13T08:42:00Z</wsu:Created>
1338 (M025)       </wsu:Timestamp>
1339 (M026)       <wsse:BinarySecurityToken wsu:Id="binaryToken" Value="...#X509v3"
1340 (M027)       EncodingType="...#Base64Binary">
1341 (M028)         MIIEZzCCA9CgAwIBAgIQEmtJZc0...
1342 (M029)       </wsse:BinarySecurityToken>
1343 (M030)       <ds:Signature>
1344 (M031)         <ds:SignedInfo>...
1345 (M032)           <ds:Reference URI="#T0">...</ds:Reference>
1346 (M033)           <ds:Reference URI="#body">...</ds:Reference>
1347 (M034)         </ds:SignedInfo>
1348 (M035)         <ds:SignatureValue>HFLP...</ds:SignatureValue>
1349 (M036)       <ds:KeyInfo>
1350 (M037)         <wsse:SecurityTokenReference>
1351 (M038)           <wsse:Reference URI="#binaryToken"/>
1352 (M039)         </wsse:SecurityTokenReference>
1353 (M040)       </ds:KeyInfo>
1354 (M041)     </ds:Signature>
1355 (M042)   </wsse:Security>
1356 (M043) </soap:Header>
1357 (M044) <soap:Body wsu:Id="body">
1358 (M045)   <xenc:EncryptedData wsu:Id="encBody">
1359 (M046)     <xenc:CipherData>
1360 (M047)       <xenc:CipherValue>...</xenc:CipherValue>
1361 (M048)     </xenc:CipherData>
1362 (M049)   </xenc:EncryptedData>
1363 (M050) </soap:Body>
1364 (M051) </soap:Envelope>

```

1365 Line (M019) is an encryption data reference that references the encrypted body of the message on lines

1366 (M043) – (M047). The encryption was required by the EncryptedParts assertion of the input message

1367 policy. Lines (M011) – (M013) hold a KeyIdentifier of the recipient’s token used to encrypt the body as  
 1368 required by the AsymmetricBinding assertion. Because the RecipientToken assertion disallowed the  
 1369 token from being inserted into the message, a KeyIdentifier is used instead of a reference to an included  
 1370 token.

1371 Lines (M022) – (M024) contain a timestamp for the message as required by the IncludeTimestamp  
 1372 assertion.

1373 Lines (M025) – (M027) contain the BinarySecurityToken holding the X.509v3 certificate of the initiator as  
 1374 required by the InitiatorToken assertion.

1375 Lines (M028) – (M039) contain the message signature.

1376 Line (M030) indicates the message timestamp is included in the signature as required by the  
 1377 IncludeTimestamp assertion definition.

1378 Line (M031) indicates the message body is included in the signature as required by the SignedParts  
 1379 assertion of the input message policy.

1380 Note that the initiator’s BinarySecurityToken is not included in the message signature as it was not  
 1381 required by policy.

1382 Lines (M035) – (M037) references the initiator’s BinarySecurityToken included in the message for  
 1383 identifying the key used for signing as dictated by the AsymmetricBinding assertion.

## 1384 **2.2.2 (WSS1.0) Mutual Authentication with X.509 Certificates, Sign, Encrypt**

1385 This scenario is based on WSS Interop, Scenario 3, [Web Services Security: Interop 1](#), Draft 06, Editor,  
 1386 Hal Lockhart, BEA Systems

1387 This use case corresponds to the situation where both parties have X.509v3 certificates (and public-  
 1388 private key pairs). The requestor wishes to identify itself to the service using its X.509 credential (strong  
 1389 authentication). The message exchange needs to be integrity protected and encrypted as well. The  
 1390 difference from previous use case is that the X509 token inserted by the client is included in the message  
 1391 signature (see <ProtectTokens />).

1392 The policy is as follows:

```

1393 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1394 (P002)   <wsp:ExactlyOne>
1395 (P003)     <wsp:All>
1396 (P004)       <sp:AsymmetricBinding>
1397 (P005)         <wsp:Policy>
1398 (P006)           <sp:InitiatorToken>
1399 (P007)             <wsp:Policy>
1400 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1401 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1402 (P009)                 <wsp:Policy>
1403 (P010)                   <sp:WssX509V3Token10/>
1404 (P011)                 </wsp:Policy>
1405 (P012)               </sp:X509Token>
1406 (P013)             </wsp:Policy>
1407 (P014)           </sp:InitiatorToken>
1408 (P015)         <sp:RecipientToken>
1409 (P016)           <wsp:Policy>
1410 (P017)             <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1411 sx/ws-securitypolicy/200702/IncludeToken/Never">
1412 (P018)               <wsp:Policy>
1413 (P019)                 <sp:WssX509V3Token10/>
1414 (P020)               </wsp:Policy>
1415 (P021)             </sp:X509Token>
1416 (P022)           </wsp:Policy>
1417 (P023)         </sp:RecipientToken>
1418 (P024)       <sp:AlgorithmSuite>
1419 (P025)         <wsp:Policy>
1420 (P026)           <sp:Basic256/>
1421 (P027)         </wsp:Policy>
  
```

```

1422 (P028)         </sp:AlgorithmSuite>
1423 (P029)         <sp:Layout>
1424 (P030)         <wsp:Policy>
1425 (P031)         <sp:Strict/>
1426 (P032)         </wsp:Policy>
1427 (P033)         </sp:Layout>
1428 (P034)         <sp:IncludeTimestamp/>
1429 (P035)         <sp:ProtectTokens />
1430 (P036)         <sp:OnlySignEntireHeadersAndBody/>
1431 (P037)         </wsp:Policy>
1432 (P038)         </sp:AsymmetricBinding>
1433 (P039)         <sp:Wss10>
1434 (P040)         <wsp:Policy>
1435 (P041)         <sp:MustSupportRefKeyIdentifier/>
1436 (P042)         </wsp:Policy>
1437 (P043)         </sp:Wss10>
1438 (P044)         </wsp:All>
1439 (P045)         </wsp:ExactlyOne>
1440 (P046) </wsp:Policy>
1441
1442 (P047) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1443 (P048) <wsp:ExactlyOne>
1444 (P049) <wsp:All>
1445 (P050) <sp:SignedParts>
1446 (P051) <sp:Body/>
1447 (P052) </sp:SignedParts>
1448 (P053) <sp:EncryptedParts>
1449 (P054) <sp:Body/>
1450 (P055) </sp:EncryptedParts>
1451 (P056) </wsp:All>
1452 (P057) </wsp:ExactlyOne>
1453 (P058) </wsp:Policy>
1454
1455 (P059) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1456 (P060) <wsp:ExactlyOne>
1457 (P061) <wsp:All>
1458 (P062) <sp:SignedParts>
1459 (P063) <sp:Body/>
1460 (P064) </sp:SignedParts>
1461 (P065) <sp:EncryptedParts>
1462 (P066) <sp:Body/>
1463 (P067) </sp:EncryptedParts>
1464 (P068) </wsp:All>
1465 (P069) </wsp:ExactlyOne>
1466 (P070) </wsp:Policy>

```

1467 Lines (P004) – (P038) contain the AsymmetricBinding assertion which indicates that the initiator’s token  
1468 must be used for the message signature and the recipient’s token must be used for message encryption.

1469 Lines (P006) – (P014) contain the InitiatorToken assertion. Within that assertion lines (P008) – (P012)  
1470 indicate that the initiator token must be an X.509 token that must be included with all messages sent to  
1471 the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as described in the  
1472 WS-Security 1.0 X.509 Token Profile.

1473 Lines (P015) – (P023) contain the RecipientToken assertion. Within that assertion lines (P017) – (P021)  
1474 dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0 X.509 Token  
1475 Profile, however as stated on line (P017) it must not be included in any message. Instead, according to  
1476 the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to identify the  
1477 token in any messages where the token is used.

1478 Line (P034) requires the inclusion of a timestamp.

1479 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the  
1480 token used to generate that signature.

1481 Lines (P047) – (P058) contain a policy that is attached to the input message. Lines (P050) – (P052)  
1482 require that the body of the input message must be signed. Lines (P053) – (P055) require the body of the  
1483 input message must be encrypted.

1484 Lines (P059) – (P070) contain a policy that is attached to the output message. Lines (P062) – (P064)  
1485 require that the body of the output message must be signed. Lines (P064) – (P066) require the body of  
1486 the output message must be encrypted.

1487 An example of an input message that conforms to the above stated policy is as follows.

```
1488 (M001) <?xml version="1.0" encoding="utf-8" ?>
1489 (M002) <soapenv:Envelope xmlns:soapenv="..."
1490 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="..." xmlns:xenc="..." ...>
1491 (M003)   <soapenv:Header>
1492 (M004)     <wsse:Security xmlns:wsse="..." xmlns:ds="..."
1493 soapenv:mustUnderstand="1">
1494 (M005)       <xenc:EncryptedKey >
1495 (M006)         <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgf1p">
1496 (M007)           <ds:DigestMethod Algorithm="...#sha1"/>
1497 (M008)         </xenc:EncryptionMethod>
1498 (M009)         <ds:KeyInfo>
1499 (M010)           <wsse:SecurityTokenReference >
1500 (M011)             <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1501 ValueTypes="...#X509SubjectKeyIdentifier">CuJd...=</wsse:KeyIdentifier>
1502 (M012)           </wsse:SecurityTokenReference>
1503 (M013)         </ds:KeyInfo>
1504 (M014)         <xenc:CipherData>
1505 (M015)           <xenc:CipherValue>Hyx...=</xenc:CipherValue>
1506 (M016)         </xenc:CipherData>
1507 (M017)         <xenc:ReferenceList>
1508 (M018)           <xenc:DataReference URI="#encBody"/>
1509 (M019)         </xenc:ReferenceList>
1510 (M020)       </xenc:EncryptedKey>
1511 (M021)       <wsu:Timestamp wsu:Id="Timestamp" >
1512 (M022)         <wsu:Created>2007-03-26T16:53:39Z</wsu:Created>
1513 (M023)       </wsu:Timestamp>
1514 (M024)       <wsse:BinarySecurityToken wsu:Id="bst" ValueType="...#X509v3"
1515 EncodingType="...#Base64Binary">MIID...=</wsse:BinarySecurityToken>
1516 (M025)       <ds:Signature>
1517 (M026)         <ds:SignedInfo>
1518 (M027)           <ds:CanonicalizationMethod Algorithm=".../xml-exc-c14n#" />
1519 (M028)           <ds:SignatureMethod Algorithm="...#rsa-sha1"/>
1520 (M029)           <ds:Reference URI="#Timestamp">
1521 (M030)             <ds:Transforms>... </ds:Transforms>
1522 (M031)             <ds:DigestMethod Algorithm="...#sha1"/>
1523 (M032)             <ds:DigestValue>+g0I...=</ds:DigestValue>
1524 (M033)           </ds:Reference>
1525 (M034)           <ds:Reference URI="#Body">...</ds:Reference>
1526 (M035)           <ds:Reference URI="#bst">...</ds:Reference>
1527 (M036)         </ds:SignedInfo>
1528 (M037)         <ds:SignatureValue>RRT...=</ds:SignatureValue>
1529 (M038)       </ds:Signature>
1530 (M039)       <wsse:SecurityTokenReference >
1531 (M040)         <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1532 ValueTypes="...#X509SubjectKeyIdentifier">Xeg5...=</wsse:KeyIdentifier>
1533 (M041)       </wsse:SecurityTokenReference>
1534 (M042)     </ds:KeyInfo>
1535 (M043)   </ds:Signature>
1536 (M044) </wsse:Security>
1537 (M045) </soapenv:Header>
1538 (M046) <soapenv:Body wsu:Id="Body" >
1539 (M047)   <xenc:EncryptedData Id="encBody" Type="...#Content"
1540 (M048)     MimeTypes="text/xml" Encoding="UTF-8" >
1541 (M049)     <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1542 (M050)     <xenc:CipherData>
```

```

1543 (M051) <xenc:CipherValue>W84fn...1</xenc:CipherValue>
1544 (M052) </xenc:CipherData>
1545 (M053) </xenc:EncryptedData>
1546 (M054) </soapenv:Body>
1547 (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.

### 1570 **2.2.2.1 (WSS1.0) Mutual Authentication, X.509 Certificates, Symmetric Encryption**

1571 This scenario is based on WSS Interop, Scenario 4, [Web Services Security: Interop 2](#).

1572 A common variation on the previous example is where X.509 Certificates are still used for authentication,  
1573 but a mutually agreed upon symmetric key is used for encryption.

1574 In this use case the policy is the same except that the InitiatorToken and RecipientToken are now each  
1575 just SignatureTokens.

1576 A second variation in this use case is that the mutually agreed upon symmetric encryption key is  
1577 characterized as an “IssuedToken” with a mutually agreed upon URI used to identify the out of band (not  
1578 included in the message) token, which is included in a SupportingTokens element.

1579 The policy is as follows:

```

1580 (P001) <wsp:Policy wsu:Id="wss10_anonymous_with_cert_policy" >
1581 (P002)   <wsp:ExactlyOne>
1582 (P003)   <wsp:All>
1583 (P004)     <sp:AsymmetricBinding>
1584 (P005)     <wsp:Policy>
1585 (P006)       <sp:InitiatorSignatureToken>
1586 (P007)       <wsp:Policy>
1587 (P008)         <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1588 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
1589 (P009)           <wsp:Policy>
1590 (P010)             <sp:WssX509V3Token10/>
1591 (P011)           </wsp:Policy>
1592 (P012)         </sp:X509Token>
1593 (P013)       </wsp:Policy>
1594 (P014)     </sp:InitiatorSignatureToken>
1595 (P015)     <sp:RecipientSignatureToken>

```

```

1596 (P016) <wsp:Policy>
1597 (P017) <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
1598 sx/ws-securitypolicy/200702/IncludeToken/Never">
1599 (P018) <wsp:Policy>
1600 (P019) <sp:WssX509V3Token10/>
1601 (P020) </wsp:Policy>
1602 (P021) </sp:X509Token>
1603 (P022) </wsp:Policy>
1604 (P023) </sp:RecipientSignatureToken>
1605 (P024) <sp:AlgorithmSuite>
1606 (P025) <wsp:Policy>
1607 (P026) <sp:Basic256/>
1608 (P027) </wsp:Policy>
1609 (P028) </sp:AlgorithmSuite>
1610 (P029) <sp:Layout>
1611 (P030) <wsp:Policy>
1612 (P031) <sp:Strict/>
1613 (P032) </wsp:Policy>
1614 (P033) </sp:Layout>
1615 (P034) <sp:IncludeTimestamp/>
1616 (P035) <sp:ProtectTokens />
1617 (P036) <sp:OnlySignEntireHeadersAndBody/>
1618 (P037) </wsp:Policy>
1619 (P038) </sp:AsymmetricBinding>
1620 (P039) <sp:SupportingTokens>
1621 (P040) <wsp:Policy>
1622 (P041) <sp:IssuedToken>
1623 (P042) <sp:Issuer>SomeMutuallyAgreedURI</sp:Issuer>
1624 (P043) <sp:RequireExternalReference/>
1625 (P044) </sp:IssuedToken>
1626 (P045) <sp:EncryptedParts>
1627 (P046) <sp:Body/>
1628 (P047) </sp:EncryptedParts>
1629 (P048) </wsp:Policy>
1630 (P049) </sp:SupportingTokens>
1631 (P050) <sp:Wss10>
1632 (P051) <wsp:Policy>
1633 (P052) <sp:MustSupportRefKeyIdentifier/>
1634 (P053) </wsp:Policy>
1635 (P054) </sp:Wss10>
1636 (P055) </wsp:All>
1637 (P056) </wsp:ExactlyOne>
1638 (P057) </wsp:Policy>
1639
1640 (P058) <wsp:Policy wsu:Id="WSS10Anonymous with Certificates_input_policy">
1641 (P059) <wsp:ExactlyOne>
1642 (P060) <wsp:All>
1643 (P061) <sp:SignedParts>
1644 (P062) <sp:Body/>
1645 (P063) </sp:SignedParts>
1646 (P064) <sp:EncryptedParts>
1647 (P065) <sp:Body/>
1648 (P066) </sp:EncryptedParts>
1649 (P067) </wsp:All>
1650 (P068) </wsp:ExactlyOne>
1651 (P069) </wsp:Policy>
1652
1653 (P070) <wsp:Policy wsu:Id="WSS10anonymous with certs_output_policy">
1654 (P071) <wsp:ExactlyOne>
1655 (P072) <wsp:All>
1656 (P073) <sp:SignedParts>
1657 (P074) <sp:Body/>
1658 (P075) </sp:SignedParts>
1659 (P076) <sp:EncryptedParts>

```

```

1660 (P077) <sp:Body/>
1661 (P078) </sp:EncryptedParts>
1662 (P079) </wsp:All>
1663 (P080) </wsp:ExactlyOne>
1664 (P081) </wsp:Policy>

```

1665 Lines (P004) – (P038) contain the AsymmetricBindingAssertion which indicates that the initiator’s token  
1666 must be used for the message signature and the recipient’s token must be used for message encryption.

1667 Lines (P006) – (P014) contain the InitiatorSignatureToken assertion. Within that assertion lines (P008) –  
1668 (P012) indicate that the initiator token must be an X.509 token that must be included with all messages  
1669 sent to the recipient. Line (P010) dictates the X.509 token must be an X.509v3 security token as  
1670 described in the WS-Security 1.0 X.509 Token Profile. By specifying that this is an  
1671 InitiatorSignatureToken, it will only be used to sign the message and not used for encryption.

1672 Lines (P015) – (P023) contain the RecipientSignatureToken assertion. Within that assertion lines (P017)  
1673 – (P021) dictate the recipient token must also be an X.509 token as described in the WS-Security 1.0  
1674 X.509 Token Profile, however as stated on line (P017) it must not be included in any message. Instead,  
1675 according to the MustSupportKeyRefIdentifier assertion on line (P040) a KeyIdentifier must be used to  
1676 identify the token in any messages where the token is used. By specifying that this is an  
1677 RecipientSignatureToken, it will only be used to sign the response and not used for encryption.

1678 Line (P034) requires the inclusion of a timestamp.

1679 Line (P035) requires token protection (ProtectTokens) which dictates that the signature must cover the  
1680 token or token reference associated with the generation of that signature.

1681 Lines (P039) – (P049) contain a SupportingTokens assertion that contains an IssuedToken (P041) –  
1682 (P044), which contains an Issuer element (P042) that identifies an explicit URI  
1683 (“SomeMutuallyAgreedURI”) that must be used to indicate what token will be used. (Since the content of  
1684 the IssuedToken element is specific to the Issuer, any mechanism can be used to identify the key, and in  
1685 this case the simplest method of identifying the issuer with the explicit key has been chosen only to  
1686 illustrate one possible method, but not to recommend as opposed to any other method.) Line (P043),  
1687 RequireExternalReference indicates that the IssuedToken requires a token that is referenced external to  
1688 the message. Lines (P045) – (P047) indicate that the token is to be used for encrypting parts of the  
1689 message, explicitly, line (P046), the Body of the message.

1690 Lines (P058) – (P069) contain a policy that is attached to the input message. Lines (P061) – (P063)  
1691 require that the body of the input message must be signed. Lines (P064) – (P066) require the body of the  
1692 input message must be encrypted.

1693 Lines (P070) – (P081) contain a policy that is attached to the output message. Lines (P073) – (P075)  
1694 require that the body of the output message must be signed. Lines (P076) – (P078) require the body of  
1695 the output message must be encrypted.

1696

1697 The following example message is derived from the WSS Interop2 document.

1698

```

1699 (M001) <?xml version="1.0" encoding="utf-8" ?>
1700 (M002) <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1701 (M003)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1702 (M004)   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1703 (M005)   <soap:Header>
1704 (M006)     <wsse:Security soap:mustUnderstand="1"
1705 (M007)       xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1706 (M008)       <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1707 (M009)         <xenc:DataReference URI="#enc" />
1708 (M010)       </xenc:ReferenceList>
1709 (M011)       <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1710 (M012)         wsu:Id="timestamp">
1711 (M013)         <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1712 (M014)       </wsu:Timestamp>
1713 (M015)       <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1714 (M016)         EncodingType="wsse:Base64Binary"
1715 (M017)         xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1716 (M018)         wsu:Id="myCert">MI...hk</wsse:BinarySecurityToken>

```

```

1717 (M019) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1718 (M020) <SignedInfo>
1719 (M021) <CanonicalizationMethod
1720 (M022)   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1721 (M023) <SignatureMethod
1722 (M024)   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1723 (M025) <Reference URI="#body">
1724 (M026)   <Transforms>
1725 (M027)     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1726 (M028)   </Transforms>
1727 (M029)   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1728 (M030)   <DigestValue>QTV...dw=</DigestValue>
1729 (M031) </Reference>
1730 (M032) <Reference URI="#myCert">
1731 (M033)   <Transforms>
1732 (M034)     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1733 (M035)   </Transforms>
1734 (M036)   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1735 (M037)   <DigestValue>XYZ...ab=</DigestValue>
1736 (M038) </Reference>
1737 (M039) <Reference URI="#timestamp">
1738 (M040)   <Transforms>
1739 (M041)     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1740 (M042)   </Transforms>
1741 (M043)   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1742 (M044)   <DigestValue>XYZ...ab=</DigestValue>
1743 (M045) </Reference>
1744 (M046) </SignedInfo>
1745 (M047) <SignatureValue>H+x0...gUw=</SignatureValue>
1746 (M048) <KeyInfo>
1747 (M049)   <wsse:SecurityTokenReference>
1748 (M050)     <wsse:Reference URI="#myCert" />
1749 (M051)   </wsse:SecurityTokenReference>
1750 (M052) </KeyInfo>
1751 (M053) </Signature>
1752 (M054) </wsse:Security>
1753 (M055) </soap:Header>
1754 (M056) <soap:Body wsu:Id="body"
1755 (M057)   xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1756 (M058)   <xenc:EncryptedData Id="enc"
1757 (M059)     Type="http://www.w3.org/2001/04/xmlenc#Content"
1758 (M060)     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1759 (M061)     <xenc:EncryptionMethod
1760 (M062)       Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc" />
1761 (M063)     <xenc:KeyInfo>
1762 (M064)       <xenc:KeyName>SomeMutuallyAgreedURI</xenc:KeyName>
1763 (M065)     </xenc:KeyInfo>
1764 (M066)     <xenc:CipherData>
1765 (M067)       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
1766 (M068)     </xenc:CipherData>
1767 (M069)   </xenc:EncryptedData>
1768 (M070) </soap:Body>
1769 (M071) </soap:Envelope>

```

1770 In the above example, the main point of attention is line (M064), where the KeyName of the  
1771 EncryptedData element is "SomeMutuallyAgreedURI". As indicated above the specific techniques used to  
1772 identify and apply this token are totally within agreed upon methods define by the mutual parties.

1773 Lines (M005) – (M055) contain the SOAP Header element, which contains the WS-Security header  
1774 (M006) – (M054).

1775 The ReferenceList (M008) – (M010) contains an internal reference (M009), "enc", identifying the  
1776 EncryptedData element Id, line (M058). This EncryptedData (M058) – (M069) inside the SOAP Body,  
1777 (M056) – (M070), was required to be encrypted by the policy (P058) – (P069) as described above.

1778 The Timestamp (M011) – (M014) is required by the policy line (P034).

1779 The BinarySecurityToken (M015) – (M018) contains the actual certificate used to sign the request as  
1780 required by the policy, (P008), IncludeToken/AlwaysToRecipient.

1781 The Signature (M019) – (M053) contains a SignedInfo (M020) – (M046) that identifies the "#body" (M025)  
1782 the "#myCert" (M032), and the "#timestamp" (M039) as the elements covered by the signature. The

1783 Reference (M032) – (M038) with URI="#myCert" that covers the BinarySecurityToken (M015) – (M018)  
1784 was required by ProtectTokens in the policy (P035).

1785 The KeyInfo (M048) – (M052) uses a SecurityTokenReference to point to the "myCert"  
1786 BinarySecurityToken.

1787 The EncryptedData (M058) – (M069) was described above, where it was pointed out that line (M064)  
1788 contains the external reference ("SomeMutuallyAgreedURI") to the mutually shared symmetric key used  
1789 for encryption.

## 1790 2.2.3 (WSS1.1) Anonymous with X.509 Certificate, Sign, Encrypt

1791 This scenario is based on the the "Examples of Secure Web Service Message Exchange Document"  
1792 [\[WS-SECURE-INTEROP\]](#) (see also sec 2.1.4)

1793 In this use case the Request is signed using DerivedKeyToken1(K), then encrypted using a  
1794 DerivedKeyToken2(K) where K is ephemeral key protected for the server's certificate. Response is signed  
1795 using DKT3(K), (if needed) encrypted using DKT4(K). The requestor does not wish to identify himself; the  
1796 message exchange is protected using derived symmetric keys. As a simpler, but less secure, alternative,  
1797 ephemeral key K (instead of derived keys) could be used for message protection by simply omitting the  
1798 sp:RequireDerivedKeys assertion.

1799 The policy is as follows:

```
1800 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">  
1801 (P002)   <wsp:ExactlyOne>  
1802 (P003)     <wsp:All>  
1803 (P004)       <sp:SymmetricBinding>  
1804 (P005)         <wsp:Policy>  
1805 (P006)           <sp:ProtectionToken>  
1806 (P007)             <wsp:Policy>  
1807 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-  
1808 sx/ws-securitypolicy/200702/IncludeToken/Never">  
1809 (P009)                 <wsp:Policy>  
1810 (P010)                   <sp:RequireDerivedKeys/>  
1811 (P011)                   <sp:RequireThumbprintReference/>  
1812 (P012)                   <sp:WssX509V3Token11/>  
1813 (P013)                 </wsp:Policy>  
1814 (P014)               </sp:X509Token>  
1815 (P015)             </wsp:Policy>  
1816 (P016)           </sp:ProtectionToken>  
1817 (P017)         <sp:AlgorithmSuite>  
1818 (P018)           <wsp:Policy>  
1819 (P019)             <sp:Basic256/>  
1820 (P020)           </wsp:Policy>  
1821 (P021)         </sp:AlgorithmSuite>  
1822 (P022)       <sp:Layout>  
1823 (P023)         <wsp:Policy>  
1824 (P024)           <sp:Strict/>  
1825 (P025)         </wsp:Policy>  
1826 (P026)       </sp:Layout>  
1827 (P027)     <sp:IncludeTimestamp/>  
1828 (P028)     <sp:OnlySignEntireHeadersAndBody/>  
1829 (P029)   </wsp:Policy>  
1830 (P030) </sp:SymmetricBinding>  
1831 (P031) <sp:Wss11>  
1832 (P032)   <wsp:Policy>  
1833 (P033)     <sp:MustSupportRefKeyIdentifier/>  
1834 (P034)     <sp:MustSupportRefIssuerSerial/>  
1835 (P035)     <sp:MustSupportRefThumbprint/>  
1836 (P036)     <sp:MustSupportRefEncryptedKey/>  
1837 (P037)     <sp:RequireSignatureConfirmation/>  
1838 (P038)   </wsp:Policy>  
1839 (P039) </sp:Wss11>  
1840 (P040) </wsp:All>
```

```

1841 (P041) </wsp:ExactlyOne>
1842 (P042) </wsp:Policy>
1843
1844 (P043) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_input_policy">
1845 (P044) <wsp:ExactlyOne>
1846 (P045) <wsp>All>
1847 (P046) <sp:SignedParts>
1848 (P047) <sp:Body/>
1849 (P048) </sp:SignedParts>
1850 (P049) <sp:EncryptedParts>
1851 (P050) <sp:Body/>
1852 (P051) </sp:EncryptedParts>
1853 (P052) </wsp>All>
1854 (P053) </wsp:ExactlyOne>
1855 (P054) </wsp:Policy>
1856
1857 (P055) <wsp:Policy wsu:Id=" WSS11_AnonymousForX509SignEncrypt_output_policy">
1858 (P056) <wsp:ExactlyOne>
1859 (P057) <wsp>All>
1860 (P058) <sp:SignedParts>
1861 (P059) <sp:Body/>
1862 (P060) </sp:SignedParts>
1863 (P061) <sp:EncryptedParts>
1864 (P062) <sp:Body/>
1865 (P063) </sp:EncryptedParts>
1866 (P064) </wsp>All>
1867 (P065) </wsp:ExactlyOne>
1868 (P066) </wsp:Policy>

```

1869 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token  
1870 must be used for both message signature and message encryption.

1871 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)  
1872 indicate that the ProtectionToken must be an X.509 token that must not be included with any message  
1873 sent between the Initiator and Recipient.

1874 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an  
1875 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the  
1876 MustSupportRefThumbprint assertion on line (P035) and RequireThumbprintReference on line (P011), a  
1877 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token  
1878 is used.

1879 Line (P027) requires the inclusion of a timestamp.

1880 Lines (P031) – (P039) contain some WS-Security 1.1 related interoperability requirements, specifically  
1881 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires  
1882 signature confirmation on the response.

1883 Lines (P043) – (P054) contain a policy that is attached to the input message. Lines (P045) – (P048)  
1884 require that the body of the input message must be signed. Lines (P049) – (P051) require the body of the  
1885 input message must be encrypted.

1886 Lines (P055) – (P066) contain a policy that is attached to the output message. Lines (P057) – (P060)  
1887 require that the body of the output message must be signed. Lines (P061) – (P063) require the body of  
1888 the output message must be encrypted.

1889 An example of an input message that conforms to the above stated policy is as follows.

1890 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,  
1891 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the  
1892 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-  
1893 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```

1894 (M001) <?xml version="1.0" encoding="utf-8" ?>
1895 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
1896 (M003) <env:Header>

```

```

1897 (M004) <wsse:Security xmlns:wsse="..." xmlns:wssell="..." xmlns:wsc="..."
1898 env:mustUnderstand="1">
1899 (M005) <xenc:EncryptedKey Id="encKey" >
1900 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
1901 (M007) <ds:DigestMethod Algorithm...#sha1" />
1902 (M008) </xenc:EncryptionMethod>
1903 (M009) <ds:KeyInfo >
1904 (M010) <wsse:SecurityTokenReference >
1905 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
1906 ValueType="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
1907 (M012) </wsse:SecurityTokenReference>
1908 (M013) </ds:KeyInfo>
1909 (M014) <xenc:CipherData>
1910 (M015) <xenc:CipherValue>TE...=</xenc:CipherValue>
1911 (M016) </xenc:CipherData>
1912 (M017) </xenc:EncryptedKey>
1913 (M018) <wsc:DerivedKeyToken Algorithm=".../p_sha1" wsu:Id="DKey1">
1914 (M019) wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1915 (M020) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1916 (M021) </wsse:SecurityTokenReference>
1917 (M022) <wsc:Generation>0</wsc:Generation>
1918 (M023) <wsc:Length>32</wsc:Length>
1919 (M024) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1920 (M025) <wsc:Nonce>39...=</wsc:Nonce>
1921 (M026) </wsc:DerivedKeyToken>
1922 (M027) <xenc:ReferenceList>
1923 (M028) <xenc:DataReference URI="#encBody"/>
1924 (M029) </xenc:ReferenceList>
1925 (M030) <wsc:DerivedKeyToken Algorithm=".../p_sha1" wsu:Id="DKey2">
1926 (M031) <wsse:SecurityTokenReference wssell:TokenType="...#EncryptedKey">
1927 (M032) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
1928 (M033) </wsse:SecurityTokenReference>
1929 (M034) <wsc:Generation>0</wsc:Generation>
1930 (M035) <wsc:Length>32</wsc:Length>
1931 (M036) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
1932 (M037) <wsc:Nonce>...=</wsc:Nonce>
1933 (M038) </wsc:DerivedKeyToken>
1934 (M039) <wsu:Timestamp wsu:Id="Timestamp" >
1935 (M040) <wsu:Created>2007-03-26T23:43:13Z</wsu:Created>
1936 (M041) </wsu:Timestamp>
1937 (M042) <ds:Signature >
1938 (M043) <ds:SignedInfo>
1939 (M044) ...
1940 (M045) <ds:Reference URI="#Timestamp">...</ds:Reference>
1941 (M046) <ds:Reference URI="#Body">...</ds:Reference>
1942 (M047) </ds:SignedInfo>
1943 (M048) <ds:SignatureValue>Yu...=</ds:SignatureValue>
1944 (M049) <ds:KeyInfo>
1945 (M050) <wsse:SecurityTokenReference >
1946 (M051) <wsse:Reference URI="#DKey2" ValueType=".../dk"/>
1947 (M052) </wsse:SecurityTokenReference>
1948 (M053) </ds:KeyInfo>
1949 (M054) </ds:Signature>
1950 (M055) </wsse:Security>
1951 (M056) </env:Header>
1952 (M057) <env:Body wsu:Id="Body" >
1953 (M058) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
1954 Encoding="UTF-8" >
1955 (M059) <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
1956 (M060) <ds:KeyInfo >
1957 (M061) <wsse:SecurityTokenReference >
1958 (M062) <wsse:Reference URI="#DKey1" ValueType=".../dk"/>
1959 (M063) </wsse:SecurityTokenReference>
1960 (M064) </ds:KeyInfo>
1961 (M065) <xenc:CipherData>
1962 (M066) <xenc:CipherValue>p53...f</xenc:CipherValue>
1963 (M067) </xenc:CipherData>
1964 (M068) </xenc:EncryptedData>
1965 (M069) </env:Body>
1966 (M070) </env:Envelope>

```

1967

1968 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1  
 1969 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.  
 1970 Because the ProtectionToken disallowed the token from being inserted into the message, a KeyIdentifier is  
 1971 used instead of a reference to an included token. In addition, Line (M011) the KeyIdentifier has the value  
 1972 type of #ThumbprintSHA1 for Thumbprint Reference, it is required by the policy line (P011) of the  
 1973 Thumbprint Reference.

1974 Lines (M018) – (M022) is the first wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines  
 1975 (M005) – (M017), which is referenced using the WSS1.1 #EncryptedKey SecurityTokenReference  
 1976 mechanism contained in lines (M019) – (M021), and used for body encryption and it is referenced on line  
 1977 (M062).

1978 Lines (M027) – (M029) is an encryption data reference that references the encrypted body of the  
 1979 message on lines (M058) – (M068). The encryption was required by the EncryptedParts assertion of the  
 1980 input message policy.

1981 Lines (M030) – (M038) is the second wsc:DerivedKeyToken. It is derived from the EncryptedKey of lines  
 1982 (M005) – (M017) and used for signature referenced on line (M051).

1983 Lines (M039) – (M041) contain a timestamp for the message as required by the IncludeTimestamp  
 1984 assertion.

1985 Lines (M042) – (M054) contain the message signature.

1986 Line (M045) indicates the message timestamp is included in the signature as required by the  
 1987 IncludeTimestamp assertion definition.

1988 Line (M046) indicates the message body is included in the signature as required by the SignedParts  
 1989 assertion of the input message policy.

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

1990 This scenario is based on the the “Examples of Secure Web Service Message Exchange Document”  
 1991 [WS-SECURE-INTEROP] (see also sec 2.1.4)  
 1992

1993 Client and server X509 certificates are used for client and server authorization respectively. Request is  
 1994 signed using K, then encrypted using K, K is ephemeral key protected for server's certificate. Signature  
 1995 corresponding to K is signed using client certificate. Response is signed using K, encrypted using K,  
 1996 encrypted key K is not included in response. Alternatively, derived keys can be used for each of the  
 1997 encryption and signature operations by simply adding an sp:RequireDerivedKeys assertion.

1998 The policy is as follows:

```

1999
2000 (P001) <wsp:Policy wsu:Id="WSS11_AnonymousForX509SignEncrypt_Policy">
2001 (P002)   <wsp:ExactlyOne>
2002 (P003)     <wsp:All>
2003 (P004)       <sp:SymmetricBinding>
2004 (P005)         <wsp:Policy>
2005 (P006)           <sp:ProtectionToken>
2006 (P007)             <wsp:Policy>
2007 (P008)               <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2008 sx/ws-securitypolicy/200702/IncludeToken/Never">
2009 (P009)                 <wsp:Policy>
2010 (P010)                   <sp:RequireDerivedKeys/>
2011 (P011)                   <sp:RequireThumbprintReference/>
2012 (P012)                   <sp:WssX509V3Token11/>
2013 (P013)                 </wsp:Policy>
2014 (P014)               </sp:X509Token>
2015 (P015)             </wsp:Policy>
2016 (P016)           </sp:ProtectionToken>
2017 (P017)         <sp:AlgorithmSuite>
2018 (P018)       <wsp:Policy>
2019 (P019)         <sp:Basic256/>
  
```

```

2020 (P020)         </wsp:Policy>
2021 (P021)         </sp:AlgorithmSuite>
2022 (P022)         <sp:Layout>
2023 (P023)         <wsp:Policy>
2024 (P024)         <sp:Strict/>
2025 (P025)         </wsp:Policy>
2026 (P026)         </sp:Layout>
2027 (P027)         <sp:IncludeTimestamp/>
2028 (P028)         <sp:OnlySignEntireHeadersAndBody/>
2029 (P029)         </wsp:Policy>
2030 (P030)         </sp:SymmetricBinding>
2031 (P031)         <sp:EndorsingSupportingTokens>
2032 (P032)         <wsp:Policy>
2033 (P033)         <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-
2034 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2035 (P034)         <wsp:Policy>
2036 (P035)         <sp:RequireThumbprintReference/>
2037 (P036)         <sp:WssX509V3Token11/>
2038 (P037)         </wsp:Policy>
2039 (P038)         </sp:X509Token>
2040 (P039)         </wsp:Policy>
2041 (P040)         </sp:EndorsingSupportingTokens>
2042 (P041)         <sp:Wss11>
2043 (P042)         <wsp:Policy>
2044 (P043)         <sp:MustSupportRefKeyIdentifier/>
2045 (P044)         <sp:MustSupportRefIssuerSerial/>
2046 (P045)         <sp:MustSupportRefThumbprint/>
2047 (P046)         <sp:MustSupportRefEncryptedKey/>
2048 (P047)         <sp:RequireSignatureConfirmation/>
2049 (P048)         </wsp:Policy>
2050 (P049)         </sp:Wss11>
2051 (P050)         </wsp:All>
2052 (P051)         </wsp:ExactlyOne>
2053 (P052)         </wsp:Policy>
2054
2055 (P053) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_input_policy">
2056 (P054) <wsp:ExactlyOne>
2057 (P055) <wsp:All>
2058 (P056) <sp:SignedParts>
2059 (P057) <sp:Body/>
2060 (P058) </sp:SignedParts>
2061 (P059) <sp:EncryptedParts>
2062 (P060) <sp:Body/>
2063 (P061) </sp:EncryptedParts>
2064 (P062) </wsp:All>
2065 (P063) </wsp:ExactlyOne>
2066 (P064) </wsp:Policy>
2067
2068 (P065) <wsp:Policy wsu:Id=" WSS11_X509DKSignEncrypt_output_policy">
2069 (P066) <wsp:ExactlyOne>
2070 (P067) <wsp:All>
2071 (P068) <sp:SignedParts>
2072 (P069) <sp:Body/>
2073 (P070) </sp:SignedParts>
2074 (P071) <sp:EncryptedParts>
2075 (P072) <sp:Body/>
2076 (P073) </sp:EncryptedParts>
2077 (P074) </wsp:All>
2078 (P075) </wsp:ExactlyOne>
2079 (P076) </wsp:Policy>

```

2080 Lines (P004) – (P030) contain the SymmetricBinding assertion which indicates that the derived key token  
2081 must be used for both message signature and message encryption.

2082 Lines (P007) – (P016) contain the ProtectionToken assertion. Within that assertion lines (P008) – (P014)  
2083 indicate that the initiator token must be an X.509 token that must not be included with all messages sent  
2084 between the recipient and Recipient.

2085 Line (P010) dictates the derived key is required. Line (P012) dictates the X.509 token must be an  
2086 X.509v3 security token as described in the WS-Security 1.1 X.509 Token Profile. According to the  
2087 MustSupportRefThumbprint assertion on line (P043) and RequireThumbprintReference on line (P011), a  
2088 Thumbprint Reference of KeyIdentifier must be used to identify the token in any messages where the token  
2089 is used.

2090 Line (P027) requires the inclusion of a timestamp.

2091 Lines (P031) – (P040) contain the EndorsingSupportingTokens assertion which indicates that the message  
2092 signature should be endorsed by client's X509 certificate on line (P033) – (P038). Line (P033)  
2093 IncludeToken=".../AlwaysToRecipient" indicates the endorsing is only required when the message is sent to recipient.  
2094 Line (P036) dictates the X.509 token must be an X.509v3 security token as described in the WS-Security  
2095 1.1 X.509 Token Profile. and RequireThumbprintReference on line (P035), a Thumbprint Reference of  
2096 KeyIdentifier must be used to identify the token in any messages where the token is used.

2097 Lines (P041) – (P049) contain some WS-Security 1.1 related interoperability requirements, specifically  
2098 support for key identifier, issuer serial number, thumbprint, encrypted key references, and requires  
2099 signature confirmation on the response.

2100 Lines (P053) – (P064) contain a policy that is attached to the input message. Lines (P055) – (P058)  
2101 require that the body of the input message must be signed. Lines (P059) – (P061) require the body of the  
2102 input message must be encrypted.

2103 Lines (P065) – (P076) contain a policy that is attached to the output message. Lines (P067) – (P070)  
2104 require that the body of the output message must be signed. Lines (P071) – (P073) require the body of  
2105 the output message must be encrypted.

2106 An example of an input message that conforms to the above stated policy is as follows.

2107 Note: this message uses WS-SecureConversation as a means to meet the requirements of the policy,  
2108 however, aside from using the wsc:DerivedKeyToken elements to meet the policy requirements for the  
2109 RequireDerivedKeys assertion (P010) the general protocol mechanisms described in WS-  
2110 SecureConversation for SecurityContextTokens are not explicitly demonstrated.

```
2111 (M001) <?xml version="1.0" encoding="utf-8" ?>
2112 (M002) <env:Envelope xmlns:env="..." xmlns:xenc="http..." xmlns:ds="..." xmlns:wsu="...">
2113 (M003) <env:Header>
2114 (M004) <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." env:mustUnderstand="1">
2115 (M005) <xenc:EncryptedKey Id="encKey" >
2116 (M006) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
2117 (M007) <ds:DigestMethod Algorithm="...#sha1" />
2118 (M008) </xenc:EncryptionMethod>
2119 (M009) <ds:KeyInfo >
2120 (M010) <wsse:SecurityTokenReference >
2121 (M011) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2122 (M012) Value="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
2123 (M013) </wsse:SecurityTokenReference>
2124 (M014) </ds:KeyInfo>
2125 (M015) <xenc:CipherData>
2126 (M016) <xenc:CipherValue>eI...=</xenc:CipherValue>
2127 (M017) </xenc:CipherData>
2128 (M018) </xenc:EncryptedKey>
2129 (M019) <wsse:BinarySecurityToken ValueType="...#X509v3"
2130 EncodingType="...#Base64Binary">MI...=</wsse:BinarySecurityToken>
2131 (M020) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_sha1"
2132 wsu:Id="derivedKeyToken1">
2133 (M021) <wsse:SecurityTokenReference wss11:TokenType="...#EncryptedKey" >
2134 (M022) <wsse:Reference ValueType="...#EncryptedKey" URI="#encKey"/>
2135 (M023) </wsse:SecurityTokenReference>
2136 (M024) <wsc:Generation>0</wsc:Generation>
2137 (M025) <wsc:Length>32</wsc:Length>
2138 (M026) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2139 (M027) <wsc:Nonce>+R...=</wsc:Nonce>
2140 (M028) </wsc:DerivedKeyToken>
```

```

2141 (M029) <wsu:Timestamp wsu:Id="Timestamp" >
2142 (M030) <wsu:Created>2007-03-31T04:27:21Z</wsu:Created>
2143 (M031) </wsu:Timestamp>
2144 (M032) <n1:ReferenceList xmlns:n1=".../xmlenc#">
2145 (M033) <n1:DataReference URI="#encBody"/>
2146 (M034) </n1:ReferenceList>
2147 (M035) <wsc:DerivedKeyToken xmlns:wsc=".../sc" Algorithm=".../p_shal"
2148 wsu:Id="derivedKeyToken2">
2149 (M036) <wsse:SecurityTokenReference wsse1:TokenType="...#EncryptedKey" >
2150 (M037) <wsse:Reference ValueType="...EncryptedKey" URI="#encKey"/>
2151 (M038) </wsse:SecurityTokenReference>
2152 (M039) <wsc:Generation>0</wsc:Generation>
2153 (M040) <wsc:Length>32</wsc:Length>
2154 (M041) <wsc:Label>WS-SecureConversationWS-SecureConversation</wsc:Label>
2155 (M042) <wsc:Nonce>wL...=</wsc:Nonce>
2156 (M043) </wsc:DerivedKeyToken>
2157 (M044) <ds:Signature Id="messageSignature">
2158 (M045) <ds:SignedInfo>
2159 (M046) ...
2160 (M047) <ds:Reference URI="#Timestamp">
2161 (M048) ...
2162 (M049) </ds:Reference>
2163 (M050) <ds:Reference URI="#Body">
2164 (M051) ...
2165 (M052) </ds:Reference>
2166 (M053) </ds:SignedInfo>
2167 (M054) <ds:SignatureValue>abcdefg</ds:SignatureValue>
2168 (M055) <ds:KeyInfo>
2169 (M056) <wsse:SecurityTokenReference >
2170 (M057) <wsse:Reference URI="#derivedKeyToken2" ValueType=".../dk"/>
2171 (M058) </wsse:SecurityTokenReference>
2172 (M059) </ds:KeyInfo>
2173 (M060) </ds:Signature>
2174 (M061) <ds:Signature >
2175 (M062) <ds:SignedInfo>
2176 (M063) ...
2177 (M064) <ds:Reference URI="#messageSignature">
2178 (M065) ...
2179 (M066) </ds:Reference>
2180 (M067) </ds:SignedInfo>
2181 (M068) <ds:SignatureValue>hijklmnop</ds:SignatureValue>
2182 (M069) <ds:KeyInfo>
2183 (M070) <wsse:SecurityTokenReference >
2184 (M071) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
2185 ValueType="...#ThumbprintSHA1">Pj...=</wsse:KeyIdentifier>
2186 (M072) </wsse:SecurityTokenReference>
2187 (M073) </ds:KeyInfo>
2188 (M074) </ds:Signature>
2189 (M075) </wsse:Security>
2190 (M076) </env:Header>
2191 (M077) <env:Body wsu:Id="Body" >
2192 (M078) <xenc:EncryptedData Id="encBody" Type="...#Content" MimeType="text/xml"
2193 Encoding="UTF-8" >
2194 (M079) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
2195 (M080) <ds:KeyInfo >
2196 (M081) <wsse:SecurityTokenReference >
2197 (M082) <wsse:Reference URI="#derivedKeyToken1" ValueType=".../dk"/>
2198 (M083) </wsse:SecurityTokenReference>
2199 (M084) </ds:KeyInfo>
2200 (M085) <xenc:CipherData>
2201 (M086) <xenc:CipherValue>70...=</xenc:CipherValue>
2202 (M087) </xenc:CipherData>
2203 (M088) </xenc:EncryptedData>
2204 (M089) </env:Body>
2205 (M090) </env:Envelope>
2206 (M091)

```

2207

2208 Lines (M005) – (M017) is the EncryptedKey information which will be reference using the WSS1.1  
2209 #EncryptedKey SecurityTokenReference function. Lines (M010) – (M012) is the security token reference.

2210 Lines (M010) – (M012) is the security token reference. Because the ProtectionToken disallowed the token  
 2211 from being inserted into the message, a KeyIdentifier is used instead of a reference to an included token.  
 2212 In addition, Line (M011) the KeyIdentifier has the value type of #ThumbprintSHA1 for Thumbprint  
 2213 Reference, and it is required by the policy line (P011) of the Thumbprint Reference.

2214 Line (M019) is an X509 BinarySecurityToken that contains the actual X509 certificate that is used by the  
 2215 endorsing signature described below. The token is required to be present in the message based on the  
 2216 line (P033) that requires this token for the message sent to the recipient.

2217 Lines (M020) – (M027) is the first derived key token. It is derived from the EncryptedKey of lines (M005) –  
 2218 (M017) and used for body encryption referenced on line (M081).

2219 Lines (M028) – (M030) contain a Timestamp element for the message as required by the  
 2220 IncludeTimestamp assertion.

2221 Lines (M031) – (M033) contain an encryption data reference that references the encrypted body of the  
 2222 message on lines (M077) – (M087). The encryption was required by the EncryptedParts assertion of the  
 2223 input message policy.

2224 Lines (M034) – (M042) is the second derived key token. It is derived from the EncryptedKey of lines  
 2225 (M005) – (M017) and used by the signature that references it on line (M056).

2226 Lines (M043) – (M059) contain the message signature. Lines (M054) – (M058) is its KeyInfo block that  
 2227 indicates the second derived key token should be used to verify the message signature.

2228 Line (M057) indicates the message timestamp is included in the signature as required by the  
 2229 IncludeTimestamp assertion definition.

2230 Line (M060) indicates the message body is included in the signature as required by the SignedParts  
 2231 assertion of the input message policy.

2232 Lines (M060) – (M073) contain the endorsing signature. It signs the message signature, referenced on  
 2233 line (M063), per EndorsingSupportingTokens assertion policy requirement on lines (P031) – (P040). Line  
 2234 (M070), the KeyIdentifier, has the value type of #ThumbprintSHA1 for Thumbprint Reference, and it is  
 2235 required by the policy line (P035) of the Thumbprint Reference. This Thumbprint Reference must match  
 2236 the Thumbprint of the X509 Certificate contained in the BinarySecurityToken on line (M019), based on the  
 2237 IncludeToken requirement line (P033).

2238

2239 An example of an output message that conforms to the above stated policy follows:

2240

```

2241 (R001) <env:Envelope xmlns:env="..." xmlns:wsu="...">
2242 (R002)   <env:Header>
2243 (R003)     <wsse:Security env:mustUnderstand="1" xmlns:wsse="...">
2244 (R004)       <wsu:Timestamp wsu:Id="Timestamp" >
2245 (R005)         <wsu:Created>2007-03-31T04:27:25Z</wsu:Created>
2246 (R006)       </wsu:Timestamp>
2247 (R007)     <wsc:DerivedKeyToken wsu:Id="derivedKeyToken1" xmlns:wsc=".../sc">
2248 (R008)       <wsse:SecurityTokenReference>
2249 (R009)         <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2250 (R010)           >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2251 (R011)       </wsse:SecurityTokenReference>
2252 (R012)     <wsc:Offset>0</wsc:Offset>
2253 (R013)     <wsc:Length>24</wsc:Length>
2254 (R014)     <wsc:Nonce>NfSNXZLYAA8mocQz19KWjg==</wsc:Nonce>
2255 (R015)     </wsc:DerivedKeyToken>
2256 (R016)     <wsc:DerivedKeyToken wsu:Id="derivedKeyToken2" xmlns:wsc=".../sc">
2257 (R017)       <wsse:SecurityTokenReference>
2258 (R018)         <wsse:KeyIdentifier ValueType="...1.1#EncryptedKeySHA1"
2259 (R019)           >nazB6DwNC9tcwFsgHoSYWXLf2wk=</wsse:KeyIdentifier>
2260 (R020)       </wsse:SecurityTokenReference>
2261 (R021)     <wsc:Nonce>1F/CtyQ9d1Ro8E3+uZYmgQ==</wsc:Nonce>
2262 (R022)     </wsc:DerivedKeyToken>
2263 (R023)     <enc:ReferenceList xmlns:enc=".../xmlenc#">
2264 (R024)       <enc:DataReference URI="#encBody"/>
2265 (R025)     </enc:ReferenceList>
2266 (R026)     <wsse11:SignatureConfirmation wsu:Id="sigconf1"
2267 (R027)       Value="abcdefg=" xmlns:wsse11="..."/>

```

```

2268 (R028) <wssell:SignatureConfirmation wsu:Id="sigconf2"
2269 Value="hijklmnop=" xmlns:wssell="..."/>
2270 (R030) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
2271 (R031) <SignedInfo>
2272 (R032) <CanonicalizationMethod
2273 (R033) Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2274 (R034) <SignatureMethod
2275 (R035) Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
2276 (R036) <Reference URI="#msgBody">
2277 (R037) ...
2278 (R038) </Reference>
2279 (R039) <Reference URI="#Timestamp">
2280 (R040) ...
2281 (R041) </Reference>
2282 (R042) <Reference URI="#sigconf1">
2283 (R043) ...
2284 (R044) </Reference>
2285 (R045) <Reference URI="#sigconf2">
2286 (R046) ...
2287 (R047) </Reference>
2288 (R048) </SignedInfo>
2289 (R049) <SignatureValue>3rAxsfJ2LjF7liRQX2EH/0DBmzE=</SignatureValue>
2290 (R050) <KeyInfo>
2291 (R051) <wsse:SecurityTokenReference>
2292 (R052) <wsse:Reference URI="#derivedKeyToken1" />
2293 (R053) </wsse:SecurityTokenReference>
2294 (R054) </KeyInfo>
2295 (R055) </Signature>
2296 (R056) </wsse:Security>
2297 (R057) </env:Header>
2298 (R058) <env:Body wsu:Id="msgBody">
2299 (R059) <enc:EncryptedData Id="encBody" Type="...xmlenc#Content"
2300 (R060) xmlns:enc=".../xmlenc#">
2301 (R061) <enc:EncryptionMethod Algorithm=".../xmlenc#aes256-cbc" />
2302 (R062) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
2303 (R063) <wsse:SecurityTokenReference xmlns:wsse="...">
2304 (R064) <wsse:Reference URI="#derivedKeyToken2" />
2305 (R065) </wsse:SecurityTokenReference>
2306 (R066) </KeyInfo>
2307 (R067) <enc:CipherData>
2308 (R068) <enc:CipherValue>y+eV...pu</enc:CipherValue>
2309 (R069) </enc:CipherData>
2310 (R070) </enc:EncryptedData>
2311 (R071) </env:Body>
2312 (R072) </env:Envelope>
2313

```

2314 Lines (R001)-(R072) contain the Response message returned to the Initiator.

2315 Lines (R004)-(R006) contain the Timestamp required by the Policy (P027).

2316 Lines (R007)-(R015) and (R016)-(R022) contain the information necessary for the Initiator to derive the  
2317 keys using the WS-SecureConversation shared secret, the identified key (R010) for DK1 and (R019) for  
2318 DK2, which reference the same key, and the Nonce (R014) for DK1 and (R021) for DK2, which are  
2319 different for the two derived keys.

2320 Lines (R023)-(R025) contain a ReferenceList that indicates the message Body (R058) contains the data  
2321 to be encrypted. The encryption was required by the output policy (P079).

2322 Lines (R026)-(R027) contain the SignatureConfirmation for the SignatureValue in the request message  
2323 (M054) which was required to be included by the policy (P047) RequireSignatureConfirmation.

2324 Lines (R028)-(R029) contain the SignatureConfirmation for the 2<sup>nd</sup> SignatureValue in the request  
2325 message (M068), which is also required by the policy (P047).

2326 Lines (R030)-(R055) contain the response message signature that covers the Timestamp element  
2327 (R039)->(R004) required to be signed by the policy (P027), the two SignatureConfirmation elements  
2328 (R042)->(R026) and (R045)->(R028), which are required to be signed because they are required  
2329 elements of the policy (P047), and the message Body (R036)->(R058), which is required to be signed by  
2330 the output message policy (P076). The signature may be verified using derivedKeyToken1 as indicated  
2331 on line (R052).

2332 Lines (R059)-(R070) contain the encrypted data as required by the policy (P079) and may be decrypted  
2333 using derivedKeyToken2 as indicated on line (R064).

## 2334 2.3 SAML Token Authentication Scenario Assertions

2335 For SAML, the combination of SAML and WSS version numbers is supported (WssSamlV11Token10,  
2336 WssSamlV11Token11, WssSamlV20Token11).

2337 Instead of explicitly including the SAML Assertion, a wsse:KeyIdentifier reference can also be used. To  
2338 enable this last behavior, the IncludeToken attribute is set to [http://docs.oasis-open.org/ws-sx/ws-](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never)  
2339 [securitypolicy/200702/IncludeToken/Never](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never).

2340 In all of the SAML scenarios, the SAML Assertion ConfirmationMethod expected to be used by the  
2341 Initiator is communicated implicitly by the context of the sp: security binding in use and type of sp:  
2342 element containing the SamlToken. There are 3 general SAML ConfirmationMethod use cases covered:  
2343 holder-of-key (hk), sender-vouches (sv), and bearer (br).

2344

2345 For the **holder-of-key** case, there is always a contained reference (or value) in the SAML  
2346 Assertion to key material that may be used for message protection in the scenario. The hk case  
2347 can be assumed if the WssSamlV\*\*Token\*\* element appears either in the sp:InitiatorToken  
2348 element of the sp:AsymmetricBinding element, or in the sp:ProtectionToken element of the  
2349 sp:SymmetricBinding element. In the sp:TransportBinding case, if the WssSamlV\*\*Token\*\*  
2350 element appears in an sp:EndorsingSupportingTokens or sp:SignedEndorsingSupportingTokens  
2351 element, the SAML Assertion type can also be assumed to be hk.

2352 The **sender-vouches** case can be assumed if the WssSamlV\*\*Token\*\* version will always  
2353 appear in an sp:SignedSupportingTokens element to indicate that the SAML Authority associated  
2354 with this token is the Initiator that signs the message.

2355 The **bearer** case can be assumed if the WssSamlV\*\*Token\*\* version appears in an  
2356 sp:SupportingTokens element (it may be signed as a SignedPart, but it is not involved in the  
2357 message protection).

2358

2359 It is recognized that other uses cases might exist, where these guidelines might need further elaboration  
2360 in order to address all contingencies, however that is outside the scope of the current version of this  
2361 document.

2362 Note: in the discussions below the term “SamlToken” or “SamlToken assertion” refers to the  
2363 policy requirement that a “SAML Assertion” be used as that token.

2364 Note: SAML Assertions have issuers. In addition, these Assertions are generally signed with an  
2365 X509 certificate. In general, the SAML IssuingAuthority may be regarded as the Subject of the  
2366 X509 certificate, which is trusted by a RelyingParty. In general, as well, there is usually a known  
2367 mapping between the Issuer identified within the SAML Assertion and the Subject of the X509  
2368 certificate used to sign that Assertion. It will be assumed in this document that the SAML  
2369 IssuingAuthority may be identified by either of these identities and that, in general, a RelyingParty  
2370 will check the details as necessary.

2371 The concept of the sp:“message signature” within the context of the **sp:TransportBinding** is not clearly  
2372 identified within the current version of [\[WS-SecurityPolicy\]](#), however, there are certain inferences that are  
2373 used in the use cases that follow that are identified here for reference.

2374 • Based on the diagrams in section 8 of [\[WS-SecurityPolicy\]](#), which show messages with a  
2375 “message signature” followed by a diagram showing use of transport security the only difference  
2376 is that the message signature diagrams contain a block labelled “Sig1”, which is the message  
2377 signature, and the transport security diagrams do not have this block.

2378 • Considering the fact that when [\[SSL\]](#) Client Certificates are used for client authentication, that the  
2379 client uses the client certificate private key to sign hash of SSL handshake messages in an SSL  
2380 CertificateVerify message that is part of the SSL protocol, the assumption is made that  
2381 subsequent data sent by the client on this link is effectively protected for the purposes of data

- 2382 integrity and confidentiality, and that the data sent on the link is authorized to be sent by the  
2383 holder of the private key associated with the client certificate.
- 2384 • Based on the considerations in the bullets above, and with intention of maintaining functional  
2385 consistency with the WS-SecurityPolicy notions of sp: message signature,  
2386 sp:SignedSupportingTokens, and sp:SignedEndorsingSupportingTokens, use of client certificates  
2387 with SSL will be considered equivalent to having a “virtual message signature” provided by the  
2388 Initiator’s client certificate which covers all the data in the SOAP request that the Initiator sends  
2389 on the SSL link.
  - 2390 • As such, in the above context, the client certificate may be regarded as providing both a virtual  
2391 Signing function for tokens and Timestamp that appear in the wsse:Security header, as well as a  
2392 virtual Endorsing function for tokens that contain the client certificate or a reference to the client  
2393 certificate that appear in the wsse:Security header.

2394 The net effect of the above assumptions for the SAML use cases in this section that use the  
2395 **sp:TransportBinding** is that if a **client certificate is required** to be used with the **sp:HttpsToken**  
2396 assertion then:

- 2397 1. A SAML **sender-vouches** Assertion contained in a wsse:Security header block may be  
2398 considered to be an **sp:SignedSupportingToken** when used in an sp:TransportBinding  
2399 using an sp:HttpsToken assertion that contains an sp:RequireClientCertificate assertion,  
2400 because the client certificate is being used as the IssuingAuthority behind the SAML  
2401 sender-vouches Assertion, which requires the IssuingAuthority to sign the combined  
2402 message and Assertion.
- 2403 2. A SAML **holder-of-key** Assertion contained in a wsse:Security header block may be  
2404 considered to be an **sp:SignedEndorsingSupportingToken** when used in an  
2405 sp:TransportBinding using an sp:HttpsToken assertion that contains an  
2406 sp:RequireClientCertificate assertion AND that the either the client certificate or a  
2407 reference to it is contained in the saml:SubjectConfirmationData/ds:KeyInfo element of  
2408 the SAML holder-of-key Assertion.
- 2409 3. A SAML **bearer** Assertion contained in a wsse:Security header block may only be  
2410 considered to be an **sp:SupportingToken**, because they are only “incidentally” covered  
2411 by the virtual message signature as a “pass through” token provided by the Requestor to  
2412 be evaluated by the Recipient that is being “passed through” by the Initiator, but which  
2413 the Initiator takes no responsibility.

2414 Ultimately, the responsibilities associated with the granting access to resources is determined by the  
2415 agreements between RelyingParties and IssuingAuthorities. Those agreements need to take into  
2416 consideration the security characteristics of the bindings which support access requests as to what kind  
2417 of bindings are required to “adequately protect” the requests for the associated business purposes. These  
2418 examples are intended to show how SAML Assertions can be used in a variety of WS-SecurityPolicy  
2419 contexts and how the different SAML token types (hk, sv, bearer) may be used in different configurations  
2420 relating the IssuingAuthority, the Requestor, the Initiator, the Recipient, and the RelyingParty.

## 2421 2.3.1 WSS 1.0 SAML Token Scenarios

### 2422 2.3.1.1 (WSS1.0) SAML1.1 Assertion (Bearer)

2423 Initiator adds a SAML assertion (bearer) representing the Requestor to the SOAP security header.  
2424 Since the SamlToken is listed in the SupportingTokens element, it will not explicitly be covered by a  
2425 message signature. The Initiator may infer that a Saml Bearer Assertion is acceptable to meet this  
2426 requirement, because the Initiator is not required to explicitly cover a SupportingToken with a signature.  
2427 The SAML assertion itself could be signed. The IssuingAuthority in this scenario is the Issuer (and signer,  
2428 if the Assertion is signed) of the SAML Assertion. The Initiator simply passes the token through and is not  
2429 actively involved in the trust relationship between the IssuingAuthority that issued the SAML Assertion  
2430 and the Requestor who is the Subject of the SAML Assertion.

2431 This scenario might be used in a SAML Federation application where a browser-based user with a SAML  
2432 Assertion indicating that user's SSO (Single Sign On) credential has submitted a request to a portal using  
2433 the Assertion as a credential (either directly or indirectly via a SAML Artifact [SAML11]), and the portal as  
2434 Initiator is generating a web service request on behalf of this user, but the trust that the Recipient has for  
2435 the Requestor is based on the Assertion and its IssuingAuthority, not the Initiator who delivered the  
2436 request.

2437

```
2438 (P001) <wsp:Policy>  
2439 (P002)   <sp:SupportingTokens>  
2440 (P003)   <wsp:Policy>  
2441 (P004)     <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
2442         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
2443 (P005)       <wsp:Policy>  
2444 (P006)         <sp:WssSamlV11Token10/>  
2445 (P007)       </wsp:Policy>  
2446 (P008)     </sp:SamlToken>  
2447 (P009)   </wsp:Policy>  
2448 (P010) </sp:SupportingTokens>  
2449 (P011) </wsp:Policy>
```

2450

2451 Lines (P001)-(P011) contain the endpoint wsp:Policy, which contains no bindings because none is  
2452 required to access the service protected by this policy, however the service does require that the  
2453 Requestor provide a Supporting Token in the sp:SupportingTokens element (P002)-(P010), which must  
2454 be an sp:SamlToken (P004)-(P008), which must be an sp:WssSamlV11Token10 (P006), which means  
2455 that the SamlToken must be a SAML 1.1 saml:Assertion that is sent in compliance with the  
2456 [WSS10-SAML11-PROFILE].

2457 As explained in section 2.3 above, the fact that the sp:SamlToken is simply in an sp:SupportingTokens  
2458 element indicates to the Initiator that a SAML bearer Assertion is what is expected to accompany the  
2459 request.

2460 The following is a sample request taken from [WSS11-SAML1120-PROFILE] that complies with the  
2461 WSS 1.0 compatible policy above

2462

```
2463 (M001) <S12:Envelope xmlns:S12="...">  
2464 (M002)   <S12:Header>  
2465 (M003)     <wsse:Security xmlns:wsse="...">  
2466 (M004)       <saml:Assertion xmlns:saml="..."  
2467 (M005)         AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"  
2468 (M006)         IssueInstant="2003-04-17T00:46:02Z"  
2469 (M007)         Issuer="www.opensaml.org"  
2470 (M008)         MajorVersion="1"  
2471 (M009)         MinorVersion="1">  
2472 (M010)       <saml:AuthenticationStatement>  
2473 (M011)         <saml:Subject>  
2474 (M012)           <saml:NameIdentifier  
2475 (M013)             NameQualifier="www.example.com"  
2476 (M014)             Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">  
2477 (M015)               uid=joe,ou=people,ou=saml-demo,o=baltimore.com  
2478 (M016)             </saml:NameIdentifier>  
2479 (M017)           <saml:SubjectConfirmation>  
2480 (M018)             <saml:ConfirmationMethod>  
2481 (M019)               urn:oasis:names:tc:SAML:1.0:cm:bearer  
2482 (M020)             </saml:ConfirmationMethod>  
2483 (M021)           </saml:SubjectConfirmation>  
2484 (M022)         </saml:Subject>  
2485 (M023)       </saml:AuthenticationStatement>  
2486 (M024)     </saml:Assertion>  
2487 (M025)   </wsse:Security>  
2488 (M026) </S12:Header>  
2489 (M027) <S12:Body>
```

```
2490 (M028) . . .
2491 (M029) </S12:Body>
2492 (M030) </S12:Envelope>
```

2493

2494 Lines (M001)-(M030) contains the SOAP:Envelope, which contains the SOAP:Header (M002)-(M026)  
2495 and the SOAP:Body (M027)-(M029).

2496 The SOAP Header contains a wsse:Security header block (M003)-(M025) which simply contains the  
2497 saml:Assertion.

2498 The saml:Assertion (M004)-(M024) is Version 1.1 (M008)-(M009), which is required by the policy  
2499 sp:WssSamlV11Token10 assertion (P006). The Assertion contains a saml:AuthenticationStatement  
2500 (M010)-(M023) that contains a saml:NameIdentifier (M012)-(M016) that identifies the saml:Subject, who is  
2501 the Requestor (who submitted the request from a browser) and it contains a saml:SubjectConfirmation  
2502 element (M017)-(M021), which specifies the saml:ConfirmationMethod to be "bearer" (M019), which  
2503 meets the policy requirement (P006).

2504 For general context to relate this scenario to Figure 1, the Requestor at a browser will have obtained  
2505 either the saml:Assertion above or an Artifact identifying that Assertion from an IssuingAuthority and sent  
2506 it to the portal in the context of some request the Requestor is trying to make. The portal recognizes that  
2507 to meet the needs of this request that it must invoke a web service that has publicized the policy above  
2508 (P001)-(P011). Therefore, the portal will now take on the role of Initiator (Fig 1) and assemble a SOAP  
2509 request (M001)-(M030) and submit it to the service as described in detail above.

### 2510 **2.3.1.2 (WSS1.0) SAML1.1 Assertion (Sender Vouches) over SSL**

2511 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #2](#)  
2512 section 4.4.4]

2513 Initiator adds a SAML Assertion (sv) to the SOAP Security Header. Because the TransportBinding  
2514 requires a Client Certificate AND the SAML token is in a SignedSupportingTokens element, when the  
2515 Initiator uses the Client Certificate to protect the message under SSL, the Initiator may be considered as  
2516 effectively "signing" the SAML sv Assertion and acting as a SAML Authority (i.e. the issuer of the sv  
2517 Assertion). By including the sv assertion in the header and using the Client Certificate to protect the  
2518 message, the Initiator takes responsibility for binding the Requestor, who is the Subject of the Assertion  
2519 to the contents of the message.

2520 Note: because SSL does not retain cryptographic protection of the message after the message is  
2521 delivered, messages protected using only this mechanism cannot be used as the basis for  
2522 non-repudiation.

2523

```
2524 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2525 wsu:Id="Wss10SamlSvV11Tran policy">
2526 (P002) <sp:TransportBinding>
2527 (P003) <wsp:Policy>
2528 (P004) <sp:TransportToken>
2529 (P005) <wsp:Policy>
2530 (P006) <sp:HttpsToken>
2531 (P007) <wsp:Policy>
2532 (P008) <sp:RequireClientCertificate>
2533 (P009) </wsp:Policy>
2534 (P010) </sp:HttpsToken>
2535 (P011) </wsp:Policy>
2536 (P012) </sp:TransportToken>
2537 (P013) <sp:AlgorithmSuite>
2538 (P014) <wsp:Policy>
2539 (P015) <sp:Basic256 />
2540 (P016) </wsp:Policy>
2541 (P017) </sp:AlgorithmSuite>
2542 (P018) <sp:Layout>
2543 (P019) <wsp:Policy>
2544 (P020) <sp:Strict />
```

```

2545 (P021) </wsp:Policy>
2546 (P022) </sp:Layout>
2547 (P023) <sp:IncludeTimestamp />
2548 (P024) </wsp:Policy>
2549 (P025) </sp:TransportBinding>
2550 (P026) <sp:SignedSupportingTokens>
2551 (P027) <wsp:Policy>
2552 (P028) <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
2553 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2554 (P029) <wsp:Policy>
2555 (P030) <sp:WssSamlV11Token10/>
2556 (P031) </wsp:Policy>
2557 (P032) </sp:SamlToken>
2558 (P033) </wsp:Policy>
2559 (P034) </sp:SignedSupportingTokens>
2560 (P035) </wsp:Policy>

```

2561

2562 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
2563 by a secure transport protocol such as SSL or TLS.

2564 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
2565 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
2566 transport token using the Basic256 algorithm suite (P015).

2567 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of  
2568 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section  
2569 4.3.1].

2570 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)  
2571 since there are no cryptographic tokens required to be present in the WS-Security header.

2572 A timestamp (P023) is required to be included in an acceptable message.

2573 Lines (P026)-(P034) contain a SignedSupportingTokens assertion, which indicates that the referenced  
2574 token is effectively "signed" by the combination of usage of the client certificate for SSL authentication  
2575 and the cryptographic protection of SSL. However, the "signed" characteristic will not be present after the  
2576 message is delivered from the transport layer to the recipient.

2577 Lines (P028)-(P032) indicate the signed token is a SAML Assertion (sender-vouches as described above  
2578 in section 2.3) and on line (P030) that it is a SAML 1.1 Assertion and that the WS-Security 1.0 SAML  
2579 Profile [[WSS10-SAML11-PROFILE](#)] is being used.

2580 This scenario is based on first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP](#) "Scenario #2 -  
2581 Sender-Vouches: Unsigned: SSL" section 4.4.4]

2582 Here is the example request from that scenario:

```

2583 (M001) <?xml version="1.0" encoding="utf-8" ?>
2584 (M002) <soap:Envelope
2585 (M003)   xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
2586 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2587 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2588 (M006)   xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401
2589 (M007) -wss-wssecurity-secext-1.0.xsd"
2590 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss
2591 (M009) -wssecurity-utility-1.0.xsd">
2592 (M010) <soap:Header>
2593 (M011)   <wss:Security soap:mustUnderstand="1">
2594 (M012)     <wsu:Timestamp>
2595 (M013)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2596 (M014)     </wsu:Timestamp>
2597 (M015)     <saml:Assertion
2598 (M016)       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2599 (M017)       MajorVersion="1" MinorVersion="1"
2600 (M018)       AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
2601 (M019)       Issuer=www.opensaml.org

```

```

2602 (M020) IssueInstant="2002-06-19T16:58:33.173Z">
2603 (M021) <saml:Conditions
2604 (M022) NotBefore="2002-06-19T16:53:33.173Z"
2605 (M023) NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2606 (M024) <saml:AuthenticationStatement
2607 (M025) AuthenticationMethod=
2608 (M026) "urn:oasis:names:tc:SAML:1.0:am:password"
2609 (M027) AuthenticationInstant="2002-06-19T16:57:30.000Z">
2610 (M028) <saml:Subject>
2611 (M029) <saml:NameIdentifier
2612 (M030) NameQualifier=www.example.com
2613 (M031) Format="">
2614 (M032) uid=joe,ou=people,ou=saml-demo,o=example.com
2615 (M033) </saml:NameIdentifier>
2616 (M034) <saml:SubjectConfirmation>
2617 (M035) <saml:ConfirmationMethod>
2618 (M036) urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2619 (M037) </saml:ConfirmationMethod>
2620 (M038) </saml:SubjectConfirmation>
2621 (M039) </saml:Subject>
2622 (M040) </saml:AuthenticationStatement>
2623 (M041) </saml:Assertion>
2624 (M042) </wsse:Security>
2625 (M043) </soap:Header>
2626 (M044) <soap:Body>
2627 (M045) <Ping xmlns="http://xmlsoap.org/Ping">
2628 (M046) <text>EchoString</text>
2629 (M047) </Ping>
2630 (M048) </soap:Body>
2631 (M049) </soap:Envelope>

```

2632

2633 Lines (M011)-(M042) contain the WS-Security 1.0 header required by the WssSamlV11Token10  
2634 assertion (P030).

2635 Lines (M012)-(M014) contain the wsu:Timestamp required by IncludeTimestamp assertion (P023).

2636 Lines (M015)-(M041) contain the SAML 1.1 Assertion required by the WssSamlV11Token10 assertion  
2637 (P030).

2638 Note that the additional requirements identified in the policy above are met by the SSL transport  
2639 capabilities and do not appear in any form in the SOAP message (M001)-(M049).

### 2640 **2.3.1.3 (WSS1.0) SAML1.1 Assertion (HK) over SSL**

2641 Initiator adds a SAML assertion (hk) to the SOAP Security Header. Because the TransportBinding  
2642 requires a Client Certificate AND the SAML token is in an EndorsingSupportingTokens element, the  
2643 Initiator may be considered to be authorized by the issuer of the hk SAML assertion to bind message  
2644 content to the Subject of the assertion. If the Client Certificate matches the certificate identified in the hk  
2645 assertion, the Initiator may be regarded as executing SAML hk responsibility of binding the Requestor,  
2646 who would be the Subject of the hk assertion, to the content of the message.

2647 In this scenario, the IssuingAuthority is the issuer(signer) of the hk SAML Assertion. The Requestor is the  
2648 Subject of the Assertion and the Initiator is authorized by the Authority to bind the Assertion to the  
2649 message using the ClientCertificate identified in the SAML Assertion, which should also be used to sign  
2650 the timestamp of the message with a separate Signature included in the WS-Security header.

2651

```

2652 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2653 (P002) wsu:Id="Wss10SamlHkV11Tran_policy">>
2654 (P003) <sp:TransportBinding>
2655 (P004) <wsp:Policy>
2656 (P005) <sp:TransportToken>
2657 (P006) <wsp:Policy>
2658 (P007) <sp:HttpsToken>

```

```

2659 (P008)         <wsp:Policy>
2660 (P009)         <sp:RequireClientCertificate>
2661 (P010)         </wsp:Policy>
2662 (P011)         </sp:HttpsToken>
2663 (P012)         </wsp:Policy>
2664 (P013)         </sp:TransportToken>
2665 (P014)         <sp:AlgorithmSuite>
2666 (P015)         <wsp:Policy>
2667 (P016)         <sp:Basic256 />
2668 (P017)         </wsp:Policy>
2669 (P018)         </sp:AlgorithmSuite>
2670 (P019)         <sp:Layout>
2671 (P020)         <wsp:Policy>
2672 (P021)         <sp:Strict />
2673 (P022)         </wsp:Policy>
2674 (P023)         </sp:Layout>
2675 (P024)         <sp:IncludeTimestamp />
2676 (P025)         </wsp:Policy>
2677 (P026)         </sp:TransportBinding>
2678 (P027)         <sp:SignedEndorsingSupportingTokens>
2679 (P028)         <wsp:Policy>
2680 (P029)         <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2681 open.org/ws-sx/ws-
2682 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2683 (P030)         <wsp:Policy>
2684 (P031)         <sp:WssSamlV11Token10/>
2685 (P032)         </wsp:Policy>
2686 (P033)         </sp:SamlToken>
2687 (P034)         </wsp:Policy>
2688 (P035)         </sp:SignedEndorsingSupportingTokens>
2689 (P036)         <wsp:Policy>

```

2690

2691 Section 2.3.2.3 contains an example message that is similar to one that could be used for the policy  
2692 above, except the Signed Endorsing Supporting Token there is a SAML Version 2.0 token, and a SAML  
2693 Version 1.1 token would be required here.

#### 2694 **2.3.1.4 (WSS1.0) SAML1.1 Sender Vouches with X.509 Certificates, Sign, Optional** 2695 **Encrypt**

2696 This scenario is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #3](#)].

2697 In this case, the SAML token is included as part of the message signature and sent only to the Recipient.  
2698 The message security is provided using X.509 certificates with both requestor and service having  
2699 exchanged these credentials via some out of band mechanism, which provides the basis of trust of each  
2700 others' certificates.

2701 In this scenario the SAML Authority is the Initiator who uses the message signature to both provide the  
2702 integrity of the message and to establish the Initiator as the SAML Authority based on the X509 certificate  
2703 used to sign the message and the SignedSupportingTokens SAML Assertion. Effectively, the SAML  
2704 Assertion is being "issued" as part of the message construction process. The Requestor is the Subject of  
2705 the SAML Assertion. The Initiator knows that the Recipient is expecting it to be the SAML Authority by the  
2706 fact that the policy specifies that the message requires a SignedSupportingTokens SamlToken.

2707

```

2708 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."
2709 (P002) wsu:Id="Wss10SamlSvV11Asymm_endpoint_policy">
2710 (P003) <wsp:ExactlyOne>
2711 (P004) <wsp:All>
2712 (P005) <sp:AsymmetricBinding>
2713 (P006) <wsp:Policy>
2714 (P007) <sp:InitiatorToken>
2715 (P008) <wsp:Policy>

```

```

2716 (P009) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2717 open.org/ws-sx/ws-
2718 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2719 <wsp:Policy>
2720 (P011) <sp:WssX509V3Token10/>
2721 (P012) </wsp:Policy>
2722 (P013) </sp:X509Token>
2723 (P014) </wsp:Policy>
2724 (P015) </sp:InitiatorToken>
2725 (P016) <sp:RecipientToken>
2726 (P017) <wsp:Policy>
2727 (P018) <sp:X509Token sp:IncludeToken="http://docs.oasis-
2728 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
2729 (P019) <wsp:Policy>
2730 (P020) <sp:WssX509V3Token10/>
2731 (P021) </wsp:Policy>
2732 (P022) </sp:X509Token>
2733 (P023) </wsp:Policy>
2734 (P024) </sp:RecipientToken>
2735 (P025) <sp:AlgorithmSuite>
2736 (P026) <wsp:Policy>
2737 (P027) <sp:Basic256/>
2738 (P028) </wsp:Policy>
2739 (P029) </sp:AlgorithmSuite>
2740 (P030) <sp:Layout>
2741 (P031) <wsp:Policy>
2742 (P032) <sp:Strict/>
2743 (P033) </wsp:Policy>
2744 (P034) </sp:Layout>
2745 (P035) <sp:IncludeTimestamp/>
2746 (P036) <sp:OnlySignEntireHeadersAndBody/>
2747 (P037) </wsp:Policy>
2748 (P038) </sp:AsymmetricBinding>
2749 (P039) <sp:Wss10>
2750 (P040) <wsp:Policy>
2751 (P041) <sp:MustSupportRefKeyIdentifier/>
2752 (P042) </wsp:Policy>
2753 (P043) </sp:Wss10>
2754 (P044) <sp:SignedSupportingTokens>
2755 (P045) <wsp:Policy>
2756 (P046) <sp:SamlToken sp:IncludeToken="http://docs.oasis-
2757 open.org/ws-sx/ws-
2758 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
2759 <wsp:Policy>
2760 (P048) <sp:WssSamlV11Token10/>
2761 (P049) </wsp:Policy>
2762 (P050) </sp:SamlToken>
2763 (P051) </wsp:Policy>
2764 (P052) </sp:SignedSupportingTokens>
2765 (P053) </wsp:All>
2766 (P054) </wsp:ExactlyOne>
2767 (P055) </wsp:Policy>
2768 (P056)
2769 (P057) <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_input_policy">
2770 (P058) <wsp:ExactlyOne>
2771 (P059) <wsp:All>
2772 (P060) <sp:SignedParts>
2773 (P061) <sp:Body/>
2774 (P062) </sp:SignedParts>
2775 (P063) <sp:EncryptedParts wsp:Optional="true">
2776 (P064) <sp:Body/>
2777 (P065) </sp:EncryptedParts>
2778 (P066) </wsp:All>
2779 (P067) </wsp:ExactlyOne>

```

```

2780      (P068)    </wsp:Policy>
2781
2782      (P069)    <wsp:Policy wsu:Id="Wss10SamlSvV11Asymm_output_policy">
2783      (P070)      <wsp:ExactlyOne>
2784      (P071)        <wsp:All>
2785      (P072)          <sp:SignedParts>
2786      (P073)            <sp:Body/>
2787      (P074)          </sp:SignedParts>
2788      (P075)          <sp:EncryptedParts>
2789      (P076)            <sp:Body/>
2790      (P077)          </sp:EncryptedParts>
2791      (P078)        </wsp:All>
2792      (P079)      </wsp:ExactlyOne>
2793      (P080)    </wsp:Policy>

```

2794 Line (P004) is a `wsp:All` element whose scope carries down to line (P053), which means all 3 of the  
2795 contained assertions: (P005)-(P038) `AsymmetricBinding`, (P039)-(P043) `WSS10`, and (P044)-(P052)  
2796 `SignedSupportingTokens` must be complied with by message exchanges to a Web Service covered by  
2797 this policy.

2798 Lines (P005)-(P038) contain an `AsymmetricBinding` assertion which indicates that the Initiator's token  
2799 must be used for the message signature and that the recipient's token must be used for message  
2800 encryption.

2801 Lines (P007)-(P015) contain the `InitiatorToken` assertion. Within the `InitiatorToken` assertion, lines  
2802 (P009)-(P013) indicate that the Initiator's token must be an `X509Token` that must always be included as  
2803 part of the request message (as opposed to an external reference). Line (P011) indicates that the `X509`  
2804 token must be an `X.509 V3` signature-verification certificate and used in the manner described in  
2805 [\[WSS10-X509-PROFILE\]](#).

2806 Lines (P016)-(P023) contain the `RecipientToken` assertion. Again, this an `X.509 V3` certificate (P020) that  
2807 must be used as described in [\[WSS10-X509-PROFILE\]](#), however the token itself, must never be included  
2808 in messages in either direction (P018).

2809 Instead (momentarily skipping slightly ahead), policy lines (P039)-(P043), the `WSS10` assertion, indicate  
2810 that the Recipient's token must be referenced by a `KeyIdentifier` element, which is described in the `WS-`  
2811 `Security 1.0` core specification [\[WSS10-SOAPMSG\]](#).

2812 Lines (P025)-(P029) contain the `AlgorithmSuite` assertion, which specifies the `sp:Basic256` set of  
2813 cryptographic components. The relevant values for this example within the `sp:Basic256` components are  
2814 the asymmetric signing algorithm, `ds:rsa-sha1`, and the digest algorithm, `ds:sha1`, where "ds:" =  
2815 "http://www.w3.org/2000/09/xmlsig#" [\[XML-DSIG\]](#).

2816 Lines (P030)-(P034) contain the `sp:Layout` assertion, which is set to `sp:Strict` (P032), which governs the  
2817 required relative layout of various `Timestamp`, `Signature` and `Token` elements in the messages.

2818 Line (P035) `IncludeTimestamp` means a `Timestamp` element must be included in the `WS-Security` header  
2819 block and signed by the message signature for messages in both directions.

2820 Line (P036) `OnlySignEntireHeaderAndBody` indicates that the message signature must explicitly cover  
2821 the `SOAP:Body` element (not children), and if there are any signed elements in the `SOAP:Header` they  
2822 must be direct child elements of the `SOAP:Header`. However, the one exception where the latter condition  
2823 is not applied is that if the child of the `SOAP:Header` is a `WS-Security` header (i.e. a `wsse:Security`  
2824 element), then individual direct child only elements of that `Security` element may also be signed.

2825 Lines (P044)-(P052) contain a `SignedSupportingTokens` assertion, which contains only one token  
2826 (P046)-(P050), a `SamlToken`, which must always be sent to the Recipient (P046) and it must be a  
2827 `SAML 1.1 Assertion` token. Because the `SamlToken` is in a "SignedSupportingTokens" element, it is  
2828 implicitly a `SAML sender-vouches ConfirmationMethod` token as described in section 2.3 above.

2829 Lines (P057)-(P068) contain a policy that applies only to input messages from the Initiator to the  
2830 Recipient.

2831 Lines (P060)-(P062) specify that the `SOAP:Body` element of the input message must be signed by the  
2832 Initiator's message signature.

2833 Lines (P063)-(P065) specify that the SOAP:Body element of the input message may optionally be  
2834 encrypted (signified by wsp:Optional="true") using the Recipient's encryption token (in this case (P020), it  
2835 is an X.509 certificate).

2836 Lines (P069)-(P080) contain a policy that applies only to output messages from the Recipient to the  
2837 Initiator.

2838 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be signed by the  
2839 Recipient's message signature.

2840 Lines (P075)-(P077) specify that the SOAP:Body element of the output message must be encrypted by  
2841 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

2842 Here is an example request:

2843

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
2844 (M002) <S12:Envelope
2845 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
2846 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
2847 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
2848 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
2849 (M007) 200401-wss-wssecurity-secext-1.0.xsd"
2850 (M008)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
2851 (M009) 200401-wss-wssecurity-utility-1.0.xsd"
2852 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#
2853 (M011)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
2854 (M012) <S12:Header>
2855 (M013)   <wsse:Security S12:mustUnderstand="1">
2856 (M014)     <wsu:Timestamp wsu:Id="timestamp">
2857 (M015)       <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
2858 (M016)     </wsu:Timestamp>
2859 (M017)     <saml:Assertion
2860 (M018)       AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2861 (M019)       IssueInstant="2003-04-17T00:46:02Z"
2862 (M020)       Issuer=www.opensaml.org
2863 (M021)       MajorVersion="1"
2864 (M022)       MinorVersion="1">
2865 (M023)       <saml:Conditions
2866 (M024)         NotBefore="2002-06-19T16:53:33.173Z"
2867 (M025)         NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
2868 (M026)       <saml:AttributeStatement>
2869 (M027)         <saml:Subject>
2870 (M028)           <saml:NameIdentifier
2871 (M029)             NameQualifier=www.example.com
2872 (M030)             Format="">
2873 (M031)             uid=joe,ou=people,ou=saml-demo,o=example.com
2874 (M032)           </saml:NameIdentifier>
2875 (M033)           <saml:SubjectConfirmation>
2876 (M034)             <saml:ConfirmationMethod>
2877 (M035)               urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
2878 (M036)             </saml:ConfirmationMethod>
2879 (M037)           </saml:SubjectConfirmation>
2880 (M038)         </saml:Subject>
2881 (M039)         <saml:Attribute>
2882 (M040)           ...
2883 (M041)         </saml:Attribute>
2884 (M042)         ...
2885 (M043)       </saml:AttributeStatement>
2886 (M044)     </saml:Assertion>
2887 (M045)     <wsse:SecurityTokenReference wsu:id="STR1">
2888 (M046)       <wsse:KeyIdentifier wsu:id="..."
2889 (M047)         ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-
2890 (M048) 2004XXwss-saml-token-profile-1.0#SAMLAssertionID">
2891 (M049)       _a75adf55-01d7-40cc-929f-dbd8372ebdfc
2892 (M049)     </wsse:KeyIdentifier>
```

```

2894 (M050) </wsse:SecurityTokenReference>
2895 (M051) <wsse:BinarySecurityToken
2896 (M052)   wsu:Id="attesterCert"
2897 (M053)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2898 (M054) 200401-wss-x509-token-profile-1.0#x509v3"
2899 (M055)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
2900 (M056) 200401-wss-soap-message-security-1.0#Base64Binary">
2901 (M057)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
2902 (M058) </wsse:BinarySecurityToken>
2903 (M059) <ds:Signature>
2904 (M060)   <ds:SignedInfo>
2905 (M061)     <ds:CanonicalizationMethod
2906 (M062)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2907 (M063)     <ds:SignatureMethod
2908 (M064)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2909 (M065)     <ds:Reference URI="#STR1">
2910 (M066)       <ds:Transforms>
2911 (M067)         <ds:Transform
2912 (M068)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
2913 (M069) 200401-wss-soap-message-security-1.0#STR-Transform">
2914 (M070)           <wsse:TransformationParameters>
2915 (M071)             <ds:CanonicalizationMethod
2916 (M072)               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2917 (M073)             </wsse:TransformationParameters>
2918 (M074)           </ds:Transform>
2919 (M075)         </ds:Transforms>
2920 (M076)       <ds:DigestMethod
2921 (M077)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2922 (M078)       <ds:DigestValue>...</ds:DigestValue>
2923 (M079)     </ds:Reference>
2924 (M080)     <ds:Reference URI="#MsgBody">
2925 (M081)       <ds:DigestMethod
2926 (M082)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2927 (M083)       <ds:DigestValue>...</ds:DigestValue>
2928 (M084)     </ds:Reference>
2929 (M085)     <ds:Reference URI="#timestamp">
2930 (M086)       <ds:DigestMethod
2931 (M087)         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2932 (M088)       <ds:DigestValue>...</ds:DigestValue>
2933 (M089)     </ds:Reference>
2934 (M090)   </ds:SignedInfo>
2935 (M091)   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
2936 (M092)   <ds:KeyInfo>
2937 (M093)     <wsse:SecurityTokenReference wsu:id="STR2">
2938 (M094)       <wsse:Reference wsu:id="..." URI="#attesterCert" />
2939 (M095)     </wsse:SecurityTokenReference>
2940 (M096)   </ds:KeyInfo>
2941 (M097) </ds:Signature>
2942 (M098) </wsse:Security>
2943 (M099) </S12:Header>
2944 (M0100) <S12:Body wsu:Id="MsgBody">
2945 (M0101)   <ReportRequest>
2946 (M0102)     <TickerSymbol>SUNW</TickerSymbol>
2947 (M0103)   </ReportRequest>
2948 (M0104) </S12:Body>
2949 (M0105) </S12:Envelope>

```

2950

2951 **Note:** For the instructional purposes of this document, in order to be compliant with WS-  
2952 SecurityPolicy, this copy of the original message had to be modified from the original. In particular,  
2953 the modifications that are applied above are the inclusion of a wsu:Id attribute in the Timestamp  
2954 element start tag (M014) and the addition of a ds:Reference element and URI attribute identifying  
2955 that Timestamp element in the message signature (M085)-(M089). (The original message in

2956 [WSS10-SAML11-INTEROP scenario #3] is not compliant with WS-SecurityPolicy because the  
2957 Timestamp that it contains is not covered by the message signature in that document.)

2958 Lines (M002)-(M0105) contain the SOAP:Envelope, i.e. the whole SOAP message.

2959 Lines (M012)-(M099) contain the SOAP:Header, which is the header portion of the SOAP message.

2960 Lines (M0100)-(M0104) contain the SOAP:Body, which is just some dummy content for test purposes.

2961 Lines (M013)-(M098) contain the wsse:Security header, which is the primary focus of this example.

2962 Lines (M014)-(M016) contain the wsu:Timestamp, which is required by the policy (P035).

2963 Lines (M017)-(M044) contain the SAML Assertion, which is the sp:Sam1Token required by the policy  
2964 sp:SignedSupportingTokens (P046)-(P050).

2965 Lines (M021)-(M022) identify the SAML Assertion as being Version 1.1 as required by the policy (P048).

2966 Line (M035) identifies the SAML Assertion as having ConfirmationMethod sender-vouches, which is the  
2967 implicit requirement of the sp:Sam1Token being in the SignedSupportingTokens as described above in the  
2968 policy section covering (P044)-(P052).

2969 Lines (M045)-(M050) contain a wsse:SecurityTokenReference which contains a wsse:KeyIdentifier, which  
2970 is used to reference the SAML Assertion, as described in [WSS10-SAML11-PROFILE] and required by  
2971 the policy (P041).

2972 Lines (M051)-(M058) contain the Initiator's wsse:BinarySecurityToken, which is required by the policy  
2973 (P007)-(P015), and in particular lines (M053)-(M054) identify the token as an X.509 V3 token as required  
2974 by the policy (P011). Line (M057) contains a truncated portion of the Base64 representation of the actual  
2975 certificate, which is included in the message as required by the policy sp:IncludeToken (P009).

2976 Lines (M059)-(M0102) contain the ds:Signature, which is the sp: message signature as required by  
2977 various parts of the Endpoint Policy (P001)-(P055) and Input Message Policy (P057)-(P068).

2978 Lines (M060)-(M095) contain the ds:SignedInfo element which describes the signing algorithm used  
2979 (M064) and specific elements that are signed (M065)-(M094) as required by the policies, which are  
2980 described in detail immediately following.

2981 Lines (M061)-(M062) contain the ds:CanonicalizationMethod Algorithm, which is specified as xml-exc-  
2982 c14n#, which is the default value required by the policy sp:AlgorithmSuite (P025)-(P029).

2983 Line (M064) identifies the SignatureMethod Algorithm as ds:rsa-sha1, which is the asymmetric key  
2984 signing algorithm required by the policy sp:AlgorithmSuite (P025)-(P029).

2985 Lines (M065)-(M079) identify the SAML Assertion (M017)-(M044) as an element that is signed, which is  
2986 referenced through the URI "#STR1" on line (M065), which references the SecurityTokenReference  
2987 (M045)-(M050), which in turn uses the identifier on line (M048) to reference the actual SAML Assertion by  
2988 its AssertionID attribute on line (M018). The SAML Assertion is required to be signed because it is  
2989 identified in the policy within the SignedSupportingTokens element on line (P048).

2990 Lines (M077)-(M078) contain the digest algorithm, which is specified to be sha1, as required by the policy  
2991 sp:Basic256 assertion (P027) in the sp:AlgorithmSuite.

2992 Lines (M080)-(M084) identify the SOAP:Body (M0100)-(M0104) as an element that is signed, which is  
2993 referenced through the URI "#MsgBody" on line (M080). The SOAP Body is required to be signed by the  
2994 input message policy lines (P060)-(P062).

2995 Lines (M085)-(M089) identify the wsu:Timestamp (M014)-(M016) as an element that is signed, which is  
2996 referenced through the URI "#timestamp" on line (M085). The wsu:Timestamp is required to be signed by  
2997 the policy sp:IncludeTimestamp assertion (P035). Note that the wsu:Timestamp occurs in the  
2998 wsse:Security header prior to the ds:Signature as required by the sp:Strict layout policy (P032).

2999 Finally, lines (M092)-(M096) contain the ds:KeyInfo element that identifies the signing key associated with  
3000 this ds:Signature element. The actual signing key is referenced through the  
3001 wsse:SecurityTokenReference (M093)-(M095), with URI "#attesterCert", which identifies the  
3002 InitiatorToken identified by the policy (P011) and contained in the wsse:BinarySecurityToken element  
3003 (M051)-(M058), which occurs in the wsse:Security header prior to this ds:Signature element, as required  
3004 by the sp:Strict layout policy assertion (P032). (Note: this BinarySecurityToken would need to be included

3005 in the signature, if the sp:AsymmetricBinding assertion in the endpoint policy contained a  
3006 sp:ProtectTokens assertion, but it does not, so it does not need to be signed.)

### 3007 2.3.1.5 (WSS1.0) SAML1.1 Holder of Key, Sign, Optional Encrypt

3008 This example is based on the first WSS SAML Profile InterOp [[WSS10-SAML11-INTEROP Scenario #4](#)].

3009 In this example the SAML token provides the key material for message security, hence acts as the  
3010 Initiator token, and therefore the Requestor and Initiator may be considered to be the same entity. The  
3011 SAML HK Assertion contains a reference to the public key of the signer of the message (the Initiator). The  
3012 Initiator knows Recipient's public key, which it may use to encrypt the request, but the Initiator does not  
3013 share a direct trust relation with the Recipient except indirectly through the SAML Assertion Issuer. The  
3014 Recipient has a trust relation with the Authority that issues the SAML HK Assertion. The indirect trust  
3015 relation between the Recipient and the Initiator is established by the fact that the Initiator's public key has  
3016 been signed by the Authority in the SAML HK Assertion. On the request the message body is signed  
3017 using Initiator's private key referenced in the SAML HK Assertion and it is optionally encrypted using  
3018 Recipient's server certificate. On the response, the server signs the message using its private key and  
3019 encrypts the message using the key provided within SAML HK Assertion.

3020 HK Note: there is a trust model aspect to the WS-Security holder-of-key examples that  
3021 implementors may want to be aware of. The [[SAML20-CORE](#)] specification defines the Subject of  
3022 the SAML Assertion such that the "presenter" of the Assertion is the entity that "attests" to the  
3023 information in the Assertion. "The attesting entity and the actual Subject **may or may not** be the  
3024 same entity." In general, these two choices map to the actors in [Figure 1](#) as follows: the  
3025 "attesting" entity may be regarded as the Initiator. The Subject entity, about whom the information  
3026 in the Assertion applies, may be regarded as the Requestor. In the case where the Subject and  
3027 attestor are one and the same, the Initiator and Requestor may be regarded as one and the  
3028 same. In this case the potential exists to use the Assertion for non-repudiation purposes with  
3029 respect to messages that the Requestor/Initiator signs. When the Subject and attestor are  
3030 separate entities, then holder-of-key is more similar to sender-vouches where the Initiator/attestor  
3031 is sending the message on behalf of the Subject. The mechanisms for determining whether the  
3032 Subject and attestor may be verified to be the same entity are dependent on the arrangements  
3033 among the business entities and the structure of the associated application scenarios.

3034

```
3035 (P001) <wsp:Policy xmlns:wsp="..." xmlns:wsu="..." xmlns:sp="..."  
3036 (P002) wsu:Id="Wss10SamlHkV11Asymm_endpoint_policy">  
3037 (P003) <wsp:ExactlyOne>  
3038 (P004) <wsp>All>  
3039 (P005) <sp:AsymmetricBinding>  
3040 (P006) <wsp:Policy>  
3041 (P007) <sp:InitiatorToken>  
3042 (P008) <wsp:Policy>  
3043 (P009) <sp:SamlToken sp:IncludeToken="http://docs.oasis-  
3044 open.org/ws-sx/ws-  
3045 securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3046 (P010) <wsp:Policy>  
3047 (P011) <wsp:WssSamlV11Token10/>  
3048 (P012) </wsp:Policy>  
3049 (P013) </sp:SamlToken>  
3050 (P014) </wsp:Policy>  
3051 (P015) </sp:InitiatorToken>  
3052 (P016) <sp:RecipientToken>  
3053 (P017) <wsp:Policy>  
3054 (P018) <sp:X509Token sp:IncludeToken="http://docs.oasis-  
3055 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">  
3056 (P019) <wsp:Policy>  
3057 (P020) <sp:WssX509V3Token10/>  
3058 (P021) </wsp:Policy>  
3059 (P022) </sp:X509Token>  
3060 (P023) </wsp:Policy>  
3061 (P024) </sp:RecipientToken>
```

```

3062 (P025) <sp:AlgorithmSuite>
3063 (P026) <wsp:Policy>
3064 (P027) <sp:Basic256/>
3065 (P028) </wsp:Policy>
3066 (P029) </sp:AlgorithmSuite>
3067 (P030) <sp:Layout>
3068 (P031) <wsp:Policy>
3069 (P032) <sp:Strict/>
3070 (P033) </wsp:Policy>
3071 (P034) </sp:Layout>
3072 (P035) <sp:IncludeTimestamp/>
3073 (P036) <sp:OnlySignEntireHeadersAndBody/>
3074 (P037) </wsp:Policy>
3075 (P038) </sp:AsymmetricBinding>
3076 (P039) <sp:Wss10>
3077 (P040) <wsp:Policy>
3078 (P041) <sp:MustSupportRefKeyIdentifier/>
3079 (P042) </wsp:Policy>
3080 (P043) </sp:Wss10>
3081 (P044) </wsp:All>
3082 (P045) </wsp:ExactlyOne>
3083 (P046) </wsp:Policy>
3084 (P047)
3085 (P048) <wsp:Policy wsu:Id="WSS10SamlHok_input_policy">
3086 (P049) <wsp:ExactlyOne>
3087 (P050) <wsp:All>
3088 (P051) <sp:SignedParts>
3089 (P052) <sp:Body/>
3090 (P053) </sp:SignedParts>
3091 (P054) <wsp:ExactlyOne>
3092 (P055) <wsp:All>
3093 (P056) <sp:EncryptedParts>
3094 (P057) <sp:Body/>
3095 (P058) </sp:EncryptedParts>
3096 (P059) </wsp:All>
3097 (P060) <wsp:All/>
3098 (P061) </wsp:ExactlyOne>
3099 (P062) </wsp:All>
3100 (P063) </wsp:ExactlyOne>
3101 (P064) </wsp:Policy>
3102 (P065)
3103 (P066) <wsp:Policy wsu:Id="WSS10SamlHok_output_policy">
3104 (P067) <wsp:ExactlyOne>
3105 (P068) <wsp:All>
3106 (P069) <sp:SignedParts>
3107 (P070) <sp:Body/>
3108 (P071) </sp:SignedParts>
3109 (P072) <sp:EncryptedParts>
3110 (P073) <sp:Body/>
3111 (P074) </sp:EncryptedParts>
3112 (P075) </wsp:All>
3113 (P076) </wsp:ExactlyOne>
3114 (P077) </wsp:Policy>

```

3115

3116 Lines (P001)-(P046) contain the endpoint policy, and lines (P048)-(P064) and (P066)-(P077) contain the  
3117 message input and message output policies, respectively.

3118 Lines (P005)-(P038) contain the AsymmetricBinding assertion, which requires separate security tokens  
3119 for the Initiator and Recipient.

3120 Lines (P007)-(P015) contain the InitiatorToken, which is defined to be a SamlToken (P009)-(P013), which  
3121 must always be sent to the Recipient (P009). The SamlToken is further specified by the  
3122 WssSamlV11Token10 assertion (P011) that requires the token to be a SAML 1.1 Assertion and the token  
3123 must be used in accordance with the WS-Security [[WSS10-SAML11-PROFILE](#)]. Furthermore, as

3124 described in section 2.3 above, because the SamlToken assertion appears within an InitiatorToken  
3125 assertion, the SAML Assertion must use the holder-of-key ConfirmationMethod, whereby for the indicated  
3126 WS-Security profile, the SAML Assertion contains a reference to the Initiator's X.509 signing certificate or  
3127 equivalent.

3128 Lines (P016)-(P024) contain the RecipientToken assertion. Again, this an X.509 V3 certificate (P020) that  
3129 must be used as described in [WSS10-SAML11-PROFILE], however the token itself, must never be  
3130 included in messages in either direction (P018) (i.e not only will the Recipient not send the token, but the  
3131 Initiator must not send the actual token, even when using it for encryption).

3132 Lines (P025)-(P029) AlgorithmSuite and (P030)-(P034) Layout are the same as described above in  
3133 section 2.3.1.4.

3134 Line (P035) IncludeTimestamp means a Timestamp element must be included in the WS-Security header  
3135 block and signed by the message signature for messages in both directions.

3136 Line (P036) OnlySignEntireHeaderAndBody indicates that the message signature must explicitly cover  
3137 the SOAP:Body element (not children), and if there are any signed elements in the SOAP:Header they  
3138 must be direct child elements of the SOAP:Header. However, the one exception where the latter condition  
3139 is not applied is that if the child of the SOAP:Header is a WS-Security header (i.e. a wsse:Security  
3140 element), then individual direct child only elements of that Security element may also be signed.

3141

3142 Lines (P039)-(P043) contain the WSS10 assertion, which indicates that the Recipient's token must be  
3143 referenced by a KeyIdentifier element (P041), the usage of which is described in the WS-Security 1.0  
3144 core specification [[WSS10-SOAPMSG](#)].

3145 Lines (P048)-(P064) contain the message input policy that applies only to messages from the Initiator to  
3146 the Recipient.

3147 Lines (P051)-(P053) specify that the SOAP:Body element of the input message must be signed by the  
3148 Initiator's message signature.

3149 Lines (P054)-(P061) specify that the SOAP:Body element of the input message may optionally (signified  
3150 by the empty policy alternative on line (P060)) be encrypted using the Recipient's encryption token (in this  
3151 case (P020), it is an X.509 certificate).

3152 Note: Because the input policy above (P048-P064) has the EncryptedParts assertion  
3153 (P056-P058) contained in an <ExactlyOne> element (P054-P061), which also contains an empty  
3154 policy element (P060), either a message with an encrypted Body element or an unencrypted  
3155 Body element will be accepted.

3156 Lines (P066)-(P077) contain a message output policy that applies only to messages from the Recipient to  
3157 the Initiator.

3158 Lines (P069)-(P071) specify that the SOAP:Body element of the output message must be signed by the  
3159 Recipient's message signature.

3160 Lines (P072)-(P074) specify that the SOAP:Body element of the output message must be encrypted by  
3161 the Initiator's encryption token (in this case (P012), it is also an X.509 certificate).

3162

3163 The following example request is taken from the [[WSS10-SAML11-INTEROP](#)] document scenario #4:

3164

```
3165 (M001) <?xml version="1.0" encoding="utf-8" ?>
3166 (M002) <S12:Envelope
3167 (M003)   xmlns:S12=http://schemas.xmlsoap.org/soap/envelope/
3168 (M004)   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
3169 (M005)   xmlns:xsd=http://www.w3.org/2001/XMLSchema
3170 (M006)   xmlns:wsse=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
3171 (M007)   wss-wssecurity-secext-1.0.xsd"
3172 (M008)   xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
3173 (M009)   wssecurity-utility-1.0.xsd"
3174 (M010)   xmlns:ds=http://www.w3.org/2000/09/xmldsig#">
3175 (M011)   <S12:Header>
```

```

3176 (M012) <wsse:Security S12:mustUnderstand="1">
3177 (M013)   <wsu:Timestamp wsu:Id="timestamp">
3178 (M014)     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
3179 (M015)   </wsu:Timestamp>
3180 (M016)   <saml:Assertion
3181 (M017)     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3182 (M018)     IssueInstant="2003-04-17T00:46:02Z"
3183 (M019)     Issuer=www.opensaml.org
3184 (M020)     MajorVersion="1"
3185 (M021)     MinorVersion="1"
3186 (M022)     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
3187 (M023)     <saml:Conditions
3188 (M024)       NotBefore="2002-06-19T16:53:33.173Z"
3189 (M025)       NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
3190 (M026)     <saml:AttributeStatement>
3191 (M027)       <saml:Subject>
3192 (M028)         <saml:NameIdentifier
3193 (M029)           NameQualifier=www.example.com
3194 (M030)           Format="">
3195 (M031)           uid=joe,ou=people,ou=saml-demo,o=example.com
3196 (M032)         </saml:NameIdentifier>
3197 (M033)       <saml:SubjectConfirmation>
3198 (M034)         <saml:ConfirmationMethod>
3199 (M035)           urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
3200 (M036)         </saml:ConfirmationMethod>
3201 (M037)         <ds:KeyInfo>
3202 (M038)           <ds:KeyValue>...</ds:KeyValue>
3203 (M039)         </ds:KeyInfo>
3204 (M040)       </saml:SubjectConfirmation>
3205 (M041)     </saml:Subject>
3206 (M042)     <saml:Attribute
3207 (M043)       AttributeName="MemberLevel"
3208 (M044)       AttributeNamespace=
3209 (M045)         "http://www.oasis-open.org/Catalyst2002/attributes">
3210 (M046)       <saml:AttributeValue>gold</saml:AttributeValue>
3211 (M047)     </saml:Attribute>
3212 (M048)     <saml:Attribute
3213 (M049)       AttributeName="E-mail"
3214 (M050)       AttributeNamespace="http://www.oasis-
3215 (M051)         open.org/Catalyst2002/attributes">
3216 (M052)       <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
3217 (M053)     </saml:Attribute>
3218 (M054)   </saml:AttributeStatement>
3219 (M055)   <ds:Signature>...</ds:Signature>
3220 (M056) </saml:Assertion>
3221 (M057) <ds:Signature>
3222 (M058)   <ds:SignedInfo>
3223 (M059)     <ds:CanonicalizationMethod
3224 (M060)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
3225 (M061)   <ds:SignatureMethod
3226 (M062)     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3227 (M063)   <ds:Reference URI="#MsgBody">
3228 (M064)     <ds:DigestMethod
3229 (M065)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3230 (M066)     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
3231 (M067)   </ds:Reference>
3232 (M068)   <ds:Reference URI="#timestamp">
3233 (M069)     <ds:DigestMethod
3234 (M070)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3235 (M071)     <ds:DigestValue>...</ds:DigestValue>
3236 (M072)   </ds:Reference>
3237 (M073) </ds:SignedInfo>
3238 (M074) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
3239 (M075) <ds:KeyInfo>

```

```

3240 (M076)      <wsse:SecurityTokenReference wsu:id="STR1">
3241 (M077)      <wsse:KeyIdentifier wsu:id="..."
3242 (M078)      Value="http://docs.oasis-open.org/wss/2004/XX/oasis-
3243 (M079)      2004XX-wss-saml-token-profile-1.0#SAMLAssertionID">
3244 (M080)      _a75adf55-01d7-40cc-929f-dbd8372ebdfc
3245 (M081)      </wsse:KeyIdentifier>
3246 (M082)      </wsse:SecurityTokenReference>
3247 (M083)      </ds:KeyInfo>
3248 (M084)      </ds:Signature>
3249 (M085)      </wsse:Security>
3250 (M086)      </S12:Header>
3251 (M087)      <S12:Body wsu:Id="MsgBody">
3252 (M088)      <ReportRequest>
3253 (M089)      <TickerSymbol>SUNW</TickerSymbol>
3254 (M090)      </ReportRequest>
3255 (M091)      </S12:Body>
3256 (M092)      </S12:Envelope>

```

3257

3258 Note: for the instructional purposes of this document, it was necessary to make changes to the  
3259 original sample request to meet the requirements of WS-SecurityPolicy. The changes to the  
3260 message above include:

- 3261 • Line (M013): added wsu:Id="timestamp" attribute to sp:Timestamp element.
- 3262 • Lines (M068)-(M072): added a ds:Reference element to include the Timestamp in the  
3263 message signature required by the policy sp:IncludeTimestamp assertion (P035).

3264 Lines (M002)-(M092) contain the SOAP Envelope, i.e. the whole SOAP message.

3265 Lines (M011)-(M086) contain the SOAP Header.

3266 Lines (M087)-(M091) contain the unencrypted SOAP Body, which is allowed to be unencrypted based on  
3267 line (P060) of the input message policy, which is the alternative choice to encrypting based on lines  
3268 (P055)-(P059).

3269 Lines (M012)-(M080) contain the wsse:Security header entry, which is the only header entry in this SOAP  
3270 Header.

3271 Lines (M013)-(M015) contain the wsu:Timestamp which is required by the endpoint policy (P035).

3272 Lines (M016)-(M056) contain the SAML Assertion, which is required in the policy by the SamlToken  
3273 (P009)-(P013) contained in the InitiatorToken. The SAML Assertion is version 1.1 (M020)-(M021) as  
3274 required by the sp:WssSamlV11Token10 assertion (P011). The SAML Assertion is of type that uses the  
3275 holder-of-key saml:ConfirmationMethod [[SAML11-CORE](#)] (M034)-(M036) as required by the fact that the  
3276 SamlToken is contained in the InitiatorToken assertion of the policy (P009)-(P013), as explained in  
3277 section 2.3 above.

3278 Line (M055) contains the ds:Signature of the Issuer of the SAML Assertion. While the details of this  
3279 signature are not shown, it is this signature that the Recipient must ultimately trust in order to trust the rest  
3280 of the message.

3281 Lines (M057)-(M084) contain the message signature in a ds:Signature element.

3282 Lines (M058)-(M073) contain the ds:SignedInfo element, which identifies the elements to be signed.

3283 Lines (M063)-(M067) contains a ds:Reference to the SOAP:Body, which is signed in the same manner as  
3284 example section 2.3.1.4 above.

3285 Lines (M068)-(M072) contains a ds:Reference to the URI "timestamp", which again is signed in the same  
3286 manner as the wsu:Timestamp in section 2.3.1.4 above.

3287 Lines (M075)-(M083) contain the ds:KeyInfo element, which identifies the key that should be used to  
3288 verify this ds:Signature element. Lines (M076)-(M082) contain a wsse:SecurityTokenReference, which  
3289 contains a wsse:KeyIdentifier that identifies the signing key as being contained in a token of  
3290 wsse:ValueType "...wss-saml-token-profile-1.0#SAMLAssertionID", which means a SAML V1.1 Assertion  
3291 [[WSS10-SAML11-PROFILE](#)]. Line (M080) contains the saml:AssertionID of the SAML Assertion that  
3292 contains the signing key. This SAML Assertion is on lines (M016)-(M056) with the correct  
3293 saml:AssertionID on line (M017). Note that the fact that the referenced token (the SAML Assertion) occurs  
3294 in the wsse:Security header before this ds:KeyInfo element that uses it in compliance with the sp:Strict  
3295 layout policy (P032) and the wsse:KeyIdentifier is an acceptable token reference mechanism as specified  
3296 in the policy sp:Wss10 assertion containing a sp:MustSupportRefKeyIdentifier assertion (P041).

3297

## 3298 2.3.2 WSS 1.1 SAML Token Scenarios

3299 This section contains SamlToken examples that use WS-Security 1.1 SOAP Message Security [WSS11]  
3300 and the WS-Security 1.1 SAML Profile [WSS11-SAML1120-PROFILE].

### 3301 2.3.2.1 (WSS1.1) SAML 2.0 Bearer

3302 This example is based on the Liberty Alliance Identity Web Services Framework (ID-WSF 2.0) Security  
3303 Mechanism for the SAML 2.0 Profile for WSS 1.1 [WSS11-LIBERTY-SAML20-PROFILE], which itself is  
3304 based on the [WSS11-SAML1120-PROFILE].

3305 In this example, an AsymmetricBinding is used for message protection provided by the Initiator, however,  
3306 Recipient trust for the content is based only on the SAML bearer token provided by the Requestor.

3307

```
3308 (P001) <wsp:Policy>
3309 (P002)   <sp:AsymmetricBinding>
3310 (P003)     <wsp:Policy>
3311 (P004)       <sp:InitiatorToken>
3312 (P005)         <wsp:Policy>
3313 (P006)           <sp:X509Token sp:IncludeToken="http://docs.oasis-
3314 open.org/ws-sx/ws-
3315 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3316 (P007)             <wsp:Policy>
3317 (P008)               <sp:WssX509V3Token10/>
3318 (P009)             </wsp:Policy>
3319 (P010)           </sp:InitiatorToken>
3320 (P011)         </wsp:Policy>
3321 (P012)       </sp:AsymmetricBinding>
3322 (P013)     <sp:RecipientToken>
3323 (P014)       <wsp:Policy>
3324 (P015)         <sp:X509Token sp:IncludeToken="http://docs.oasis-
3325 open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
3326 (P016)           <wsp:Policy>
3327 (P017)             <sp:WssX509V3Token10/>
3328 (P018)           </wsp:Policy>
3329 (P019)         </sp:RecipientToken>
3330 (P020)       </wsp:Policy>
3331 (P021)     </sp:SupportingTokens>
3332 (P022)   <sp:AlgorithmSuite>
3333 (P023)     <wsp:Policy>
3334 (P024)       <sp:Basic256/>
3335 (P025)     </wsp:Policy>
3336 (P026)   </sp:AlgorithmSuite>
3337 (P027) <sp:Layout>
3338 (P028)   <wsp:Policy>
3339 (P029)     <sp:Strict/>
3340 (P030)   </wsp:Policy>
3341 (P031) </sp:Layout>
3342 (P032) <sp:IncludeTimestamp/>
3343 (P033) <sp:OnlySignEntireHeadersAndBody/>
3344 (P034) </wsp:Policy>
3345 (P035) </sp:AsymmetricBinding>
3346 (P036) <sp:Wss11>
3347 (P037)   <wsp:Policy>
3348 (P038)     <sp:MustSupportRefKeyIdentifier/>
3349 (P039)     <sp:MustSupportRefIssuerSerial/>
3350 (P040)     <sp:MustSupportRefThumbprint/>
3351 (P041)     <sp:MustSupportRefEncryptedKey/>
3352 (P042)   </wsp:Policy>
3353 (P043) </sp:Wss11>
3354 (P044) <sp:SupportingTokens>
3355 (P045)   <wsp:Policy>
```

```

3356 (P046) <sp:SamIToken sp:IncludeToken="http://docs.oasis-open.org/ws-
3357 sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3358 (P047) <wsp:Policy>
3359 (P048) <sp:WssSamlV20Token11/>
3360 (P049) </wsp:Policy>
3361 (P050) </sp:SamIToken>
3362 (P051) </wsp:Policy>
3363 (P052) </sp:SupportingTokens>
3364 (P053) </wsp:Policy>

```

3365 Lines (P001)-(P045) contain the endpoint policy, which contains 3 assertions: an sp:AsymmetricBinding  
3366 assertion, an sp:Wss11 assertion, and an sp:SupportingTokens assertion.

3367 Lines (P002)-(P035) contain the sp:AsymmetricBinding assertion, which is identical to the same assertion  
3368 that is described in section 2.1.3. Please refer to section 2.1.3 for details.

3369 Lines (P036)-(P043) contain an sp:Wss11 assertion, which contains a set of assertions that specify the  
3370 token referencing techniques that are required to be supported (P038)-(P041).

3371 Lines (P044)-(P052) contain an sp:SupportingTokens assertion, which contains an sp:SamIToken  
3372 (P046)-(P050), which specifies that it must always be sent to the Recipient. The sp:SamIToken contains  
3373 an sp:WssSamlV20Token11 assertion (P048), which means that the SamIToken provided must be a  
3374 SAML Version 2.0 Assertion that is submitted in compliance with the [WSS11-SAML1120-PROFILE].

3375 The fact that the sp:SamIToken is contained in an sp:SupportingTokens element indicates to the Initiator  
3376 that the SAML Assertion must use the saml2:SubjectConfirmation@Method "bearer" as described above  
3377 in introductory section 2.3.

3378 The following is an example request compliant with the above policy:

3379

```

3380 (M001) <?xml version="1.0" encoding="UTF-8"?>
3381 (M002) <s:Envelope xmlns:s=".../soap/envelope/" xmlns:sec="..."
3382 (M003) xmlns:wss="..." xmlns:wsu="..." xmlns:wsa="...addressing"
3383 (M004) xmlns:sb="...liberty:sb" xmlns:pp="...liberty.id-sis-pp"
3384 (M005) xmlns:ds="...xmldsig#" xmlns:xenc="...xmlenc#">
3385 (M006) <s:Header>
3386 (M007) <!-- see Liberty SOAP Binding Spec for reqd, optional hdrs -->
3387 (M008) <wsa:MessageID wsu:Id="mid">...</wsa:MessageID>
3388 (M009) <wsa:To wsu:Id="to">...</wsa:To>
3389 (M010) <wsa:Action wsu:Id="action">...</wsa:Action>
3390 (M011) <wsse:Security mustUnderstand="1">
3391 (M012) <wsu:Timestamp wsu:Id="ts">
3392 (M013) <wsu:Created>2005-06-17T04:49:17Z</wsu:Created >
3393 (M014) </wsu:Timestamp>
3394 (M015) <!-- this is the bearer token -->
3395 (M016) <saml2:Assertion xmlns:saml2="...SAML:2.0:assertion"
3396 (M017) Version="2.0" ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
3397 (M018) IssueInstant="2005-04-01T16:58:33.173Z">
3398 (M019) <saml2:Issuer>http://authority.example.com/</saml2:Issuer>
3399 (M020) <!-- signature by the issuer over the assertion -->
3400 (M021) <ds:Signature>...</ds:Signature>
3401 (M022) <saml2:Subject>
3402 (M023) <saml2:EncryptedID>
3403 (M024) <xenc:EncryptedData
3404 (M025) >U2XTCNvRX7B11NK182nmY00TEk==</xenc:EncryptedData>
3405 (M026) <xenc:EncryptedKey>...</xenc:EncryptedKey>
3406 (M027) </saml2:EncryptedID>
3407 (M028) <saml2:SubjectConfirmation
3408 (M029) Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3409 (M030) </saml2:Subject>
3410 (M031) <!-- audience restriction on assertion can limit scope of
3411 (M032) which entity should consume the info in the assertion. -->
3412 (M033) <saml2:Conditions NotBefore="2005-04-01T16:57:20Z"
3413 (M034) NotOnOrAfter="2005-04-01T21:42:4 3Z">
3414 (M035) <saml2:AudienceRestrictionCondition>

```

```

3415 (M036)         <saml2:Audience>http://wsp.example.com</saml2:Audience>
3416 (M037)         </saml2:AudienceRestrictionCondition>
3417 (M038)         </saml2:Conditions>
3418 (M039)         <!-- The AuthnStatement carries info that describes authN
3419 (M040)         event of the Subject to an Authentication Authority -->
3420 (M041)         <saml2:AuthnStatement AuthnInstant="2005-04-01T16:57:30.000Z"
3421 (M042)         SessionIndex="6345789">
3422 (M043)         <saml2:AuthnContext>
3423 (M044)         <saml2:AuthnContextClassRef
3424 (M045)         >...:SAML:2.0:ac:classes:PasswordProtectedTransport
3425 (M046)         </saml2:AuthnContextClassRef>
3426 (M047)         </saml2:AuthnContext>
3427 (M048)         </saml2:AuthnStatement>
3428 (M049)         <!-- This AttributeStatement carries an EncryptedAttribute.
3429 (M050)         Once this element decrypted with supplied key an <Attribute>
3430 (M051)         element bearing endpoint reference can be found, specifying
3431 (M052)         resources which invoker may access. Details on element can be
3432 (M053)         found in discovery service specification. -->
3433 (M054)         <saml2:AttributeStatement>
3434 (M055)         <saml2:EncryptedAttribute>
3435 (M056)         <xenc:EncryptedData
3436 (M057)         Type="http://www.w3.org/2001/04/xmlenc#Element" >
3437 (M058)         mQEMAzRniWkAAAEH9RWir0eKDkyFAB7PoFazx3ftp0vWwbbzqXdgcX8
3438 (M059)         ... hg6nZ5c0I6L6Gn9A=HCQY
3439 (M060)         </xenc:EncryptedData>
3440 (M061)         <xenc:EncryptedKey> ... </xenc:EncryptedKey>
3441 (M062)         </saml2:EncryptedAttribute>
3442 (M063)         </saml2:AttributeStatement>
3443 (M064)         </saml2:Assertion>
3444 (M065)         <!-- This SecurityTokenReference is used to reference the SAML
3445 (M066)         Assertion from a ds:Reference -->
3446 (M067)         <wsse:SecurityTokenReference xmlns:wsse="..." xmlns:wsp="..."
3447 (M068)         xmlns:wss1="..." wsp:Id="str1" wss1:TokenType=
3448 (M069)         ".../wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
3449 (M070)         <wsse:KeyIdentifier ValueType=
3450 (M071)         ".../wss/oasis-wss-saml-token-profile-1.1#SAMLID"
3451 (M072)         >sxJu9g/vvLG9sAN9bKp/8q0NKU=</wsse:KeyIdentifier>
3452 (M073)         </wsse:SecurityTokenReference>
3453 (M074)         <ds:Signature>
3454 (M075)         <ds:SignedInfo>
3455 (M076)         <!-- in general include a ds:Reference for each wsa:header
3456 (M077)         added according to SOAP binding, plus timestamp, plus ref to
3457 (M078)         assertion to avoid token substitution attacks, plus Body -->
3458 (M079)         <ds:Reference URI="#to">...</ds:Reference>
3459 (M080)         <ds:Reference URI="#action">...</ds:Reference>
3460 (M081)         <ds:Reference URI="#mid">...</ds:Reference>
3461 (M082)         <ds:Reference URI="#ts">...</ds:Reference>
3462 (M083)         <ds:Reference URI="#Str1">
3463 (M084)         <ds:Transform Algorithm="...#STR-Transform">
3464 (M085)         <wsse:TransformationParameters>
3465 (M086)         <ds:CanonicalizationMethod Algorithm=
3466 (M087)         "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
3467 (M088)         </wsse:TransformationParameters>
3468 (M089)         </ds:Transform>
3469 (M090)         </ds:Reference>
3470 (M091)         <ds:Reference URI="#MsgBody">...</ds:Reference>
3471 (M092)         </ds:SignedInfo>
3472 (M093)         ...
3473 (M094)         </ds:Signature>
3474 (M095)         </wsse:Security>
3475 (M096)         </s:Header>
3476 (M097)         <s:Body wsp:Id="MsgBody">
3477 (M098)         <pp:Modify>
3478 (M099)         <!-- this is an ID-SIS-PP Modify message -->

```

```
3479 (M0100) </pp:Modify>
3480 (M0101) </s:Body>
3481 (M0102) </s:Envelope>
```

3482

3483 The sample message above was taken and cosmetically edited from the [\[WSS11-LIBERTY-SAML20-](#)  
3484 [PROFILE\]](#).

3485 Lines (M002)-(M0102) contain the SOAP:Envelope, which contains the SOAP:Header (M006)-(M096)  
3486 and the SOAP:Body (M097)-(M0101).

3487 The SOAP Header contains some WS-Addressing (wsa:) parameters (M008)-(M010) (not discussed  
3488 here) and a wsse:Security header.

3489 The wsse:Security header (M011)-(M095) contains a wsu:Timestamp (M012)-(M014), a saml2:Assertion  
3490 (M016)-(M064), a wsse:SecurityTokenReference (M067)-(M073), and a ds:Signature (M074)-(M094).

3491 The wsu:Timestamp (M012)-(M014) is required by the policy sp:IncludeTimestamp assertion (P032).

3492 The saml2:Assertion (M016)-(M064) is required by the policy sp:WssSamlV20Token11 (P048). The  
3493 saml2:Assertion is Version 2.0 (M017) and it is signed by the IssuingAuthority (M021). This is the  
3494 ds:Signature (M021) of the IssuingAuthority, identified in the saml2:Issuer element (M019), that the  
3495 Recipient trusts with respect to recognizing the Requestor and determining whether the request should be  
3496 granted. In addition, this saml2:Assertion uses the “bearer” saml2:SubjectConfirmation@Method, as  
3497 required by the policy sp:SamIToken in the sp:SupportingTokens element described above  
3498 (P044)-(P052).

3499 Note: the saml2:Assertion contains a saml2:EncryptedID (M023)-(M027), which contains the  
3500 identity of the Requestor and a saml2:EncryptedAttribute (M062), which contains information  
3501 about the Requestor. It is presumed that prior to issuing this request, that the Requestor  
3502 contacted the IssuingAuthority (directly or indirectly) and was granted permission to access the  
3503 web service that is now the target of this request. As such, the IssuingAuthority has knowledge of  
3504 this web service and presumably the public certificate associated with that service and has used  
3505 the public key contained in that certificate to encrypt the aforementioned portions of the  
3506 saml2:Assertion, which can only be decrypted by the RelyingParty who presumably has entrusted  
3507 the private key of the service with the Recipient, in order that the Recipient may decrypt the  
3508 necessary data.

3509 However, before the Recipient can evaluate these aspects of the request, the requirements of the policy  
3510 sp:AsymmetricBinding (P002)-(P035) must be met in terms of proper “presentation” of the request as  
3511 described below.

3512 Lines (M074)-(M094) contain the message signature ds:Signature element, which contains a  
3513 ds:SignedInfo that contains ds:Reference elements that cover the wsa: headers (M079)-(M081), the  
3514 wsu:Timestamp (M082) (required by the policy sp:IncludeTimestamp assertion (P032), the  
3515 saml2:Assertion (M083)-(M090), and the SOAP:Body (M091).

3516 The ds:Reference (M083)-(M090) covering the saml2:Assertion warrants further examination.

3517 The first point to note is that to reference the saml2:Assertion the ds:Reference uses an STR-  
3518 Transform (M084) to reference a wsse:SecurityTokenReference (M067)-(M073), which is in  
3519 compliance with policy sp:Wss11 sp:MustSupportRefKeyIdentifier assertion (P038).

3520 The second point to note about this ds:Reference is that is covering the saml2:Assertion even  
3521 though the policy references the sp:SamIToken in an sp:SupportingTokens element and not an  
3522 sp:SignedSupportingTokens element. The reason this is the case is that it is the fact that an  
3523 sp:SupportingTokens (P044)-(P052) element is used that tells the Initiator that a SAML bearer  
3524 Assertion is required. However, this only means that the SamIToken is not “required” to be  
3525 signed. It is the Initiator’s choice whether to sign it or not, and it is generally good practice to do  
3526 so in order to prevent a token substitution attack.

3527

3528

### 3529 **2.3.2.2 (WSS1.1) SAML2.0 Sender Vouches over SSL**

3530 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP Scenario](#)  
3531 #2].

3532 Similar to 2.3.1.2 except SAML token is of version 2.0.

3533

3534

```
3535 (P001) <wsp:Policy>
3536 (P002)   <sp:TransportBinding>
3537 (P003)   <wsp:Policy>
3538 (P004)   <sp:TransportToken>
3539 (P005)   <wsp:Policy>
3540 (P006)   <sp:HttpsToken>
3541 (P007)   <wsp:Policy>
3542 (P008)   <sp:RequireClientCertificate>
3543 (P009)   </wsp:Policy>
3544 (P010)   </sp:HttpsToken>
3545 (P011)   </wsp:Policy>
3546 (P012)   </sp:TransportToken>
3547 (P013)   <sp:AlgorithmSuite>
3548 (P014)   <wsp:Policy>
3549 (P015)   <sp:Basic256 />
3550 (P016)   </wsp:Policy>
3551 (P017)   </sp:AlgorithmSuite>
3552 (P018)   <sp:Layout>
3553 (P019)   <wsp:Policy>
3554 (P020)   <sp:Strict />
3555 (P021)   </wsp:Policy>
3556 (P022)   </sp:Layout>
3557 (P023)   <sp:IncludeTimestamp />
3558 (P024)   </wsp:Policy>
3559 (P025)   </sp:TransportBinding>
3560 (P026)   <sp:SignedSupportingTokens>
3561 (P027)   <wsp:Policy>
3562 (P028)   <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-
3563         sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3564 (P029)   <wsp:Policy>
3565 (P030)   <sp:WssSamlV20Token11/>
3566 (P031)   </wsp:Policy>
3567 (P032)   </sp:SamlToken>
3568 (P033)   </wsp:Policy>
3569 (P034)   </sp:SignedSupportingTokens>
3570 (P035)   </wsp:Policy>
```

3571 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by  
3572 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and  
3573 sp:SignedSupportingTokens (P026).

3574 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
3575 by a secure transport protocol such as SSL or TLS.

3576 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
3577 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
3578 transport token using the Basic256 algorithm suite (P015).

3579 In addition, because this is SAML sender-vouches, a client certificate is required (P008) as the basis of  
3580 trust for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section  
3581 4.3.1].

3582 The requirements for the sp:Layout assertion (P018)-(P022) should be automatically met (or may be  
3583 considered moot) since there are no cryptographic tokens required to be present in the WS-Security  
3584 header. (However, if a signature element was included to cover the wsse:Timestamp, then the layout  
3585 would need to be considered.)

3586 A timestamp (P023) is required to be included in an acceptable message.

3587

3588 Here is an example request:

3589

```
(M001) <?xml version="1.0" encoding="utf-8" ?>
3591 (M002) <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3592 (M003) xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3593 (M004) xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3594 (M005) xmlns:wsu=".../oasis-200401-wsswssecurity-utility-1.0.xsd"
3595 (M006) xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3596 (M007) xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3597 (M008) <S11:Header>
3598 (M009) <wsse:Security S11:mustUnderstand="1">
3599 (M010) <wsu:Timestamp>
3600 (M011) <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3601 (M012) </wsu:Timestamp>
3602 (M013) <saml2:Assertion ID=" a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3603 (M014) IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3604 (M015) <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3605 (M016) <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3606 (M017) NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3607 (M018) <saml2:Subject>
3608 (M019) <saml2:NameID Format=
3609 (M020) "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3610 (M021) >uid=joe,ou=people,ou=saml demo,o=example.com</saml2:NameID>
3611 (M022) <saml2:SubjectConfirmation Method=
3612 (M023) "urn:oasis:names:tc:SAML:2.0:cm:sender-vouches" />
3613 (M024) </saml2:Subject>
3614 (M025) <saml2:AttributeStatement>
3615 (M026) <saml2:Attribute>...</saml2:Attribute>
3616 (M027) <saml2:Attribute>...</saml2:Attribute>
3617 (M028) </saml2:AttributeStatement>
3618 (M029) </saml2:Assertion>
3619 (M030) </wsse:Security>
3620 (M031) </S11:Header>
3621 (M032) <S11:Body>
3622 (M033) <Ping xmlns="http://xmlsoap.org/Ping">
3623 (M034) <text>EchoString</text>
3624 (M035) </Ping>
3625 (M036) </S11:Body>
3626 (M037) </S11:Envelope>
```

3627 Lines (M002)-(M037) contain the SOAP:Envelope, which contains the SOAP:Header (M003)-(M031) and  
3628 the SOAP:Body (M032)-(M036).

3629 Lines (M009)-(M030) contain the wsse:Security header, which, in conjunction with the client  
3630 certificate (P008), is the primary basis for trust in this example.

3631 Lines (M010)-(M012) contain the wsu:Timestamp, which meets the sp:IncludeTimestamp assertion policy  
3632 requirement (P023).

3633 Lines (M013)-(M029) contain the saml2:Assertion, which is required to be Version 2.0 (M014) and to be  
3634 used within the [WSS11-SAML1120-PROFILE], because of the policy sp:SamIToken assertion  
3635 (P028)-(P032), which contains the sp:WssSamIv20Token11 (P030).

3636 Lines (M022)-(M023) indicate that the SAML Assertion uses the saml2:Method sender-vouches for the  
3637 purposes of saml2:SubjectConfirmation, which is required by the fact that the sp:SamIToken appears in  
3638 an sp:SignedSupportingTokens assertion (P026)-(P034) in conjunction with the client certificate required  
3639 by the sp:RequireClientCertificate assertion (P008) within the sp:TransportBinding as described in section  
3640 2.3 above. In addition, the Recipient should be able to correlate the saml2:Issuer (M015) as being  
3641 properly associated with the client certificate received from the HTTPS (SSL) connection.

3642

3643

### 3644 2.3.2.3 (WSS1.1) SAML2.0 HoK over SSL

3645 This scenario is based on second WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)  
3646 Scenario #5].

3647 Similar to 2.3.1.3 except SAML token is of version 2.0.

3648 Initiator adds a SAML Assertion (hk) to the SOAP Security Header. In this policy the sp:TransportBinding  
3649 requires a Client Certificate AND the sp:SamlToken is in an sp:SignedEndorsingSupportingTokens  
3650 element. A SAML holder-of-key Assertion meets these requirements because it is “virtually signed” by the  
3651 message signature as a result of the SSL client certificate authentication procedure as described in  
3652 section 2.3. Furthermore, the SAML hk Assertion in this case is a “virtually endorsing”, because the key  
3653 identified in the holder-of-key saml2:SubjectConfirmationData is also the client certificate, which is  
3654 virtually endorsing its own signature, under the authority of the IssuingAuthority who has signed the  
3655 SAML hk Assertion.

3656 As a result, the Initiator may be considered to be authorized by the saml2:Issuer of the hk SAML  
3657 Assertion to bind message content to the Subject of the Assertion. If the Client Certificate matches the  
3658 certificate identified in the hk Assertion, the Initiator may be regarded as executing SAML hk responsibility  
3659 of binding the Requestor, who would be the Subject of the hk Assertion, to the content of the message.

3660 (Note: the same considerations described in section 2.3.1.4 with respect to whether the Subject  
3661 of the Assertion and the Subject of the Client Certificate are the same entity determining whether  
3662 the Client Certificate is attesting for the Assertion Subject or whether the Client Certificate is  
3663 authenticating as the Assertion Subject apply here.)

3664 In this scenario, the IssuingAuthority is the saml2:Issuer(signer) of the hk SAML Assertion. The Requestor  
3665 is the Subject of the Assertion and the Initiator is authorized by the IssuingAuthority to bind the Assertion  
3666 to the message using the ClientCertificate identified in the SAML Assertion, which may also be  
3667 considered to be virtually signing the wsu:Timestamp of the message. Optionally, a separate Signature  
3668 may be used to sign the wsu:Timestamp, which the Recipient would also be required to verify was signed  
3669 by the client certificate in this example.

3670

```
3671 (P001) <wsp:Policy>  
3672 (P002)   <sp:TransportBinding>  
3673 (P003)   <wsp:Policy>  
3674 (P004)     <sp:TransportToken>  
3675 (P005)       <wsp:Policy>  
3676 (P006)         <sp:HttpsToken>  
3677 (P007)           <wsp:Policy>  
3678 (P008)             <sp:RequireClientCertificate>  
3679 (P009)               </wsp:Policy>  
3680 (P010)             </sp:HttpsToken>  
3681 (P011)           </wsp:Policy>  
3682 (P012)         </sp:TransportToken>  
3683 (P013)       <sp:AlgorithmSuite>  
3684 (P014)         <wsp:Policy>  
3685 (P015)           <sp:Basic256 />  
3686 (P016)           </wsp:Policy>  
3687 (P017)         </sp:AlgorithmSuite>  
3688 (P018)       <sp:Layout>  
3689 (P019)         <wsp:Policy>  
3690 (P020)           <sp:Strict />  
3691 (P021)           </wsp:Policy>  
3692 (P022)         </sp:Layout>  
3693 (P023)       <sp:IncludeTimestamp />  
3694 (P024)     </wsp:Policy>  
3695 (P025)   </sp:TransportBinding>  
3696 (P026) <sp:SignedEndorsingSupportingTokens>  
3697 (P027)   <wsp:Policy>  
3698 (P028)     <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-  
3699           sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">  
3700 (P029)   </wsp:Policy>
```

```

3701      (P030)          <sp:WssSamlV20Token11/>
3702      (P031)          </wsp:Policy>
3703      (P032)          </sp:SamlToken>
3704      (P033)          </wsp:Policy>
3705      (P034)          </sp:SignedEndorsingSupportingTokens>
3706      (P035)          </wsp:Policy>

```

3707

3708 Lines (P001)-(P035) contain the policy that requires all the contained assertions to be complied with by  
3709 the Initiator. In this case there are 2 assertions: sp:TransportBinding (P002) and  
3710 sp:EndorsingSupportingTokens (P026).

3711 Lines (P002)-(P025) contain a TransportBinding assertion that indicates the message must be protected  
3712 by a secure transport protocol such as SSL or TLS.

3713 Lines (P004)-(P012) contain a TransportToken assertion indicating that the transport is secured by  
3714 means of an HTTPS TransportToken, requiring cryptographic operations to be performed based on the  
3715 transport token using the Basic256 algorithm suite (P015).

3716 In addition, because this is SAML holder-of-key, a client certificate is required (P008) as the basis of trust  
3717 for the SAML Assertion and for the content of the message [[WSS10-SAML11-INTEROP](#) section 4.3.1]. In  
3718 the holder-of-key case, there will be an additional certificate required in the trust chain, which is the  
3719 certificate used to sign the SAML hk Assertion, which will be contained or referenced in the Assertion.

3720 The layout requirement in this case (P018)-(P022) is automatically met (or may be considered moot)  
3721 since there are no cryptographic tokens required to be present in the WS-Security header. (However, if a  
3722 signature element was included to cover the wsse:Timestamp, then the layout would need to be  
3723 considered.)

3724 A timestamp (P023) is required to be included in an acceptable message.

3725 Lines (P026)-(P034) contain an sp:SignedEndorsingSupportingTokens element, which means that the  
3726 contained supporting token (the SAML hk Assertion) references a key that will be “signing” (endorsing)  
3727 the message signature. The token itself may also be considered to be “signed”, because it is contained in  
3728 the message sent over the SSL link. In the case of sp:TransportBinding, there may be no actual  
3729 “message signature”, however, when a client certificate is used, the service can be assured that the  
3730 connection was set up by the client and that the SSL link guarantees the integrity of the data that is sent  
3731 on the link by the client, while it is on the link and when it is received from the link by using the key  
3732 referenced by the token. However, it does not guarantee the integrity after the data is received (i.e. after it  
3733 is received there is no way to tell whether any changes have been made to it since it has been received).  
3734 In any event, within this context a Signed Endorsing Supporting Token can be used to tell the Recipient  
3735 that the Issuer of the token is making claims related to the holder of the private key that is referenced by  
3736 the token, which in this case would be the private key associated with the client certificate used to set up  
3737 the SSL link, as described further below.

3738 Lines (P028)-(P032) contain an sp:SamlToken, which must always be sent to the Recipient.

3739 Line (P030) specifies that the token is of type WssSamlV20Token11, which means that the Endorsing  
3740 Supporting Token must be a SAML Version 2.0 token and it must be sent using WS-Security 1.1 using  
3741 the [[WSS11-SAML1120-PROFILE](#)]. Note that because the SamlToken is contained in an  
3742 sp:EndorsingSupportingTokens element, that it implicitly must be a holder-of-key token as described in  
3743 section 2.3 above.

3744 Here is an example request taken from the WSS SAML Profile InterOp [[WSS11-SAML1120-INTEROP](#)  
3745 Scenario #5] containing only minor cosmetic modifications and corrections.

3746

```

3747      (M001)      <?xml version="1.0" encoding="utf-8" ?>
3748      (M002)      <S11:Envelope
3749      (M003)          xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
3750      (M004)          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3751      (M005)          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3752      (M006)          xmlns:wsu=".../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3753      (M007)          xmlns:wsse=".../oasis-200401-wss-wssecurity-secext-1.0.xsd"
3754      (M008)          xmlns:wssell=".../oasis-wss-wssecurity-secext-1.1.xsd"

```

```

3755 (M009)      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3756 (M010)      xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
3757 (M011)      <S11:Header>
3758 (M012)      <wsse:Security S11:mustUnderstand="1">
3759 (M013)      <wsu:Timestamp>
3760 (M014)      <wsu:Created>2005-03-18T19:53:13Z</wsu:Created>
3761 (M015)      </wsu:Timestamp>
3762 (M016)      <saml2:Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3763 (M017)      IssueInstant="2005-04-17T00:46:02Z" Version="2.0">
3764 (M018)      <saml2:Issuer>www.opensaml.org</saml2:Issuer>
3765 (M019)      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3766 (M020)      <ds:SignedInfo>
3767 (M021)      <ds:CanonicalizationMethod
3768 (M022)      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3769 (M023)      <ds:SignatureMethod
3770 (M024)      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3771 (M025)      <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3772 (M026)      <ds:Transforms>
3773 (M027)      <ds:Transform Algorithm=
3774 (M028)      "http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
3775 (M029)      <ds:Transform Algorithm=
3776 (M030)      "http://www.w3.org/2001/10/xml-exc-c14n#">
3777 (M031)      <InclusiveNamespaces
3778 (M032)      PrefixList="#default saml ds xs xsi"
3779 (M033)      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3780 (M034)      </ds:Transform>
3781 (M035)      </ds:Transforms>
3782 (M036)      <ds:DigestMethod Algorithm=
3783 (M037)      "http://www.w3.org/2000/09/xmldsig#sha1" />
3784 (M038)      <ds:DigestValue
3785 (M039)      >Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3786 (M040)      </ds:Reference>
3787 (M041)      </ds:SignedInfo>
3788 (M042)      <ds:SignatureValue>
3789 (M043)      hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3790 (M044)      7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmtp3TD
3791 (M045)      MwuL/cBUj20tBZQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3792 (M046)      </ds:SignatureValue>
3793 (M047)      <ds:KeyInfo>
3794 (M048)      <ds:X509Data>
3795 (M049)      <ds:X509Certificate>
3796 (M050)      MIIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxkCzAJBgNVBAYTAlVT
3797 (M051)      ...
3798 (M052)      8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREG8cCx3w/w==
3799 (M053)      </ds:X509Certificate>
3800 (M054)      </ds:X509Data>
3801 (M055)      </ds:KeyInfo>
3802 (M056)      </ds:Signature>
3803 (M057)      <saml2:Conditions NotBefore="2005-06-19T16:53:33.173Z"
3804 (M058)      NotOnOrAfter="2006-06-19T17:08:33.173Z" />
3805 (M059)      <saml2:Subject>
3806 (M060)      <saml2:NameID
3807 (M061)      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
3808 (M062)      >uid=joe,ou=people,ou=saml-demo,o=example.com</saml2:NameID>
3809 (M063)      <saml2:SubjectConfirmation
3810 (M064)      Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key" />
3811 (M065)      <saml2:SubjectConfirmationData
3812 (M066)      xsi:type="saml2:KeyInfoConfirmationDataType">
3813 (M067)      <ds:KeyInfo>
3814 (M068)      <ds:X509Data>
3815 (M069)      <ds:X509IssuerName>
3816 (M070)      C=ZA, ST=Western Cape, L=Cape Town,
3817 (M071)      O=Thawte Consulting cc,
3818 (M072)      OU=Certification Services Division,

```

```

3819 (M073)          CN=Thawte Server CA/Email=server-certs@thawte.com
3820 (M074)          </ds:X509IssuerName>
3821 (M075)          <X509SerialNumber>12345678</X509SerialNumber>
3822 (M076)          </ds:X509Data>
3823 (M077)          </ds:KeyInfo>
3824 (M078)          </saml2:SubjectConfirmationData>
3825 (M079)          </saml2:Subject>
3826 (M080)          <saml2:AttributeStatement>
3827 (M081)          <saml2:Attribute Name="MemberLevel">
3828 (M082)          <saml2:AttributeValue>gold</saml2:AttributeValue>
3829 (M083)          </saml2:Attribute>
3830 (M084)          <saml2:Attribute Name="E-mail">
3831 (M085)          <saml2:AttributeValue
3832 (M086)          >joe@yahoo.com</saml2:AttributeValue>
3833 (M087)          </saml2:Attribute>
3834 (M088)          </saml2:AttributeStatement>
3835 (M089)          </saml2:Assertion>
3836 (M090)          </wsse:Security>
3837 (M091)          </S11:Header>
3838 (M092)          <S11:Body wsu:Id="MsgBody">
3839 (M093)          <ReportRequest>
3840 (M094)          <TickerSymbol>SUNW</TickerSymbol>
3841 (M095)          </ReportRequest>
3842 (M096)          </S11:Body>
3843 (M097)          </S11:Envelope>

```

3844 Lines (M002)-(M097) contain the SOAP:Envelope, which contains the SOAP:Header (M011)-(M091) and  
3845 the SOAP:Body (M092)-(M096).

3846 Lines (M012)-(M090) contain the wsse:Security header, which contains a wsu:Timestamp (M013)-(M015)  
3847 and a saml2:Assertion (M016)-(M089).

3848 The wsu:Timestamp (M013)-(M015) may be considered to be virtually signed by the client certificate as  
3849 explained above and in section 2.3.

3850 The saml2:Assertion was issued by an IssuingAuthority identified in the saml2:Issuer element (M018).

3851 The saml2:Issuer has used the private key of its enterprise certificate to produce the ds:Signature  
3852 element (M019)-(M056) that is an enveloped-signature (M028) that covers its parent XML element, the  
3853 saml2:Assertion (M016)-(M089). The ds:KeyInfo (M047)-(M055) within this ds:Signature contains an  
3854 actual copy of the Issuer's enterprise certificate, that the Recipient can verify for authenticity to establish  
3855 that this message is in conformance with any agreement the RelyingParty may have with the  
3856 IssuingAuthority. The details of this verification are outside the scope of this document, however they  
3857 involve the structures and systems the RelyingParty has in place recognize and verify certificates and  
3858 signatures it receives that are associated with business arrangements with the holders of the private keys  
3859 of those certificates.

3860 The saml2:Subject of the saml2:Assertion is contained in lines (M059)-(M079). Within the saml2:Subject  
3861 is the saml2:NameID (M060)-(M062), which contains the official name of the Subject of this Assertion and  
3862 this entity may be considered to the Requestor associated with this request.

3863 The saml2:SubjectConfirmation (M063)-(M064) has Method "holder-of-key" which implies that there is a  
3864 saml2:SubjectConfirmationData element (M065)-(M078) which will contain information that identifies a  
3865 signing key that the Initiator will use to bind this saml2:Assertion to a message that is associated with the  
3866 Requestor. In this context, the Initiator is acting as "attesting entity" with respect to the Subject as defined  
3867 in [SAML20], which means that the Initiator is authorized by the IssuingAuthority to present  
3868 saml2:Assertions pertaining to saml2:Subjects to Recipients/RelyingParties. In this example there is a  
3869 ds:KeyInfo (M067)-(M077) that identifies a specific certificate (ds:X509IssuerName (M069)-(M074) and  
3870 ds:SerialNumber (M075)) that the Initiator must prove possession of the associated private key to verify  
3871 this message. In this policy that private key is the one associated with the client certificate that the Initiator  
3872 is required to use (P008).

3873 The saml2:AttributeStatement (M080)-(M088) contains some information about the Subject that is  
3874 officially being provided in this saml2:Assertion by the IssuingAuthority that may have some significance  
3875 to the Relying Party in terms of determining whether access is granted for this request.

3876

3877 **2.3.2.4 (WSS1.1) SAML1.1/2.0 Sender Vouches with X.509 Certificate, Sign,**  
3878 **Encrypt**

3879 Here the message and SAML Assertion are signed using a key derived from the ephemeral key K. The  
3880 ephemeral key is encrypted using the Recipient's public key for the request input message and the same  
3881 shared ephemeral key, K, is encrypted using the Initiators public key for the response output message.

3882 Alternatively, derived keys can be used for each of signing and encryption operations.

3883 In this scenario the Authority is the Initiator who signs the message with the generated key. In order to  
3884 establish trust in the generated key, the Initiator must sign the message signature with a second signature  
3885 using an X509 certificate, which is indicated as the EndorsingSupportingToken. This X509 certificate  
3886 establishes the Initiator as the SAML Authority.

3887 (Note: there are instructive similarities and differences between this example and example 2.3.1.4, where  
3888 the difference is that: here the binding is symmetric and the WSS1.1 is used, whereas in 2.3.1.4 the  
3889 binding is asymmetric and WSS1.0 is used. One particular item is that in 2.3.1.4 an  
3890 EndorsingSupportingToken is not needed because in 2.3.1.4 the asymmetric binding uses X.509  
3891 certificates, which may be inherently trusted as opposed to the ephemeral key, K, used here.)

3892

```
3893 (P001) <wsp:Policy wsu:Id="WSS11SamlWithCertificates_policy">
3894 (P002)   <wsp:ExactlyOne>
3895 (P003)   <wsp>All>
3896 (P004)     <sp:SymmetricBinding>
3897 (P005)       <wsp:Policy>
3898 (P006)         <sp:ProtectionToken>
3899 (P007)           <wsp:Policy>
3900 (P008)             <sp:X509Token sp:IncludeToken=
3901 "http://docs.oasis-open.org/ws-sx/ws-
3902 securitypolicy/200702/IncludeToken/Never">
3903 (P009)               <wsp:Policy>
3904 (P010)                 <sp:RequireThumbprintReference/>
3905 (P011)                 <sp:RequireDerivedKeys wsp:Optional="true"/>
3906 (P012)                 <sp:WssX509V3Token10/>
3907 (P013)               </wsp:Policy>
3908 (P014)             </sp:X509Token>
3909 (P015)           </wsp:Policy>
3910 (P016)         </sp:ProtectionToken>
3911 (P017)       <sp:AlgorithmSuite>
3912 (P018)         <wsp:Policy>
3913 (P019)           <sp:Basic256/>
3914 (P020)         </wsp:Policy>
3915 (P021)       </sp:AlgorithmSuite>
3916 (P022)     <sp:Layout>
3917 (P023)       <wsp:Policy>
3918 (P024)         <sp:Strict/>
3919 (P025)       </wsp:Policy>
3920 (P026)     </sp:Layout>
3921 (P027)   <sp:IncludeTimestamp/>
3922 (P028)   <sp:OnlySignEntireHeadersAndBody/>
3923 (P029) </wsp:Policy>
3924 (P030) </sp:SymmetricBinding>
3925 (P031) <sp:SignedSupportingTokens>
3926 (P032)   <wsp:Policy>
3927 (P033)     <sp:Sam1Token sp:IncludeToken=
3928 "http://docs.oasis-open.org/ws-sx/ws-
3929 securitypolicy/200702/IncludeToken/AlwaysToRecipient">
3930 (P034)       <wsp:Policy>
3931 (P035)         <sp:WssSam1V11Token11/>
3932 (P036)       </wsp:Policy>
3933 (P037)     </sp:Sam1Token>
3934 (P038)   </wsp:Policy>
3935 (P039) </sp:SignedSupportingTokens>
```

```

3936 (P040) <sp:EndorsingSupportingTokens>
3937 (P041) <wsp:Policy>
3938 (P042) <sp:X509Token sp:IncludeToken="AlwaysToRecipient">
3939 (P043) <wsp:Policy>
3940 (P044) <sp:WssX509V3Token11/>
3941 (P045) </wsp:Policy>
3942 (P046) </sp:X509Token>
3943 (P047) </wsp:Policy>
3944 (P048) </sp:EndorsingSupportingTokens>
3945 (P049) <sp:Wss11>
3946 (P050) <wsp:Policy>
3947 (P051) <sp:MustSupportRefKeyIdentifier/>
3948 (P052) <sp:MustSupportRefIssuerSerial/>
3949 (P053) <sp:MustSupportRefThumbprint/>
3950 (P054) <sp:MustSupportRefEncryptedKey/>
3951 (P055) </wsp:Policy>
3952 (P056) </sp:Wss11>
3953 (P057) </wsp:All>
3954 (P058) </wsp:ExactlyOne>
3955 (P059) </wsp:Policy>
3956 (P060)
3957 (P061) <wsp:Policy wsu:Id="SamlForCertificates_input_policy">
3958 (P062) <wsp:ExactlyOne>
3959 (P063) <wsp:All>
3960 (P064) <sp:SignedParts>
3961 (P065) <sp:Body/>
3962 (P066) </sp:SignedParts>
3963 (P067) <sp:EncryptedParts>
3964 (P068) <sp:Body/>
3965 (P069) </sp:EncryptedParts>
3966 (P070) </wsp:All>
3967 (P071) </wsp:ExactlyOne>
3968 (P072) </wsp:Policy>
3969 (P073)
3970 (P074) <wsp:Policy wsu:Id="SamlForCertificate_output_policy">
3971 (P075) <wsp:ExactlyOne>
3972 (P076) <wsp:All>
3973 (P077) <sp:SignedParts>
3974 (P078) <sp:Body/>
3975 (P079) </sp:SignedParts>
3976 (P080) <sp:EncryptedParts>
3977 (P081) <sp:Body/>
3978 (P082) </sp:EncryptedParts>
3979 (P083) </wsp:All>
3980 (P084) </wsp:ExactlyOne>
3981 (P085) </wsp:Policy>

```

3982 Lines (P001)-(P059) contain the policy that requires all the contained assertions to be complied with by  
3983 the Initiator. In this case there are 4 assertions: sp:SymmetricBinding (P004)-(P030),  
3984 sp:SignedSupportingTokens (P031)-(P039), sp:EndorsingSupportingTokens (P040)-(P048), and  
3985 sp:Wss11 (P049)-(P056).

3986 The sp:SymmetricBinding assertion (P004) contains an sp:ProtectionToken (P006)-(P016), which  
3987 indicates that both the Initiator and Recipient must use each other's public key respectively associated  
3988 with the X509Token (P008)-(P014) associated with the respective message receiver to encrypt the  
3989 shared ephemeral symmetric key as explained at the beginning of this section. The messages may either  
3990 be encrypted and signed using keys derived from the ephemeral key as indicated by the  
3991 sp:RequireDerivedKeys assertion (P011) or the encryption and signing can be done using the ephemeral  
3992 key itself without deriving keys, because the RequireDerivedKey assertion is an optional requirement as  
3993 indicated by the wsp:Optional attribute on line (P011).

3994 The sp:SignedSupportingTokens assertion (P031) contains an sp:SamlToken assertion (P033)-(P037),  
3995 which indicates that a signed SAML Assertion must always be included in the Initiator's request to the  
3996 Recipient (AlwaysToRecipient (P033)). This SAML Assertion must be included in the WS-Security header

3997 and referenced and signed as described in the WS-Security 1.1 Profile for SAML [\[WSS11-SAML1120-](#)  
3998 [PROFILE\]](#) as indicated by the sp:WssSamlV11Token11 assertion (P036), which indicates the SAML 1.1  
3999 option in that profile (as opposed to the SAML 2.0 option, which would have been indicated by  
4000 sp:WssSamlV20Token11).

4001 The sp:EndorsingSupportingToken assertion (P040) is needed because there are no guarantees that the  
4002 ephemeral key, K, is still being used by the Initiator with whom the Recipient would have collaborated  
4003 originally to establish the ephemeral key as a shared secret. The purpose of the endorsing token is to  
4004 sign the signature made by the ephemeral key, using the Initiator's private key, which will explicitly  
4005 guarantee the content of this particular Initiator request. The endorsing token in this policy is an  
4006 sp:X509Token (P042)-(P046), which, in particular, is an sp:WssX509V3Token11, which must be used in  
4007 accordance with the WS-Security 1.1 Profile for X509 Tokens [\[WSS11-X509-PROFILE\]](#). (Note: it may be  
4008 the case that this X509 certificate is the same X509 certificate referred to in the ProtectionToken  
4009 assertion (P012). However, it is important to keep in mind that line (P012) only indicates that the Initiator's  
4010 token will be used by the Recipient to protect the ephemeral key for the symmetric binding. Therefore, the  
4011 fact that the token is identified in line (P012) as an sp:X509V3Token10 is not directly related to the fact  
4012 that the same key may be used for the additional purpose of explicitly signing the request message).

4013 The sp:Wss11 assertion (P049-P056) indicates that WS-Security 1.1 constructs are accepted. (Note also  
4014 that eitherWssX509V3Token10 or WssX509V3Token11 may be used with the Wss11 since both WS-  
4015 Security 1.0 and WS-Security 1.1 Profiles are supported by WS-Security 1.1)

4016 There are also 2 Policys, one each for the input message and the output message, each of which  
4017 contains an assertion indicating the message SOAP Body must be signed (P064)-(P066) for the input  
4018 message and (P077)-(P079) for the output message, and each contains an assertion that the message  
4019 SOAP Body must be encrypted (P067)-(P069) for the input message and (P080)-(P082) for the output  
4020 message.

4021 The following is a sample request that is compliant with this policy.

```
4022 (M001) <?xml version="1.0" encoding="utf-8" ?>
4023 (M002) <S12:Envelope
4024 (M003)   xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
4025 (M004)   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4026 (M005)   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4027 (M006)   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4028 wss-wssecurity-secext-1.0.xsd"
4029 (M007)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4030 wssecurity-utility-1.0.xsd"
4031 (M008)   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
4032 (M009)   xmlns:xenc="..."
4033 (M010)   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
4034 (M011) <S12:Header>
4035 (M012) <wsse:Security S12:mustUnderstand="1">
4036 (M013) <wsu:Timestamp wsu:Id="timestamp">
4037 (M014) <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
4038 (M015) </wsu:Timestamp>
4039 (M016) <xenc:EncryptedKey Id="encKey" >
4040 (M017) <xenc:EncryptionMethod Algorithm="...#rsa-oaep-mgflp">
4041 (M018) <ds:DigestMethod Algorithm="...#sha1"/>
4042 (M019) </xenc:EncryptionMethod>
4043 (M020) <ds:KeyInfo >
4044 (M021) <wsse:SecurityTokenReference >
4045 (M022) <wsse:KeyIdentifier EncodingType="...#Base64Binary"
4046 Value="...#ThumbprintSHA1">c2...=</wsse:KeyIdentifier>
4047 (M023) </wsse:SecurityTokenReference>
4048 (M024) </ds:KeyInfo>
4049 (M025) <xenc:CipherData>
4050 (M026) <xenc:CipherValue>TE...=</xenc:CipherValue>
4051 (M027) </xenc:CipherData>
4052 (M028) </xenc:EncryptedKey>
4053 (M029) <n1:ReferenceList xmlns:n1=".../xmlenc#">
4054 (M030) <n1:DataReference URI="#encBody"/>
4055 (M031) </n1:ReferenceList>
```

```

4056 (M032) <saml:Assertion
4057 (M033)   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
4058 (M034)   IssueInstant="2003-04-17T00:46:02Z"
4059 (M035)   Issuer="www.opensaml.org"
4060 (M036)   MajorVersion="1"
4061 (M037)   MinorVersion="1">
4062 (M038)   <saml:Conditions
4063 (M039)     NotBefore="2002-06-19T16:53:33.173Z"
4064 (M040)     NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
4065 (M041)   <saml:AttributeStatement>
4066 (M042)     <saml:Subject>
4067 (M043)       <saml:NameIdentifier
4068 (M044)         NameQualifier="www.example.com"
4069 (M045)         Format="">
4070 (M046)         uid=joe,ou=people,ou=saml-demo,o=example.com
4071 (M047)       </saml:NameIdentifier>
4072 (M048)     <saml:SubjectConfirmation>
4073 (M049)       <saml:ConfirmationMethod>
4074 (M050)         urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
4075 (M051)       </saml:ConfirmationMethod>
4076 (M052)     </saml:SubjectConfirmation>
4077 (M053)   </saml:Subject>
4078 (M054)   <saml:Attribute>
4079 (M055)     ...
4080 (M056)   </saml:Attribute>
4081 (M057)     ...
4082 (M058)   </saml:AttributeStatement>
4083 (M059) </saml:Assertion>
4084 (M060) <wsse:SecurityTokenReference wsu:id="STR1">
4085 (M061)   <wsse:KeyIdentifier wsu:id="..."
4086 (M062)     ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
4087 (M063)     profile-1.0#SAMLAssertionID">
4088 (M064)       _a75adf55-01d7-40cc-929f-dbd8372ebdfc
4089 (M064)     </wsse:KeyIdentifier>
4090 (M065)   </wsse:SecurityTokenReference>
4091 (M066) <wsse:BinarySecurityToken
4092 (M067)   wsu:Id="attesterCert"
4093 (M068)   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4094 (M069)   wss-x509-token-profile-1.0#X509v3"
4095 (M069)   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
4096 (M069)   200401-wss-soap-message-security-1.0#Base64Binary">
4097 (M070)   MIEZzCCA9CgAwIBAgIQEmtJZc0...
4098 (M071) </wsse:BinarySecurityToken>
4099 (M072) <ds:Signature wsu:Id="message-signature">
4100 (M073)   <ds:SignedInfo>
4101 (M074)     <ds:CanonicalizationMethod
4102 (M075)       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4103 (M076)     <ds:SignatureMethod
4104 (M077)       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4105 (M078)     <ds:Reference URI="#STR1">
4106 (M079)       <ds:Transforms>
4107 (M080)         <ds:Transform
4108 (M081)           Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
4109 (M081)           200401-wss-soap-message-security-1.0#STR-Transform">
4110 (M082)         <wsse:TransformationParameters>
4111 (M083)           <ds:CanonicalizationMethod
4112 (M084)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
4113 (M085)           </wsse:TransformationParameters>
4114 (M086)         </ds:Transform>
4115 (M087)       </ds:Transforms>
4116 (M088)     <ds:DigestMethod
4117 (M089)       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
4118 (M090)     <ds:DigestValue>...</ds:DigestValue>
4119 (M091)   </ds:Reference>

```

```

4120 (M092) <ds:Reference URI="#MsgBody">
4121 (M093) <ds:DigestMethod
4122 (M094) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4123 (M095) <ds:DigestValue>...</ds:DigestValue>
4124 (M096) </ds:Reference>
4125 (M097) <ds:Reference URI="#timestamp">
4126 (M098) <ds:DigestMethod
4127 (M099) Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
4128 (M100) <ds:DigestValue>...</ds:DigestValue>
4129 (M101) </ds:Reference>
4130 (M102) </ds:SignedInfo>
4131 (M103) <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
4132 (M104) <ds:KeyInfo>
4133 (M105) <wsse:SecurityTokenReference wsu:id="STR2">
4134 (M106) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
4135 (M107) </wsse:SecurityTokenReference>
4136 (M108) </ds:KeyInfo>
4137 (M109) </ds:Signature>
4138 (M110) <ds:Signature wsu:Id="endorsing-signature">
4139 (M111) <ds:SignedInfo>
4140 (M112) ...
4141 (M113) <ds:Reference URI="#message-signature">
4142 (M114) ...
4143 (M115) </ds:Reference>
4144 (M116) </ds:SignedInfo>
4145 (M117) <ds:SignatureValue>Bu ...=</ds:SignatureValue>
4146 (M118) <ds:KeyInfo>
4147 (M119) <wsse:SecurityTokenReference >
4148 (M120) <wsse:Reference URI="#attesterCert" ValueType="...#X509v3"/>
4149 (M121) </wsse:SecurityTokenReference>
4150 (M122) </ds:KeyInfo>
4151 (M123) </ds:Signature>
4152 (M124) </wsse:Security>
4153 (M125) </S12:Header>
4154 (M126) <S12:Body wsu:Id="MsgBody">
4155 (M127) <xenc:EncryptedData Id="encBody" Type="...#Content"
4156 MimeType="text/xml" Encoding="UTF-8" >
4157 (M128) <xenc:EncryptionMethod Algorithm="http...#aes256-cbc"/>
4158 (M129) <ds:KeyInfo >
4159 (M130) <wsse:SecurityTokenReference>
4160 (M131) <wsse:Reference URI="#enckey" ValueType="../EncryptedKey"/>
4161 (M132) </wsse:SecurityTokenReference>
4162 (M133) </ds:KeyInfo>
4163 (M134) <xenc:CipherData>
4164 (M135) <xenc:CipherValue>70...=</xenc:CipherValue>
4165 (M136) </xenc:CipherData>
4166 (M137) </xenc:EncryptedData>
4167 (M138) </S12:Body>
4168 (M139) </S12:Envelope>

```

4169

4170 The message above contains a request compliant with the policy described in (P001)-(P072), which  
4171 includes the endpoint policy and the input policy.

4172 Lines (M016)-(M028) contain an xenc:EncryptedKey element that contains an Initiator-generated  
4173 symmetric signing and encryption key, K, that can be used to decrypt the xenc:EncryptedData contained  
4174 in the SOAP S12:Body (M0126) as indicated by the wsse:SecurityTokenReference (M0130)-(M0132) that  
4175 uses a direct reference to the Id of the xenc:EncryptedKey, "encKey" on lines (M0131) and (M016).

4176 The xenc:EncryptedKey contains a dsig KeyInfo (M020)-(M024) reference to the Thumbprint of the  
4177 Recipient's public X509 certificate (M022). This complies with the policy (P068) that requires the sp:Body  
4178 to of the input message to be encrypted. It also complies with the policy (P013) to protect using  
4179 encryption based on X509 token and to refer to it by Thumbprint (P014).

4180 Lines (M029)-(M031) contain an xenc:ReferenceList referring to the xenc:EncryptedData in the S12:Body  
4181 (M0127)-(M0137).

4182 Lines (M032)-(M059) contain the saml:Assertion as a SignedSupportingToken as required by the  
4183 endpoint policy (P035). A saml:Assertion used as a SignedSupportingToken uses the “sender-vouches”  
4184 ConfirmationMethod (M049)-(M051) as described in the introductory section 2.3.

4185 Lines (M060)-(M065) contain a WS-Security wsse:SecurityTokenReference that has a KeyIdentifier of  
4186 ValueType “...1.0#SAMLAssertionID”, which indicates a SAML 1.1 Assertion as described in the WS-  
4187 Security 1.1 [[SAML1120\\_TOKEN\\_PROFILE](#)].

4188 Lines (M066)-(M071) contain a wsse:BinarySecurityToken that contains the Initiator’s X509 certificate for  
4189 use as an EndorsingSupportingToken as required by line (P042) of the sp:EndorsingSupportingTokens  
4190 assertion (P040)-(P048).

4191 Lines (M072)-(M0109) contain the message signature. The dsig Signature covers the following:

- 4192 ➤ The SAML Assertion using the ds:Reference (M078)-(M091), which uses the WS-Security STR-  
4193 Transform technique to refer to the SAML Assertion indirectly through the  
4194 wsse:SecurityTokenReference (M060) described above. This signature is required by the  
4195 sp:SignedSupportingTokens assertion (P031)-(P039).
- 4196 ➤ The message sp:Body (M0126)-(M0138) using the ds:Reference (M092)-(M096) as required by  
4197 the input message policy (P065).
- 4198 ➤ The message wsu:Timestamp (M013)-(M015) using the ds:Reference (M097)-(M0101) as  
4199 required by the endpoint policy (P027).

4200 The key used to sign the message signature is referenced in the dsig KeyInfo (M0104)-(M0108), which  
4201 contains a SecurityTokenReference with a direct URI reference to the xenc:EncryptedKey, “encKey”,  
4202 which contains the Initiator-generated signing and encryption key, K, as described above.

4203 Lines (M0110)-(M0123) contain the endorsing signature. The dsig endorsing Signature covers the  
4204 following:

- 4205 ➤ The message Signature (M072)-(M0109) using the ds:Reference (M0113)-(M0115), which is a  
4206 direct URI reference to the Id “message-signature” on line (M072).

4207 This signature is required by the policy EndorsingSupportingTokens assertion (P040)-(P048) to be an  
4208 X509 certificate (P044). The dsig KeyInfo (M0118)-(M0122) contains a WS-Security  
4209 SecurityTokenReference with a direct URI to the Initiator’s X509 certificate on line (M066)-(M071) with Id  
4210 = “attesterCert”.

4211 Lines (M0127)-(M0137) contain the xenc:EncryptedData, which contains the SOAP S12:Body message  
4212 payload that is encrypted using the encryption key, K, contained in the xenc:EncryptedKey CipherValue  
4213 (M026), which can only be decrypted using the Recipient’s X509 certificate referred to by  
4214 ThumbprintSHA1 on line (M022).

### 4215 **2.3.2.5 (WSS1.1) SAML1.1/2.0 Holder of Key, Sign, Encrypt**

4216 This scenario is based on WS-SX Interop Scenarios Phase 2 (October 31, 2006) [[WSSX-WSTR-WSSC-](#)  
4217 [INTEROP](#)] Scenario 5 (Client and STS: Mutual Certificate WSS1.1 (section 3.5 of interop ref), Client and  
4218 Service: Issued SAML 1.1 Token for Certificate WSS1.1 (section 4.3 of interop ref)).

4219 In this scenario, the service specifies that the client must obtain an IssuedToken from a designated  
4220 Security Token Service (STS), which must be a SAML 1.1 Assertion. The Assertion contains an  
4221 ephemeral key K2 in an EncryptedKey element encrypted using service’s certificate. The client also  
4222 obtains the same ephemeral key K2 from the RequestedProofToken returned by the STS. The body of  
4223 the message from the client to the service is signed using DKT1(K2), encrypted using DKT2(K2), and  
4224 endorsed using DKT3(K2), which are keys the client derives from K2 using the algorithm specified in  
4225 section 7 of [[WS-SecureConversation](#)]. The response from the service is also signed using derived keys.  
4226 In a simpler alternative, ephemeral key K itself could be used for message protection.

4227 Note: In this scenario, in terms of Figure 1 [[Figure01](#)], the STS (Issuing Authority) is the  
4228 Issuer of the SAML 1.1 holder-of-key Assertion that contains the ephemeral symmetric  
4229 key K2. The service (combined Recipient/RelyingParty) can trust the client (combined

4230 Initiator/Requestor) that uses the ephemeral symmetric key K2 obtained from the  
4231 RequestedProofToken, because the same key gets delivered to the service in the SAML  
4232 1.1 holder-of-key Assertion, which is signed by the trusted Authority.

4233 This scenario is a 2-step sequence from a WS-SP perspective. These 2 steps will be described at a high  
4234 level first in order to explain the context for the policies and messages that follow.

4235 **In step 1** the following takes place:

- 4236 • the client accesses the service's WS-SP policy (P001 -> P116 below), and determines that it  
4237 needs to use an IssuedToken from a Security Token Service (STS), specified in the policy.  
4238 This IssuedToken will serve as the basis of trust by the service. Effectively, the service only trusts  
4239 the client because the client is able to obtain the IssuedToken from the STS.

4240 To complete step 1, the client will then do the following:

- 4241 • the client will access the STS and process the STS policy (PSTS-001 -> PSTS-098 below)
- 4242 • based on the STS policy, the client will send a request to the STS for an IssuedToken (MSTS-001  
4243 -> MSTS-0231 below)
- 4244 • the STS will send a response to the client containing the IssuedToken, which in this example is a  
4245 SAML 1.1 holder-of-key Assertion (RSTS-001 -> RSTS-0263 below)

4246 **In step 2** the following takes place:

- 4247 • because the client now has the IssuedToken it is now able fulfill the requirements of the service's  
4248 WS-SP policy (P001-P116 below) that it accessed in step 1
- 4249 • based on the service's policy the clients sends a request to the service (M001-M0229 below)
- 4250 • the service will send a response to the client containing the requested resource data (R001 -  
4251 R0153 below)

4252 As an aid to understanding the security properties of the policies and messages in this example, the  
4253 following is a list of identifiers used in the text descriptions to reference the cryptographic keys that are  
4254 called for in the policies and used in the messages in this example:

- 4255 ➤ **X509T1**: Client's (Requestor/Initiator) X509 certificate used for authentication to the STS.
- 4256 ➤ **X509T2**: STS' (Issuing Authority) X509 certificate used to encrypt keys sent to STS, used to  
4257 authenticate STS to Service.
- 4258 ➤ **X509T3**: Service's (Recipient/RelyingParty/ValidatingAuthority) X509 certificate used to encrypt  
4259 keys sent to Service.
- 4260 ➤ **K1**: Client-generated ephemeral symmetric key for crypto communication to the STS.
- 4261 ➤ **K2**: Client-generated ephemeral symmetric key for crypto communication between the Client and  
4262 the Service.
- 4263 ➤ **K3**: STS-generated proof key for use by Client to authenticate with Service via saml:Assertion.
- 4264 ➤ **DKT1(K2)**: Client-generated derived key used for signing requests to the Service.
- 4265 ➤ **DKT2(K2)**: Client-generated derived key used for encrypting requests to the Service.
- 4266 ➤ **DKT3(K3)**: Client-generated derived key used for endorsing signed requests to the Service.
- 4267 ➤ **DKT4(K2)**: Service-generated derived key used for encrypting responses to the Client.
- 4268 ➤ **DKT5(K2)**: Service-generated derived key used for signing responses to the Client.

4269

4270 Here is the WS-SP Policy that the Service presents to a Client:

4271

```
4272 (P001) <wsp:Policy wsu:Id="Service5-Policy"  
4273 (P002)  
4274 (P003) xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4275 (P004) xmlns:sp"..."  
4276 (P005)
```

```

4277 (P006)      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4278 wssecurity-utility-1.0.xsd"
4279 (P007)      >
4280 (P008)      <wsp:ExactlyOne>
4281 (P009)      <wsp:All>
4282 (P010)      <sp:SymmetricBinding>
4283 (P011)      <wsp:Policy>
4284 (P012)      <sp:ProtectionToken>
4285 (P013)      <wsp:Policy>
4286 (P014)      <sp:X509Token IncludeToken=
4287 (P015)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4288 /Never">
4289 (P016)      <wsp:Policy>
4290 (P017)      <sp:RequireDerivedKeys/>
4291 (P018)      <sp:RequireThumbprintReference/>
4292 (P019)      <sp:WssX509V3Token10/>
4293 (P020)      </wsp:Policy>
4294 (P021)      </sp:X509Token>
4295 (P022)      </wsp:Policy>
4296 (P023)      </sp:ProtectionToken>
4297 (P024)      <sp:AlgorithmSuite>
4298 (P025)      <wsp:Policy>
4299 (P026)      <sp:Basic256/>
4300 (P027)      </wsp:Policy>
4301 (P028)      </sp:AlgorithmSuite>
4302 (P029)      <sp:Layout>
4303 (P030)      <wsp:Policy>
4304 (P031)      <sp:Strict/>
4305 (P032)      </wsp:Policy>
4306 (P033)      </sp:Layout>
4307 (P034)      <sp:IncludeTimestamp/>
4308 (P035)      <sp:OnlySignEntireHeadersAndBody/>
4309 (P036)      </wsp:Policy>
4310 (P037)      </sp:SymmetricBinding>
4311 (P038)      <sp:EndorsingSupportingTokens>
4312 (P039)      <wsp:Policy>
4313 (P040)      <sp:IssuedToken IncludeToken=
4314 (P041)      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken
4315 /AlwaysToRecipient">
4316 (P042)      <sp:Issuer>
4317 (P043)      <a:Address>http://example.com/STS</a:Address>
4318 (P044)      </sp:Issuer>
4319 (P045)      <sp:RequestSecurityTokenTemplate>
4320 (P046)      <t:TokenType
4321 (P047)      >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4322 1.1#SAMLV1.1</t:TokenType>
4323 (P048)      <t:KeyType
4324 (P049)      >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:KeyTy
4325 pe>
4326 (P050)      <t:KeySize>256</t:KeySize>
4327 (P051)      <t:CanonicalizationAlgorithm
4328 (P052)      >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorithm>
4329 (P053)      <t:EncryptionAlgorithm
4330 (P054)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4331 (P055)      <t:EncryptWith
4332 (P056)      >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4333 (P057)      <t:SignWith
4334 (P058)      >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4335 (P059)      </sp:RequestSecurityTokenTemplate>
4336 (P060)      <wsp:Policy>
4337 (P061)      <sp:RequireDerivedKeys/>
4338 (P062)      <sp:RequireInternalReference/>
4339 (P063)      </wsp:Policy>
4340 (P064)      </sp:IssuedToken>

```

```

4341 (P065)         </wsp:Policy>
4342 (P066)         </sp:EndorsingSupportingTokens>
4343 (P067)         <sp:Wss11>
4344 (P068)         <wsp:Policy>
4345 (P069)         <sp:MustSupportRefKeyIdentifier/>
4346 (P070)         <sp:MustSupportRefIssuerSerial/>
4347 (P071)         <sp:MustSupportRefThumbprint/>
4348 (P072)         <sp:MustSupportRefEncryptedKey/>
4349 (P073)         <sp:RequireSignatureConfirmation/>
4350 (P074)         </wsp:Policy>
4351 (P075)         </sp:Wss11>
4352 (P076)         <sp:Trust13>
4353 (P077)         <wsp:Policy>
4354 (P078)         <sp:MustSupportIssuedTokens/>
4355 (P079)         <sp:RequireClientEntropy/>
4356 (P080)         <sp:RequireServerEntropy/>
4357 (P081)         </wsp:Policy>
4358 (P082)         </sp:Trust13>
4359 (P083)         </wsp:All>
4360 (P084)         </wsp:ExactlyOne>
4361 (P085)         </wsp:Policy>
4362
4363 (P086)         <wsp:Policy wsu:Id="InOut-Policy"
4364 (P087)
4365 (P088)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4366 (P089)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4367 (P090)
4368 (P091)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4369 (P092)         wssecurity-utility-1.0.xsd">
4370 (P092)         <wsp:ExactlyOne>
4371 (P093)         <wsp:All>
4372 (P094)         <sp:SignedParts>
4373 (P095)         <sp:Body/>
4374 (P096)         <sp:Header Name="To"
4375 (P097)         Namespace="http://www.w3.org/2005/08/addressing"/>
4376 (P098)         <sp:Header Name="From"
4377 (P099)         Namespace="http://www.w3.org/2005/08/addressing"/>
4378 (P100)         <sp:Header Name="FaultTo"
4379 (P101)         Namespace="http://www.w3.org/2005/08/addressing"/>
4380 (P102)         <sp:Header Name="ReplyTo"
4381 (P103)         Namespace="http://www.w3.org/2005/08/addressing"/>
4382 (P104)         <sp:Header Name="MessageID"
4383 (P105)         Namespace="http://www.w3.org/2005/08/addressing"/>
4384 (P106)         <sp:Header Name="RelatesTo"
4385 (P107)         Namespace="http://www.w3.org/2005/08/addressing"/>
4386 (P108)         <sp:Header Name="Action"
4387 (P109)         Namespace="http://www.w3.org/2005/08/addressing"/>
4388 (P110)         </sp:SignedParts>
4389 (P111)         <sp:EncryptedParts>
4390 (P112)         <sp:Body/>
4391 (P113)         </sp:EncryptedParts>
4392 (P114)         </wsp:All>
4393 (P115)         </wsp:ExactlyOne>
4394 (P116)         </wsp:Policy>

```

4395  
4396 When a Client encounters the above Service Policy, it determines that it must obtain an IssuedToken  
4397 from the designated Issuer. After obtaining the IssuedToken, the client may send the request in  
4398 accordance with the rest of the policy. Details of the Service Policy follow. The STS Policy its details  
4399 follow after the Service Policy.

4400 Lines (P001)-(P085) above contain the Service Endpoint wsp:Policy, which contains a wsp:All assertion  
4401 that requires all 4 of its contained assertions to be complied with: sp:SymmetricBinding (P010)-(P037),  
4402 sp:EndorsingSupportingTokens (P038)-(P066), sp:Wss11 (P067)-(P075), and sp:Trust13 (P076)-(P082).

4403 Lines (P010)-(P037) contain the **sp:SymmetricBinding assertion** that requires that derived keys (DKT1,  
4404 DKT2, DKT3) be used to protect the message (P017) and that the ephemeral key (K2) used to derive  
4405 these keys be encrypted using the service's X509 certificate (X509T3) as specified by the sp:X509Token  
4406 assertion (P014)-(P021), which also indicates that the X509 token, itself should not be sent (P015), but  
4407 that a Thumbprint reference (P018) to it be sent. Finally, the sp:SymmetricBinding specifies that the  
4408 sp:Basic256 sp:AlgorithmSuite be used (P024)-(P028), that sp:Strict sp:Layout be used (P029)-(P033),  
4409 that an sp:Timestamp be included (P034), and that only the complete message Body and Headers be  
4410 signed (P035), where the Headers part means that only direct child elements of the WS-Security SOAP  
4411 header element be signed.

4412 Lines (P038)-(P066) contain the **sp:EndorsingSupportingTokens assertion** that an sp:IssuedToken  
4413 (P040)-(P064) be used to sign the message signature and that the IssuedToken must be included with  
4414 the request (P040)-(P041). Lines (P042)-(P044) specify the address of the SecurityTokenService (STS)  
4415 from which the IssuedToken must be obtained. Lines (P045)-(P059) contain an  
4416 sp:RequestSecurityTokenTemplate (P046)-(P058) which contains explicit WS-Trust elements that the  
4417 client should directly copy to a t:SecondaryParameters element to include with the WS-Trust  
4418 t:RequestSecurityToken to the STS to obtain the sp:IssuedToken. Of particular interest here is that the  
4419 t:TokenType (P046)-(P047) requested is a SAML 1.1 Assertion, which will be used to contain the  
4420 ephemeral symmetric key (K2) (P048)-(P049). K2 will be used for communication between the Client and  
4421 the Service. K2 will be encrypted by the STS using the Service's X509 certificate (X509T3). The Client is  
4422 also informed by the IssuedToken assertion that the IssuedToken may only be referenced internally  
4423 within the message (P062) and that the ephemeral key (K2) associated with the IssuedToken be used by  
4424 the Client to derive the keys (DKT1(K2), DKT2(K2), DKT3(K2)) used in the Client's request to the Service.

4425 Lines (P067)-(P075) contain the **sp:Wss11 assertion** that indicates that WS-Security 1.1 will be used,  
4426 which includes Wss11-only features such as Thumbprint (P073), EncryptedKey (P074), and  
4427 SignatureConfirmation (P075).

4428 Lines (P076)-(P082) contain the **sp:Trust13 assertion** that indicates the Client should expect to use WS-  
4429 Trust 1.3 ([WSTRUST](#)) to obtain the IssuedToken from the STS.

4430 Lines (P086)-(P0116) contain the **operation input and output policies** that the client should use to  
4431 determine what parts of the messages are to be signed (P094)-(P0110) and encrypted (P0111)-(P0113).

4432

4433 Here is the WS-SP Policy that the STS presents to a Client:

```
4434 (PSTS-001) <wsp:Policy wsu:Id="STS5-Policy"  
4435 (PSTS-002)  
4436 (PSTS-003)   xmlns:wsp="http://www.w3.org/ns/ws-policy"  
4437 (PSTS-004)   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"  
4438 (PSTS-005)  
4439 (PSTS-006)   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-  
4440 wss-wssecurity-utility-1.0.xsd"  
4441 (PSTS-007)   >  
4442 (PSTS-008)   <wsp:ExactlyOne>  
4443 (PSTS-009)     <wsp>All>  
4444 (PSTS-010)       <sp:SymmetricBinding>  
4445 (PSTS-011)         <wsp:Policy>  
4446 (PSTS-012)           <sp:ProtectionToken>  
4447 (PSTS-013)             <wsp:Policy>  
4448 (PSTS-014)               <sp:X509Token IncludeToken=  
4449 (PSTS-015) "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo  
4450 ken/Never">  
4451 (PSTS-016)                 <wsp:Policy>  
4452 (PSTS-017)                   <sp:RequireThumbprintReference/>  
4453 (PSTS-018)                   <sp:WssX509V3Token10/>  
4454 (PSTS-019)                 </wsp:Policy>  
4455 (PSTS-020)               </sp:X509Token>  
4456 (PSTS-021)             </wsp:Policy>  
4457 (PSTS-022)           </sp:ProtectionToken>  
4458 (PSTS-023)         <sp:AlgorithmSuite>  
4459 (PSTS-024)           <wsp:Policy>  
4460 (PSTS-025)             <sp:Basic256/>
```

```

4461 (PSTS-026)         </wsp:Policy>
4462 (PSTS-027)         </sp:AlgorithmSuite>
4463 (PSTS-028)         <sp:Layout>
4464 (PSTS-029)         <wsp:Policy>
4465 (PSTS-030)         <sp:Strict/>
4466 (PSTS-031)         </wsp:Policy>
4467 (PSTS-032)         </sp:Layout>
4468 (PSTS-033)         <sp:IncludeTimestamp/>
4469 (PSTS-034)         <sp:OnlySignEntireHeadersAndBody/>
4470 (PSTS-035)         </wsp:Policy>
4471 (PSTS-036)         </sp:SymmetricBinding>
4472 (PSTS-037)         <sp:EndorsingSupportingTokens>
4473 (PSTS-038)         <wsp:Policy>
4474 (PSTS-039)         <sp:X509Token IncludeToken=
4475 (PSTS-040)         "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeTo
4476 (PSTS-040)         ken/AlwaysToRecipient">
4477 (PSTS-041)         <wsp:Policy>
4478 (PSTS-042)         <sp:RequireThumbprintReference/>
4479 (PSTS-043)         <sp:WssX509V3Token10/>
4480 (PSTS-044)         </wsp:Policy>
4481 (PSTS-045)         </sp:X509Token>
4482 (PSTS-046)         </wsp:Policy>
4483 (PSTS-047)         </sp:EndorsingSupportingTokens>
4484 (PSTS-048)         <sp:Wss11>
4485 (PSTS-049)         <wsp:Policy>
4486 (PSTS-050)         <sp:MustSupportRefKeyIdentifier/>
4487 (PSTS-051)         <sp:MustSupportRefIssuerSerial/>
4488 (PSTS-052)         <sp:MustSupportRefThumbprint/>
4489 (PSTS-053)         <sp:MustSupportRefEncryptedKey/>
4490 (PSTS-054)         <sp:RequireSignatureConfirmation/>
4491 (PSTS-055)         </wsp:Policy>
4492 (PSTS-056)         </sp:Wss11>
4493 (PSTS-057)         <sp:Trust13>
4494 (PSTS-058)         <wsp:Policy>
4495 (PSTS-059)         <sp:MustSupportIssuedTokens/>
4496 (PSTS-060)         <sp:RequireClientEntropy/>
4497 (PSTS-061)         <sp:RequireServerEntropy/>
4498 (PSTS-062)         </wsp:Policy>
4499 (PSTS-063)         </sp:Trust13>
4500 (PSTS-064)         </wsp:All>
4501 (PSTS-065)         </wsp:ExactlyOne>
4502 (PSTS-066)         </wsp:Policy>
4503
4504 (PSTS-067)         <wsp:Policy wsu:Id="InOut-Policy"
4505 (PSTS-068)
4506 (PSTS-069)         xmlns:wsp="http://www.w3.org/ns/ws-policy"
4507 (PSTS-070)         xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
4508 (PSTS-071)
4509 (PSTS-072)         xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4510 (PSTS-072)         wss-wssecurity-utility-1.0.xsd"
4511 (PSTS-073)         >
4512 (PSTS-074)         <wsp:ExactlyOne>
4513 (PSTS-075)         <wsp:All>
4514 (PSTS-076)         <sp:SignedParts>
4515 (PSTS-077)         <sp:Body/>
4516 (PSTS-078)         <sp:Header Name="To"
4517 (PSTS-079)         Namespace="http://www.w3.org/2005/08/addressing"/>
4518 (PSTS-080)         <sp:Header Name="From"
4519 (PSTS-081)         Namespace="http://www.w3.org/2005/08/addressing"/>
4520 (PSTS-082)         <sp:Header Name="FaultTo"
4521 (PSTS-083)         Namespace="http://www.w3.org/2005/08/addressing"/>
4522 (PSTS-084)         <sp:Header Name="ReplyTo"
4523 (PSTS-085)         Namespace="http://www.w3.org/2005/08/addressing"/>
4524 (PSTS-086)         <sp:Header Name="MessageID"

```

```

4525 (PSTS-087)      Namespace="http://www.w3.org/2005/08/addressing"/>
4526 (PSTS-088)      <sp:Header Name="RelatesTo"
4527 (PSTS-089)      Namespace="http://www.w3.org/2005/08/addressing"/>
4528 (PSTS-090)      <sp:Header Name="Action"
4529 (PSTS-091)      Namespace="http://www.w3.org/2005/08/addressing"/>
4530 (PSTS-092)      </sp:SignedParts>
4531 (PSTS-093)      <sp:EncryptedParts>
4532 (PSTS-094)      <sp:Body/>
4533 (PSTS-095)      </sp:EncryptedParts>
4534 (PSTS-096)      </wsp:All>
4535 (PSTS-097)      </wsp:ExactlyOne>
4536 (PSTS-098)      </wsp:Policy>

```

4537 Above is the STS Policy that the Client will encounter when obtaining the IssuedToken required by the  
4538 Service Policy. This policy is quite similar in detail to the Service Policy and therefore only the differences  
4539 that are noteworthy will be discussed in the details below.

4540 Similar to the Service Policy, the STS endpoint Policy contains 4 assertions to be complied with:  
4541 sp:SymmetricBinding (PSTS-010)-(PSTS-036), sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047),  
4542 sp:Wss11 (PSTS-048)-(PSTS-056), and sp:Trust13 (PSTS-057)-(PSTS-063).

4543 Lines (PSTS-010)-(PSTS-036) contain the **sp:SymmetricBinding assertion**, where the main difference  
4544 from the previous policy is that here the sp:ProtectionToken is an sp:X509Token that will be used to  
4545 encrypt the ephemeral client-generated key (K1) that will be used for Client-STS communication. Derived  
4546 keys will not be required in this communication.

4547 Lines (PSTS-037)-(PSTS-047) contain the **sp:EndorsingSupportingTokens assertion**, which in this  
4548 case contains an sp:X509Token assertion (PSTS-039)-(PSTS-045) that requires the Client to include  
4549 (PSTS-040) its X509 certificate, which must be used to sign the message signature. The STS uses this  
4550 mechanism to authenticate the Client.

4551 The **sp:Wss11 assertion** (PSTS-048)-(PSTS-056), **sp:Trust13 assertion** (PSTS-057)-(PSTS-063) and  
4552 the **operation input and output policies** (PSTS-067)-(PSTS-098) are the same as those described for  
4553 the Service policy above (P067)-(P0116).

4554

4555 Below are included sample messages that comply with the above policies. The messages are presented  
4556 in the same order that they would be used in a real scenario and therefore the Client-STS request  
4557 (MSTS-001)-(MSTS-0231) and response (RSTS-001)-(RSTS-0263) are presented first, which are then  
4558 followed by the Client-Service request (M001-M229) and response (R001)-(R153).

4559 Here is an example Client request to the STS:

```

4560 (MSTS-001)      <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope
4561 (MSTS-002)      xmlns:a=http://www.w3.org/2005/08/addressing
4562 (MSTS-003)      xmlns:e=http://www.w3.org/2001/04/xmlenc#
4563 (MSTS-004)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4564 (MSTS-005)      wssecurity-secext-1.0.xsd"
4565 (MSTS-005)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4566 (MSTS-006)      wssecurity-utility-1.0.xsd" >
4567 (MSTS-006)      <s:Header>
4568 (MSTS-007)      <a:Action s:mustUnderstand="1" u:Id="_3">
4569 (MSTS-008)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
4570 (MSTS-009)      </a:Action>
4571 (MSTS-010)      <a:MessageID u:Id="_4">
4572 (MSTS-011)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4573 (MSTS-012)      </a:MessageID>
4574 (MSTS-013)      <a:ReplyTo u:Id="_5">
4575 (MSTS-014)      <a:Address>
4576 (MSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4577 (MSTS-016)      </a:Address>
4578 (MSTS-017)      </a:ReplyTo>
4579 (MSTS-018)      <a:To s:mustUnderstand="1" u:Id="_6">
4580 (MSTS-019)      http://server.example.com/STS/Scenarios5-6
4581 (MSTS-020)      </a:To>
4582 (MSTS-021)      <o:Security s:mustUnderstand="1">

```

4583 (MSTS-022) <u:Timestamp

4584 (MSTS-023) **u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">**

4585 (MSTS-024) <u:Created>2005-10-25T00:47:36.144Z</u:Created>

4586 (MSTS-025) <u:Expires>2005-10-25T00:52:36.144Z</u:Expires>

4587 (MSTS-026) </u:Timestamp>

4588 (MSTS-027) <e:EncryptedKey

4589 (MSTS-028) **Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-9">**

4590 (MSTS-029) <e:EncryptionMethod Algorithm=

4591 (MSTS-030) "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>

4592 (MSTS-031) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

4593 (MSTS-032) <o:SecurityTokenReference>

4594 (MSTS-033) <o:KeyIdentifier Value=

4595 (MSTS-034) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd#ThumbprintSHA1">

4596

4597 (MSTS-035) W+rGyBmLmVEG//scD7Vo8Kq5G7I=

4598 (MSTS-036) </o:KeyIdentifier>

4599 (MSTS-037) </o:SecurityTokenReference>

4600 (MSTS-038) </KeyInfo>

4601 (MSTS-039) <e:CipherData>

4602 (MSTS-040) <e:CipherValue>

4603 (MSTS-041) <!--base64 encoded cipher-->

4604 (MSTS-042) </e:CipherValue>

4605 (MSTS-043) </e:CipherData>

4606 (MSTS-044) <e:ReferenceList>

4607 (MSTS-045) <e:DataReference URI="#\_2"/>

4608 (MSTS-046) </e:ReferenceList>

4609 (MSTS-047) </e:EncryptedKey>

4610 (MSTS-048) <o:BinarySecurityToken

4611 (MSTS-049) **u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-6"**

4612 (MSTS-050) Value=

4613 (MSTS-051) "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"

4614

4615 (MSTS-052) EncodingType=

4616 (MSTS-053) "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">

4617

4618 (MSTS-054)

4619 (MSTS-055) MIIDDDCCafSgAwIBAgIQM6YEf7FVYx/tZyEXgVComTANBgkqhkiG9w0BAQUFADAwMQ4w

4620 (MSTS-056) DAYDVQKDAVPQVNJUzEeMBwGAlUEAwVT0FTSVMgSW50ZXJvcCBUXXN0IENBMB4XDTA1

4621 (MSTS-057) MDMxOTAwMDAwMFoXDTE4MDMxOTIzNTk1OVowQjEOMAwGAlUECgwFT0FTSVMxIDAeBgNV

4622 (MSTS-058) BAsMF09BU01TIEludGVyb3AgVGZzdCBDZXJ0MQ4wDAYDVQQDDAVBbG1jZTCBnzANBgkq

4623 (MSTS-059) hkiG9w0BAQEFAAOBjQAwGykCgYEAoqi99By1VYo0aHrkKCNT4DkIgL/SgahbeKdGhrb

4624 (MSTS-060) u3K2XG7arfD9tqIBIKMfrx4Gp90NJa85AV1yiNsEyyvq+mUnMpNcKnLXL0jkTmMcqDYbb

4625 (MSTS-061) kehJlXPnaWLzve+mW0pJdPxtf3rbD4PS/cBQIvtpjmrDAU8VsZKT8DN5Kyz+EzsCAwEA

4626 (MSTS-062) AaOBkzCBkDAJBgnVHRMEAjAAMDGAlUdHwQsmCowKKImhiRodHRWoi8vaW50ZXJvcC5i

4627 (MSTS-063) YnRlc3QubmV0L2Nybc9jYs5jcmwwDgYDVR0PAQH/BAQDAgSwMB0GAlUdQWBBQK410T

4628 (MSTS-064) UHZ1QV3V2QtllNDm+PoxiDafBgNVHSMEGDAWgBTAnSj8wes1oR3WqqqgHbPnWkKPDzAN

4629 (MSTS-065) BgkqhkiG9w0BAQUFAAOCAQEABTqpOpvW+6yrLXyU1P2xJbEkohXHI5OWwKWleOb9h1kh

4630 (MSTS-066) WntUalfcFOJAgUyH30TtpHldzx1+vK2LPzhoUFKYHE1IyQvokBN2JjFO64BquKCKnZhl

4631 (MSTS-067) dLRPxBghkTdxQgdf5rCK/wh3xVsZCNTfuMnmlAM61OAg8QduDah3WFZpEA0s2nwQaCNQ

4632 (MSTS-068) TNmjJC8tav1CBR6+E5FAMwPXP7pJxn9Fw9OXRYqbRA4v2y7YpbGkG2GI9UvOHw6SGvff4

4633 (MSTS-069) FRStHMMO35YbpikGsLix3vAsXWwi4rWfVOYzQK00FPNi9RMCUdSH06m9uLWckiCxjos0

4634 (MSTS-070) FQODZE9l4ATGy9s9hNVwryOJTW==

4635 (MSTS-071) </o:BinarySecurityToken>

4636 (MSTS-072) <Signature Id="\_0" xmlns="http://www.w3.org/2000/09/xmldsig#">

4637 (MSTS-073) <SignedInfo>

4638 (MSTS-074) <CanonicalizationMethod Algorithm=

4639 (MSTS-075) "http://www.w3.org/2001/10/xml-exc-c14n#">

4640 (MSTS-076) <SignatureMethod Algorithm=

4641 (MSTS-077) "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>

4642 (MSTS-078) <Reference URI="#\_1">

4643 (MSTS-079) <Transforms>

4644 (MSTS-080) <Transform Algorithm=

4645 (MSTS-081) "http://www.w3.org/2001/10/xml-exc-c14n#">

4646 (MSTS-082) </Transforms>

```

4647 (MSTS-083) <DigestMethod Algorithm=
4648 (MSTS-084) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4649 (MSTS-085) <DigestValue>VX1fCPwCzVsSc1hZf0BSbCgW2hM=</DigestValue>
4650 (MSTS-086) </Reference>
4651 (MSTS-087) <Reference URI="#_3">
4652 (MSTS-088) <Transforms>
4653 (MSTS-089) <Transform Algorithm=
4654 (MSTS-090) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4655 (MSTS-091) </Transforms>
4656 (MSTS-092) <DigestMethod Algorithm=
4657 (MSTS-093) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4658 (MSTS-094) <DigestValue>FwiFAUuqNDo9SDkk5A28Mg7Pa8Q=</DigestValue>
4659 (MSTS-095) </Reference>
4660 (MSTS-096) <Reference URI="#_4">
4661 (MSTS-097) <Transforms>
4662 (MSTS-098) <Transform Algorithm=
4663 (MSTS-099) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4664 (MSTS-100) </Transforms>
4665 (MSTS-101) <DigestMethod Algorithm=
4666 (MSTS-102) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4667 (MSTS-103) <DigestValue>oM59PsOTpMrDdOcwXYQzjVU10xw=</DigestValue>
4668 (MSTS-104) </Reference>
4669 (MSTS-105) <Reference URI="#_5">
4670 (MSTS-106) <Transforms>
4671 (MSTS-107) <Transform Algorithm=
4672 (MSTS-108) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4673 (MSTS-109) </Transforms>
4674 (MSTS-110) <DigestMethod Algorithm=
4675 (MSTS-111) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4676 (MSTS-112) <DigestValue>KIK3vklFN1QmMdQkplq2azfzrzg=</DigestValue>
4677 (MSTS-113) </Reference>
4678 (MSTS-114) <Reference URI="#_6">
4679 (MSTS-115) <Transforms>
4680 (MSTS-116) <Transform Algorithm=
4681 (MSTS-117) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4682 (MSTS-118) </Transforms>
4683 (MSTS-119) <DigestMethod Algorithm=
4684 (MSTS-120) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4685 (MSTS-121) <DigestValue>RJEE3hrcyCD6PzFJo6fyut6biVg=</DigestValue>
4686 (MSTS-122) </Reference>
4687 (MSTS-123) <Reference
4688 (MSTS-124) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-10">
4689 (MSTS-125) <Transforms>
4690 (MSTS-126) <Transform Algorithm=
4691 (MSTS-127) "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4692 (MSTS-128) </Transforms>
4693 (MSTS-129) <DigestMethod Algorithm=
4694 (MSTS-130) "http://www.w3.org/2000/09/xmldsig#sha1"/>
4695 (MSTS-131) <DigestValue>zQdN5XpejfqXn0Wko0m51ZYiasE=</DigestValue>
4696 (MSTS-132) </Reference>
4697 (MSTS-133) </SignedInfo>
4698 (MSTS-134) <SignatureValue
4699 (MSTS-135) >iHGJ+xV2VZTjM1Rc7AQJrwLY/aM=</SignatureValue>
4700 (MSTS-136) <KeyInfo>
4701 (MSTS-137) <o:SecurityTokenReference>
4702 (MSTS-138) <o:Reference
4703 (MSTS-139) ValueType=
4704 (MSTS-140) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4705 1.1.xsd#EncryptedKey"
4706 (MSTS-141) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-9"/>
4707 (MSTS-142) </o:SecurityTokenReference>
4708 (MSTS-143) </KeyInfo>
4709 (MSTS-144) </Signature>
4710 (MSTS-145) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

```

```

4711 (MSTS-0146) <SignedInfo>
4712 (MSTS-0147)   <CanonicalizationMethod Algorithm=
4713 (MSTS-0148)     "http://www.w3.org/2001/10/xml-exc-c14n#" />
4714 (MSTS-0149)   <SignatureMethod Algorithm=
4715 (MSTS-0150)     "http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
4716 (MSTS-0151)   <Reference URI="#_0">
4717 (MSTS-0152)     <Transforms>
4718 (MSTS-0153)       <Transform Algorithm=
4719 (MSTS-0154)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
4720 (MSTS-0155)     </Transforms>
4721 (MSTS-0156)     <DigestMethod Algorithm=
4722 (MSTS-0157)       "http://www.w3.org/2000/09/xmldsig#sha1" />
4723 (MSTS-0158)     <DigestValue>UZKtShk8q6iu9WR5uQZp04iAitg=</DigestValue>
4724 (MSTS-0159)     </Reference>
4725 (MSTS-0160)   </SignedInfo>
4726 (MSTS-0161)   <SignatureValue>
4727 (MSTS-0162)   Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWIcmG3xGLZYcc8tbmCtZFuQz9tnW35Lmst6vI
4728 (MSTS-0163)   RefuPA7ewRLYORAOjf92SxMbeVTlrIxQbIQNw0bs4SBSLfAo14=
4729 (MSTS-0164)   </SignatureValue>
4730 (MSTS-0165)   <KeyInfo>
4731 (MSTS-0166)     <o:SecurityTokenReference>
4732 (MSTS-0167)       <o:Reference
4733 (MSTS-0168)         ValueType=
4734 (MSTS-0169)       "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4735 (MSTS-0170)       profile-1.0#X509v3"
4736 (MSTS-0171)         URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-6" />
4737 (MSTS-0172)       </o:SecurityTokenReference>
4738 (MSTS-0173)     </KeyInfo>
4739 (MSTS-0174)   </Signature>
4740 (MSTS-0175) </s:Header>
4741 (MSTS-0176) <s:Body u:Id="_1">
4742 (MSTS-0177)   <e:EncryptedData
4743 (MSTS-0178)     Id="_2"
4744 (MSTS-0179)     Type="http://www.w3.org/2001/04/xmlenc#Content">
4745 (MSTS-0180)     <e:EncryptionMethod Algorithm=
4746 (MSTS-0181)       "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
4747 (MSTS-0182)     <e:CipherData>
4748 (MSTS-0183)       <e:CipherValue>
4749 (MSTS-0184)         <!-- base64 encoded octets with encrypted RST request-->
4750 (MSTS-0185)         <!-- Unencrypted form: -->
4751 (MSTS-0186)         <!--
4752 (MSTS-0187)       <t:RequestSecurityToken>
4753 (MSTS-0188)         <t:RequestType>
4754 (MSTS-0189)           http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
4755 (MSTS-0190)         </t:RequestType>
4756 (MSTS-0191)       <wsp:AppliesTo
4757 (MSTS-0192)         xmlns:wsp="http://www.w3.org/ns/ws-policy">
4758 (MSTS-0193)         <a:EndpointReference
4759 (MSTS-0194)           xmlns:a="http://www.w3.org/2005/08/addressing">
4760 (MSTS-0195)           <a:Address
4761 (MSTS-0196)             >http://server.example.com/Scenarios5</a:Address>
4762 (MSTS-0197)         </a:EndpointReference>
4763 (MSTS-0198)       </wsp:AppliesTo>
4764 (MSTS-0199)     <t:Entropy>
4765 (MSTS-0200)       <t:BinarySecret
4766 (MSTS-0201)         u:Id="uuid-4acf589c-0076-4a83-8b66-5f29341514b7-3"
4767 (MSTS-0202)         Type=
4768 (MSTS-0203)           "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce"
4769 (MSTS-0204)         >Uv38QLxDQM9gLoDZ6OwYDiFk094nmwu3Wmay7EdKmhW=</t:BinarySecret>
4770 (MSTS-0205)       </t:Entropy>
4771 (MSTS-0206)     <t:KeyType>
4772 (MSTS-0207)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey
4773 (MSTS-0208)     </t:KeyType>

```

```

4775 (MSTS-0209) <t:KeySize>256</t:KeySize>
4776 (MSTS-0210) <t:ComputedKeyAlgorithm>
4777 (MSTS-0211) http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
4778 (MSTS-0212) </t:ComputedKeyAlgorithm>
4779 (MSTS-0213) <t:SecondaryParameters>
4780 (MSTS-0214) <t:TokenType
4781 (MSTS-0215) >http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
4782 1.1#SAMLV1.1</t:TokenType>
4783 (MSTS-0216) <t:KeyType
4784 (MSTS-0217) >http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey</t:Ke
4785 yType>
4786 (MSTS-0218) <t:KeySize>256</t:KeySize>
4787 (MSTS-0219) <t:CanonicalizationAlgorithm
4788 (MSTS-0220) >http://www.w3.org/2001/10/xml-exc-c14n#</t:CanonicalizationAlgorith
4789 m>
4790 (MSTS-0221) <t:EncryptionAlgorithm
4791 (MSTS-0222) >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptionAlgorithm>
4792 (MSTS-0222) <t:EncryptWith
4793 (MSTS-0223) >http://www.w3.org/2001/04/xmlenc#aes256-cbc</t:EncryptWith>
4794 (MSTS-0223) <t:SignWith
4795 (MSTS-0224) >http://www.w3.org/2000/09/xmldsig#hmac-sha1</t:SignWith>
4796 (MSTS-0224) </t:SecondaryParameters>
4797 (MSTS-0225) </t:RequestSecurityToken>
4798 (MSTS-0226) -->
4799 (MSTS-0227) </e:CipherValue>
4800 (MSTS-0228) </e:CipherData>
4801 (MSTS-0229) </e:EncryptedData>
4802 (MSTS-0230) </s:Body>
4803 (MSTS-0231) </s:Envelope>

```

4804 The message above is a request by the Client to the STS in compliance with the STS policy (PSTS-001)-  
4805 (PSTS-098). Salient features of this message appropriate to the compliance are described below.

4806 Lines (MSTS-027)-(MSTS-047) contain the **e:EncryptedKey element**, which contains the encrypted  
4807 client-generated ephemeral key (K1) (MSTS-039)-(MSTS-043). K1 is encrypted using the STS certificate  
4808 (X509T2) which is specified by its Thumbprint identifier in the WS-Security o:SecurityTokenReference  
4809 (MSTS-032)-(MSTS-037). The e:ReferenceList in the e:EncryptedKey (MSTS-044)-(MSTS-046) indicates  
4810 that the encryption key K1 is used to directly encrypt the message Body which is referenced by the  
4811 e:DataReference URI="#\_2" (MSTS-045).

4812 Lines (MSTS-048)-(MSTS-071) contain a **WS-Security o:BinarySecurityToken**, which contains the  
4813 Client's public X509 certificate (X509T1) that is required for Client authentication to the STS by the  
4814 sp:EndorsingToken in the STS policy (PSTS-037)-(PSTS-047).

4815 Lines (MSTS-072)-(MSTS-0144) contain the **WS-SP message signature** in an XML Digital Signature  
4816 (dsig) element (namespace = ...xmldsig (MSTS-072)). The elements covered by the dsig Signature are  
4817 identified in the dsig Reference elements:

- 4818 ➤ dsig Reference (MSTS-078) covers the s:Body (Id="\_1" (MSTS-0176))
- 4819 ➤ dsig Reference (MSTS-087) covers the a:Action (Id="\_3" (MSTS-007))
- 4820 ➤ dsig Reference (MSTS-096) covers the a:MessageID (Id="\_4" (MSTS-010))
- 4821 ➤ dsig Reference (MSTS-0105) covers the a:ReplyTo (Id="\_5" (MSTS-013))
- 4822 ➤ dsig Reference (MSTS-0114) covers the a:To (Id="\_6" (MSTS-018))
- 4823 ➤ dsig Reference (MSTS-0123) covers the u:Timestamp (Id="uuid ... 3849-10" (MSTS-023))

4824 all as required by the STS Input and Output Policy sp:SignedParts (PSTS-076)-(PSTS-092), which covers  
4825 the first 5 elements above (note: if an element is not present that is identified in the policy (such as  
4826 FaultTo or RelatesTo) it obviously is not required to be signed, but if present must be signed). The  
4827 u:Timestamp is required to be signed by its presence in the STS endpoint policy (PSTS-033).

4828 Lines (MSTS-0136)-(MSTS-0143) of the **WS-SP message signature contain the dsig KeyInfo**, which  
4829 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing  
4830 validation key (i.e. the key which can be used to verify this dsig:Signature), which in this case is contained

4831 in the e:EncryptedKey element (Id="uuid ... 3849-9" (MSTS-027)) that was described above. Note: at this  
4832 point to verify the Signature, the STS will decrypt the e:EncryptedKey using the private key from the STS  
4833 certificate, X509T2, which will produce the client-generated ephemeral signing and encryption key, K1,  
4834 which, in turn, may be used to validate the message signature. Note also: at this point the STS only  
4835 knows whether the data covered by the message signature is valid or not, but the STS does not yet know  
4836 the identity of the entity that actually sent the data. That is covered next:

4837 Lines (MSTS-0145)-(MSTS-0173) contain the **WS-SP endorsing signature**, also in an XML Digital  
4838 Signature element. The endorsing signature only covers one element, the message signature element,  
4839 which is identified by the dsig Reference element:

4840 ➤ dsig Reference (MSTS-0151) covers the dsig Signature (Id="\_0" (MSTS-072))  
4841 as required by the STS endpoint policy sp:EndorsingSupportingTokens (PSTS-037)-(PSTS-047).

4842 Lines (MSTS-0165)-(MSTS-0174) of the **WS-SP endorsing signature contain the dsig KeyInfo**, which  
4843 contains a WS-Security o:SecurityTokenReference, which contains an o:Reference to the signing  
4844 validation key, which in this case is in the o:BinarySecurityToken element (Id="uuid ... 3849-6"  
4845 (MSTS-048)) that was also described above. Note: now the STS finally has the credentials necessary to  
4846 authenticate the Client. The STS will generally have an identity database of trusted clients and their  
4847 associated X509 certificates. If a client successfully produces a signature that can be validated by the  
4848 associated certificate in the STS database, which would have to match the certificate in the  
4849 o:BinarySecurityToken element, then the client is presumed to be authenticated to the level of security  
4850 provided by this mechanism (strong single factor authentication).

4851 Lines (MSTS-0176)-(MSTS-0230) contain the **SOAP message s:Body element**, which contains its  
4852 payload in an e:EncryptedData element (MSTS-0177)-(MSTS-0229). This e:EncryptedData element was  
4853 identified above by its Id="\_2" in the e:ReferenceList (MSTS-044) of the e:EncryptedKey that was  
4854 processed by the STS above to obtain the client ephemeral key (K1), with which this e:EncryptedData  
4855 can be decrypted. Generally, when the e:EncryptedKey is processed, this e:EncryptedData would have  
4856 been decrypted at that time and made available for processing in the event of successful Client  
4857 authentication as described above. Because the contents of this payload are relevant to the rest of this  
4858 example, the contents of the payload will be briefly described.

4859 Lines (MSTS-0187)-(MSTS-0225) contain the **decrypted contents of the SOAP message s:Body**  
4860 **element**, which contain a WS-Trust t:RequestSecurityToken element. The t:RequestType (MSTS-0189)  
4861 is "...Issue", which means this is a request to issue a security token, which is the main service of the STS.  
4862 The WS-Policy wsp:AppliesTo element (MSTS-0191)-(MSTS-0198) contains the a:Address of the service,  
4863 to which the Client is requesting access, a service which the STS is presumably responsible for  
4864 authenticating and equipping validated clients to enable their access to. In this case the service is  
4865 identified by the URL <http://server.example.com/Scenarios5>, which was part of the WS-SX Interop  
4866 [[WSSX-WSTR-WSSC-INTEROP](#)].

4867 Lines (MSTS-0199)-(MSTS-0205) contain the Client entropy required by the Service policy (P079), which  
4868 is entropy data provided by the Client to aid in production of the ephemeral key, K2, that will be used for  
4869 Client-Service communication. Lines (MSTS-0206)-(MSTS-0212) contain additional parameters used in  
4870 the WS-Trust interface to the STS that will not be described here.

4871 Lines (MSTS-0213)-(MSTS-0224) contain the WS-Trust t:SecondaryParameters, which contains the  
4872 contents of the Service policy's RequestSecurityTokenTemplate (P045)-(P059), which the Service  
4873 requires that the Client pass to the STS as described above. The t:SecondaryParameters contains the  
4874 information the Service has requested about the SAML 1.1 IssuedToken that the STS is being requested  
4875 to create and deliver to the Client in order the enable the Client to access the Service successfully.

4876 Assuming everything above has executed successfully, the STS will then issue the SAML 1.1  
4877 IssuedToken and return it to the Client in a WS-Trust t:RequestSecurityTokenResponse that is described  
4878 next.

4879

4880 Here is an example STS response to the above Client request:

```
4881 (RSTS-001) <s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope  
4882 (RSTS-002) xmlns:a=http://www.w3.org/2005/08/addressing  
4883 (RSTS-003) xmlns:e=http://www.w3.org/2001/04/xmlenc#
```

```

4884 (RSTS-004)      xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
4885 secext-1.1.xsd"
4886 (RSTS-005)      xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4887 wssecurity-secext-1.0.xsd"
4888 (RSTS-006)      xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4889 wssecurity-utility-1.0.xsd" >
4890 (RSTS-007)      <s:Header>
4891 (RSTS-008)      <a:Action s:mustUnderstand="1" u:Id="_4">
4892 (RSTS-009)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal
4893 (RSTS-010)      </a:Action>
4894 (RSTS-011)      <a:RelatesTo u:Id="_5">
4895 (RSTS-012)      urn:uuid:04d386bf-f850-459e-918b-ad80f3d1e088
4896 (RSTS-013)      </a:RelatesTo>
4897 (RSTS-014)      <a:To s:mustUnderstand="1" u:Id=" 6">
4898 (RSTS-015)      http://www.w3.org/2005/08/addressing/anonymous
4899 (RSTS-016)      </a:To>
4900 (RSTS-017)      <o:Security s:mustUnderstand="1">
4901 (RSTS-018)      <u:Timestamp
4902 (RSTS-019)      u:Id="uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4903 (RSTS-020)      <u:Created>2005-10-25T00:47:38.718Z</u:Created>
4904 (RSTS-021)      <u:Expires>2005-10-25T00:52:38.718Z</u:Expires>
4905 (RSTS-022)      </u:Timestamp>
4906 (RSTS-023)      <e:ReferenceList>
4907 (RSTS-024)      <e:DataReference URI="#_3"/>
4908 (RSTS-025)      </e:ReferenceList>
4909 (RSTS-026)      <k:SignatureConfirmation u:Id="_0"
4910 (RSTS-027)      Value="iHGJ+xV2VZTjM1Rc7AQJrwLY/aM="/>
4911 (RSTS-028)      <k:SignatureConfirmation u:Id="_1"
4912 (RSTS-029)      Value=
4913 (RSTS-030)      "Ovxdeg4KQcfQ1T/hEBJz+Z8dQUAfChaWlcmG3xGLZYcc8tbmCtZFuQz9tnW35
4914 (RSTS-031)      Lmst6vRefuPA7ewRLYORAOjf92SxMbeVT1rIxQbIQNw0bs4SBSLFAo14="/>
4915 (RSTS-032)      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
4916 (RSTS-033)      <SignedInfo>
4917 (RSTS-034)      <CanonicalizationMethod Algorithm=
4918 (RSTS-035)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4919 (RSTS-036)      <SignatureMethod Algorithm=
4920 (RSTS-037)      "http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
4921 (RSTS-038)      <Reference URI="#_2">
4922 (RSTS-039)      <Transforms>
4923 (RSTS-040)      <Transform Algorithm=
4924 (RSTS-041)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4925 (RSTS-042)      </Transforms>
4926 (RSTS-043)      <DigestMethod Algorithm=
4927 (RSTS-044)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4928 (RSTS-045)      <DigestValue>kKx5bpLLlyucgXQ6exv/PbjSf1A=</DigestValue>
4929 (RSTS-046)      </Reference>
4930 (RSTS-047)      <Reference URI="#_4">
4931 (RSTS-048)      <Transforms>
4932 (RSTS-049)      <Transform Algorithm=
4933 (RSTS-050)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4934 (RSTS-051)      </Transforms>
4935 (RSTS-052)      <DigestMethod Algorithm=
4936 (RSTS-053)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4937 (RSTS-054)      <DigestValue>LB+VGn4fP2z45jg0Mdzyo8yTAWQ=</DigestValue>
4938 (RSTS-055)      </Reference>
4939 (RSTS-056)      <Reference URI="#_5">
4940 (RSTS-057)      <Transforms>
4941 (RSTS-058)      <Transform Algorithm=
4942 (RSTS-059)      "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4943 (RSTS-060)      </Transforms>
4944 (RSTS-061)      <DigestMethod Algorithm=
4945 (RSTS-062)      "http://www.w3.org/2000/09/xmldsig#sha1"/>
4946 (RSTS-063)      <DigestValue>izHLxm6V4Lc3PSS9Y6VRv3I5RPw=</DigestValue>
4947 (RSTS-064)      </Reference>

```

```

4948 (RSTS-065) <Reference URI="#_6">
4949 (RSTS-066)   <Transforms>
4950 (RSTS-067)     <Transform Algorithm=
4951 (RSTS-068)       "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4952 (RSTS-069)     </Transforms>
4953 (RSTS-070)     <DigestMethod Algorithm=
4954 (RSTS-071)       "http://www.w3.org/2000/09/xmldsig#sha1"/>
4955 (RSTS-072)     <DigestValue>6LS4X08vC/GMGay2vwmD8fL7J2U=</DigestValue>
4956 (RSTS-073)     </Reference>
4957 (RSTS-074)     <Reference
4958 (RSTS-075)       URI="#uuid-0c947d47-f527-410a-a674-753a9d7d97f7-18">
4959 (RSTS-076)     <Transforms>
4960 (RSTS-077)       <Transform Algorithm=
4961 (RSTS-078)         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4962 (RSTS-079)     </Transforms>
4963 (RSTS-080)     <DigestMethod Algorithm=
4964 (RSTS-081)       "http://www.w3.org/2000/09/xmldsig#sha1"/>
4965 (RSTS-082)     <DigestValue>uXGSpCBfbT1fLNBLdGMgy6DGDio=</DigestValue>
4966 (RSTS-083)     </Reference>
4967 (RSTS-084)     <Reference URI="#_0">
4968 (RSTS-085)     <Transforms>
4969 (RSTS-086)       <Transform Algorithm=
4970 (RSTS-087)         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4971 (RSTS-088)     </Transforms>
4972 (RSTS-089)     <DigestMethod Algorithm=
4973 (RSTS-090)       "http://www.w3.org/2000/09/xmldsig#sha1"/>
4974 (RSTS-091)     <DigestValue>z86w+GrzqRZF56ciuz6ogzVXAUA=</DigestValue>
4975 (RSTS-092)     </Reference>
4976 (RSTS-093)     <Reference URI="#_1">
4977 (RSTS-094)     <Transforms>
4978 (RSTS-095)       <Transform Algorithm=
4979 (RSTS-096)         "http://www.w3.org/2001/10/xml-exc-c14n#"/>
4980 (RSTS-097)     </Transforms>
4981 (RSTS-098)     <DigestMethod Algorithm=
4982 (RSTS-099)       "http://www.w3.org/2000/09/xmldsig#sha1"/>
4983 (RSTS-100)     <DigestValue>Z9Tfd20a5aGngsyOPEvQuE0urvQ=</DigestValue>
4984 (RSTS-101)     </Reference>
4985 (RSTS-102)     </SignedInfo>
4986 (RSTS-103)     <SignatureValue>Q7HhboPUaZyXqUKgG7NCYlhMTXI=</SignatureValue>
4987 (RSTS-104)     <KeyInfo>
4988 (RSTS-105)       <o:SecurityTokenReference
4989 (RSTS-106)         k:TokenType="http://docs.oasis-open.org/wss/
4990 (RSTS-107)           oasis-wss-soap-message-security-1.1#EncryptedKey"
4991 (RSTS-108)         xmlns:k="http://docs.oasis-open.org/wss/
4992 (RSTS-109)           oasis-wss-wssecurity-secext-1.1.xsd">
4993 (RSTS-110)       <o:KeyIdentifier ValueType=
4994 (RSTS-111)         "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
4995 (RSTS-112)         1.1.xsd#EncryptedKeySHA1">
4996 (RSTS-113)           CixQW5yEb3mw6XYqD8Ysvrf8cwI=
4997 (RSTS-114)         </o:KeyIdentifier>
4998 (RSTS-115)       </o:SecurityTokenReference>
4999 (RSTS-116)     </KeyInfo>
5000 (RSTS-117)     </Signature>
5001 (RSTS-118)     </o:Security>
5002 (RSTS-119)     </s:Header>
5003 (RSTS-120)     <s:Body u:Id="_2">
5004 (RSTS-121)       <e:EncryptedData Id="_3"
5005 (RSTS-122)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5006 (RSTS-123)       <e:EncryptionMethod Algorithm=
5007 (RSTS-124)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5008 (RSTS-125)     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5009 (RSTS-126)       <o:SecurityTokenReference
5010 (RSTS-127)         k:TokenType="http://docs.oasis-open.org/wss/
5011 (RSTS-128)         oasis-wss-soap-message-security-1.1#EncryptedKey"

```

```

5012 (RSTS-0127)          xmlns:k="http://docs.oasis-open.org/wss/
5013 (RSTS-0128)          oasis-wss-wssecurity-secext-1.1.xsd">
5014 (RSTS-0129)          <o:KeyIdentifier ValueType=
5015 (RSTS-0130)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5016 (RSTS-0131)          1.1.xsd#EncryptedKeySHA1">
5017 (RSTS-0132)          CixQW5yEb3mw6XYqD8Ysvrf8cwI=
5018 (RSTS-0133)          </o:KeyIdentifier>
5019 (RSTS-0134)          </o:SecurityTokenReference>
5020 (RSTS-0135)          </KeyInfo>
5021 (RSTS-0136)          <e:CipherData>
5022 (RSTS-0137)          <e:CipherValue>
5023 (RSTS-0138)          <!--base64 encoded octets of encrypted RSTR-->
5024 (RSTS-0139)          <!--
5025 (RSTS-0140)          <t:RequestSecurityTokenResponseCollection>
5026 (RSTS-0141)          <t:RequestSecurityTokenResponse>
5027 (RSTS-0142)          <t:TokenType>
5028 (RSTS-0143)          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5029 (RSTS-0144)          1.1#SAMLV1.1
5030 (RSTS-0145)          </t:TokenType>
5031 (RSTS-0146)          <t:KeySize>256</t:KeySize>
5032 (RSTS-0147)          <t:RequestedAttachedReference>
5033 (RSTS-0148)          <o:SecurityTokenReference>
5034 (RSTS-0149)          <o:KeyIdentifier ValueType=
5035 (RSTS-0150)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5036 (RSTS-0151)          1.0#SAMLAssertionID">
5037 (RSTS-0152)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5038 (RSTS-0153)          </o:KeyIdentifier>
5039 (RSTS-0154)          </o:SecurityTokenReference>
5040 (RSTS-0155)          </t:RequestedAttachedReference>
5041 (RSTS-0156)          <t:RequestedUnattachedReference>
5042 (RSTS-0157)          <o:SecurityTokenReference>
5043 (RSTS-0158)          <o:KeyIdentifier ValueType=
5044 (RSTS-0159)          "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5045 (RSTS-0160)          1.0#SAMLAssertionID">
5046 (RSTS-0161)          uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5047 (RSTS-0162)          </o:KeyIdentifier>
5048 (RSTS-0163)          </o:SecurityTokenReference>
5049 (RSTS-0164)          </t:RequestedUnattachedReference>
5050 (RSTS-0165)          <t:Lifetime>
5051 (RSTS-0166)          <u:Created>2005-10-24T20:19:26.526Z</u:Created>
5052 (RSTS-0167)          <u:Expires>2005-10-25T06:24:26.526Z</u:Expires>
5053 (RSTS-0168)          </t:Lifetime>
5054 (RSTS-0169)          <t:RequestedSecurityToken>
5055 (RSTS-0170)          <saml:Assertion MajorVersion="1" MinorVersion="1"
5056 (RSTS-0171)          AssertionID="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16"
5057 (RSTS-0172)          Issuer="Test STS" IssueInstant="2005-10-24T20:24:26.526Z"
5058 (RSTS-0173)          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
5059 (RSTS-0174)          <saml:Conditions NotBefore="2005-10-24T20:19:26.526Z"
5060 (RSTS-0175)          NotOnOrAfter="2005-10-25T06:24:26.526Z">
5061 (RSTS-0176)          <saml:AudienceRestrictionCondition>
5062 (RSTS-0177)          <saml:Audience
5063 (RSTS-0178)          >http://server.example.com/Scenarios5</saml:Audience>
5064 (RSTS-0179)          </saml:AudienceRestrictionCondition>
5065 (RSTS-0180)          </saml:Conditions>
5066 (RSTS-0181)          <saml:Advice>
5067 (RSTS-0182)          </saml:Advice>
5068 (RSTS-0183)          <saml:AttributeStatement>
5069 (RSTS-0184)          <saml:Subject>
5070 (RSTS-0185)          <saml:SubjectConfirmation>
5071 (RSTS-0186)          <saml:ConfirmationMethod>
5072 (RSTS-0187)          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
5073 (RSTS-0188)          </saml:ConfirmationMethod>
5074 (RSTS-0189)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5075 (RSTS-0190)          <e:EncryptedKey

```

```

5076 (RSTS-0187)          xmlns:e="http://www.w3.org/2001/04/xmlenc#">
5077 (RSTS-0188)          <e:EncryptionMethod Algorithm=
5078 (RSTS-0189)          "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
5079 (RSTS-0190)          </e:EncryptionMethod>
5080 (RSTS-0191)          <KeyInfo>
5081 (RSTS-0192)          <o:SecurityTokenReference xmlns:o=
5082 (RSTS-0193)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5083 (RSTS-0194)          secext-1.0.xsd">
5084 (RSTS-0194)          <o:KeyIdentifier ValueType=
5085 (RSTS-0195)          "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5086 (RSTS-0196)          1.1.xsd#ThumbprintSHA1">
5087 (RSTS-0196)          NQM0IBvuplAtETQvk+6gn8C13wE=
5088 (RSTS-0197)          </o:KeyIdentifier>
5089 (RSTS-0198)          </o:SecurityTokenReference>
5090 (RSTS-0199)          </KeyInfo>
5091 (RSTS-0200)          <e:CipherData>
5092 (RSTS-0201)          <e:CipherValue>
5093 (RSTS-0202)          EEcYjwNoYcJ+20xTYE5e/fixl5K0gzgrfaYAxkDFv/VXiuKfl084h8PmogTfM+azcgAf
5094 (RSTS-0203)          mArVQvOyKWXRb5vmXYfVHLlhZTbXacy+nowSUNnEjp37VDbI3RJ5k6tBHF+ow0NM/P6G
5095 (RSTS-0204)          PNZ9ZqJi1lGDgWJkFsJzNZXNbbMgwuFu3cA=</e:CipherValue>
5096 (RSTS-0205)          </e:CipherData>
5097 (RSTS-0206)          </e:EncryptedKey>
5098 (RSTS-0207)          </KeyInfo>
5099 (RSTS-0208)          </saml:SubjectConfirmation>
5100 (RSTS-0209)          </saml:Subject>
5101 (RSTS-0210)          </saml:AttributeStatement>
5102 (RSTS-0211)          <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5103 (RSTS-0212)          <SignedInfo>
5104 (RSTS-0213)          <CanonicalizationMethod Algorithm=
5105 (RSTS-0214)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5106 (RSTS-0215)          </CanonicalizationMethod>
5107 (RSTS-0216)          <SignatureMethod Algorithm=
5108 (RSTS-0217)          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
5109 (RSTS-0218)          </SignatureMethod>
5110 (RSTS-0219)          <Reference
5111 (RSTS-0220)          URI="#uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16">
5112 (RSTS-0221)          <Transforms>
5113 (RSTS-0222)          <Transform Algorithm=
5114 (RSTS-0223)          "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
5115 (RSTS-0224)          </Transform>
5116 (RSTS-0225)          <Transform Algorithm=
5117 (RSTS-0226)          "http://www.w3.org/2001/10/xml-exc-c14n#">
5118 (RSTS-0227)          </Transform>
5119 (RSTS-0228)          </Transforms>
5120 (RSTS-0229)          <DigestMethod Algorithm=
5121 (RSTS-0230)          "http://www.w3.org/2000/09/xmldsig#sha1">
5122 (RSTS-0231)          </DigestMethod>
5123 (RSTS-0232)          <DigestValue
5124 (RSTS-0233)          >7nHBrFPsm+LEFAoV4NoQPoEl5Lk=</DigestValue>
5125 (RSTS-0234)          </Reference>
5126 (RSTS-0235)          </SignedInfo>
5127 (RSTS-0236)          <SignatureValue
5128 (RSTS-0237)          >TugV4pTIwCH87bLD4jiMgVGtkbRBt1tRlHXJArL34A/YfA4AnGBLXB4pJdUsUxMUtbQ
5129 (RSTS-0238)          14PoGgEsdLNq8C77peARELGP1/Tqw7T3u5zBYHxCHCiV2FWBBfeOmwJmgoaBf8XZJ4Al
5130 (RSTS-0239)          yqPq61P61jrQjZJafpHuYpAZnZQsvisiJaBPQ=</SignatureValue>
5131 (RSTS-0240)          <KeyInfo>
5132 (RSTS-0241)          <o:SecurityTokenReference xmlns:o=
5133 (RSTS-0242)          "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
5134 (RSTS-0243)          secext-1.0.xsd">
5135 (RSTS-0243)          <o:KeyIdentifier ValueType=
5136 (RSTS-0244)          http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5137 (RSTS-0244)          1.1.xsd#ThumbprintSHA1
5138 (RSTS-0245)          >W+rqYBmLmVEG//scD7Vo8Kq5G7I=</o:KeyIdentifier>
5139 (RSTS-0246)          </o:SecurityTokenReference>

```

```

5140 (RSTS-0247)         </KeyInfo>
5141 (RSTS-0248)         </Signature>
5142 (RSTS-0249)         </saml:Assertion>
5143 (RSTS-0250)         </t:RequestedSecurityToken>
5144 (RSTS-0251)         <t:RequestedProofToken>
5145 (RSTS-0252)         <t:BinarySecret
5146 (RSTS-0253)         u:Id="uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-14"
5147 (RSTS-0254)         >zT8LWAUwUrIVKA/rkCr0kxlEmKAehcB6TGWJuAgucBM=</t:BinarySecret>
5148 (RSTS-0255)         </t:RequestedProofToken>
5149 (RSTS-0256)         </t:RequestSecurityTokenResponse>
5150 (RSTS-0257)         </t:RequestSecurityTokenResponseCollection>
5151 (RSTS-0258)         -->
5152 (RSTS-0259)         </e:CipherValue>
5153 (RSTS-0260)         </e:CipherData>
5154 (RSTS-0261)         </e:EncryptedData>
5155 (RSTS-0262)         </s:Body>
5156 (RSTS-0263)         </s:Envelope>

```

5157 From here on the description will be less detailed except where new concepts that have not been covered  
5158 previously in this example. For instance, tracing the dsig References to their associated elements and  
5159 policy requirements will not be detailed since it follows the same patterns that have just been described in  
5160 the message above.

5161 The message above is a response from the STS to the Client that contains the requested security tokens  
5162 that the Client needs to access the Service.

5163 Lines (RSTS-023)-(RSTS-025) contain a **standalone e:ReferenceList** that has an e:DataReference  
5164 (Id="\_3") that points to the e:EncryptedData element in the SOAP s:Body (RSTS-0119). This  
5165 e:DataReference will be used later in the processing of this message.

5166 Lines (RSTS-026)-(RSTS-031) contain **2 WS-Security 1.1 k:SignatureConfirmation elements** that  
5167 indicate to the Client that the STS has processed the data in the Client request and in particular, has  
5168 processed the information covered by the 2 dsig Signatures in that request, that are identified by their  
5169 respective dsig SignatureValue elements (see lines (MSTS-0134)-(MSTS-0135) and (MSTS-0161)-  
5170 (MSTS-0163) in the Client request to the STS above to compare SignatureValues).

5171 Lines (RSTS-032)-(RSTS-0115) contain the **WS-SP message signature** for this message.

5172 Lines (RSTS-0103)-(RSTS-0114) contain the **WS-SP message signature dsig KeyInfo element**, which  
5173 contains a WS-Security 1.1 o:SecurityTokenReference, which contains an o:KeyIdentifier of ValueType  
5174 "... EncryptedKeySHA1". This is a WS-Security 1.1 construct [[WS\\_SECURITY\\_11](#)] used to reference a  
5175 security token that is not included in the message, which in this case is the Client-generated ephemeral  
5176 key, K1, that the Client used to prepare the request and delivered to the STS in the e:EncryptedKey  
5177 element in that request (MSTS-027) that was described in detail above. The contents of the  
5178 o:KeyIdentifier (RSTS-0111) consist of the SHA1 of K1. This should be sufficient information to let the  
5179 Client know that the STS used its X509 certificate, X509T2, to decrypt K1 from the e:EncryptedKey in the  
5180 Client request, which will enable the Client to trust the contents of this STS response.

5181 Lines (RSTS-0119)-(RSTS-0261) contain the **e:EncryptedData**, which is also provided in decrypted form  
5182 for instructive purposes. As mentioned above, this element is referred to by the standalone  
5183 e:ReferenceList element in the WS-Security header (RSTS-023) and as a result can be not tied to any  
5184 particular encryption key as described in [[WS\\_SECURITY\\_11](#)], and since referenced will be processed by  
5185 WS-Security.

5186 Lines (RSTS-0123)-(RSTS-0134) contain the **e:EncryptedData dsig KeyInfo**, which refers to the same  
5187 ephemeral key, K1, as described above for the message signature for this message.

5188 Lines (RSTS-0140)-(RSTS-0256) contain the **decrypted WS-Trust t:RequestSecurityTokenResponse**  
5189 from the STS. Briefly, lines (RSTS-0145)-(RSTS-0160) contain WS-Security o:SecurityTokenReference  
5190 elements that refer to the SAML 1.1 Assertion that is provided below. These are convenience elements  
5191 for the Client to use in subsequent message preparation where the SAML Assertion is used.

5192 Lines (RSTS-0165)-(RSTS-0250) contain **the actual t:RequestedSecurityToken** that has been the main  
5193 object of discussion up until this point, which is the SAML 1.1 Assertion, saml:Assertion (RSTS-0166)-  
5194 (RSTS-0249), that is the sp:IssuedToken provided by the STS for the Client to use to access the Service.

5195 Lines (RSTS-0170)-(RSTS-0176) of the saml:Assertion contain the **saml:Conditions** that specify that this  
5196 token is intended only for the Service, identified by the URL in the **saml:Audience** element (RSTS-0174).

5197 Lines (RSTS-0181)-(RSTS-0208) contain the **all-important saml:SubjectConfirmation element**, which  
5198 contains the **STS-generated encrypted ephemeral key, K3**, that will be used by the Client and Service  
5199 to communicate securely. There are **2 copies of this key, K3**, in this t:RequestSecurityTokenResponse.  
5200 This copy is for the Service. The Client's copy is described below. In any event, the  
5201 saml:SubjectConfirmation element contains a dsig KeyInfo (RSTS-0185)-(RSTS-0207), which contains an  
5202 e:EncryptedKey element (RSTS-0186)-(RSTS-0206), which, in turn, contains a second dsig KeyInfo,  
5203 which contains a WS-Security o:SecurityTokenReference element that contains an o:KeyIdentifier (RSTS-  
5204 0194)-(RSTS-0197) that identifies the key, X509T3, the Service's public X509 certificate, that can be  
5205 used by the Service to decrypt the ultimate object here, which is the ephemeral key, K3, contained in the  
5206 e:CipherData (RSTS-0200)-(RSTS-0205). The Service's public X509 certificate is identified by its WS-  
5207 Security 1.1 ThumbprintSHA1 (RSTS-0196). Therefore, when the Service receives this saml:Assertion, it  
5208 has the ability to obtain the ephemeral key, K3, contained in the saml:SubjectConfirmation, with which it  
5209 can securely communicate with the Client, based on the assurances provided by the STS.

5210 Lines (RSTS-0211)-(RSTS-0248) contain the **saml:Assertion dsig Signature** that is contained in and  
5211 covers the saml:Assertion via the saml:AssertionID, the value of which appears on lines (RSTS-0220) and  
5212 (RSTS-0167).

5213 Lines (RSTS-0240)-(RSTS-0247) contain the **saml:Assertion dsig Signature KeyInfo element**, which  
5214 contains the ThumbprintSHA1 of the **STS public key, X509T2**, which is the same certificate that the  
5215 Client referenced in the e:EncryptedKey when it made initial contact with the STS above (MSTS-035).  
5216 This completes the initial discussion of the characteristics of the IssuedToken saml:Assertion that will be  
5217 used in the remainder of this example.

5218 Lines (RSTS-0251)-(RSTS-0255) of the t:RequestSecurityTokenResponse contains the  
5219 **t:RequestedProofToken**, which contains the **Client copy of the STS-generated ephemeral key, K3**,  
5220 which will be used in the Client-Service communication to authenticate the client to the Service.

5221 At this point we have completed the setup portion of this example that has enabled secure trusted  
5222 communication between the Client and Service as governed by an STS. The exact strength of the  
5223 security protecting this example would be the subject of an official security analysis that is beyond the  
5224 scope of this document, however, the intent has been to provide sufficient detail that all parties concerned  
5225 with such an analysis would have sufficient context to understand and evaluate such an analysis.

5226  
5227 Here is an example of a Client request to the Service using the tokens from the STS response:  
5228

```

5229 (M001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5230 (M002)   xmlns:a=http://www.w3.org/2005/08/addressing
5231 (M003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5232 (M004)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5233 (M005)   wssecurity-secext-1.0.xsd"
5234 (M006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5235 (M007)   secureconversation/200512"
5236 (M008)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5237 (M009)   wssecurity-utility-1.0.xsd" >
5238 (M010)   <s:Header>
5239 (M011)     <a:Action s:mustUnderstand="1" u:Id="_5">
5240 (M012)       http://example.org/Ping
5241 (M013)     </a:Action>
5242 (M014)     <a:MessageID u:Id="_6">
5243 (M015)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5244 (M016)     </a:MessageID>
5245 (M017)     <a:ReplyTo u:Id="_7">
5246 (M018)       <a:Address>
5247 (M019)         http://www.w3.org/2005/08/addressing/anonymous
5248 (M020)       </a:Address>
5249 (M021)     </a:ReplyTo>
5250 (M022)     <a:To s:mustUnderstand="1" u:Id="_8">
5251 (M023)       http://server.example.com/Scenarios5

```

```

5252 (M021) </a:To>
5253 (M022) <o:Security s:mustUnderstand="1">
5254 (M023) <u:Timestamp
5255 (M024) <u:Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5256 (M025) <u:Created>2005-10-25T00:47:38.222Z</u:Created>
5257 (M026) <u:Expires>2005-10-25T00:52:38.222Z</u:Expires>
5258 (M027) </u:Timestamp>
5259 (M028) <e:EncryptedKey
5260 (M029) <Id="uuid-40f5bac7-f9af-4384-80db-cfab34263849-4">
5261 (M030) <e:EncryptionMethod Algorithm=
5262 (M031) http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5263 (M032) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5264 (M033) <o:SecurityTokenReference>
5265 (M034) <o:KeyIdentifier ValueType=
5266 (M035) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5267 1.1.xsd#ThumbprintSHA1">
5268 (M036) NQM0IBvuplAtETQvk+6gn8C13wE=
5269 (M037) </o:KeyIdentifier>
5270 (M038) </o:SecurityTokenReference>
5271 (M039) </KeyInfo>
5272 (M040) <e:CipherData>
5273 (M041) <e:CipherValue>
5274 (M042) <!-- base64 encoded octets of encrypted key K2 -->
5275 (M043) </e:CipherValue>
5276 (M044) </e:CipherData>
5277 (M045) </e:EncryptedKey>
5278 (M046) <sc:DerivedKeyToken u:Id="_0" >
5279 (M047) <o:SecurityTokenReference>
5280 (M048) <o:Reference
5281 (M049) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5282 (M050) </o:SecurityTokenReference>
5283 (M051) <sc:Offset>0</sc:Offset>
5284 (M052) <sc:Length>24</sc:Length>
5285 (M053) <sc:Nonce>7hI6U16OHavffYgpquHWuQ==</sc:Nonce>
5286 (M054) </sc:DerivedKeyToken>
5287 (M055) <sc:DerivedKeyToken u:Id="_2">
5288 (M056) <o:SecurityTokenReference>
5289 (M057) <o:Reference
5290 (M058) <URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-4"/>
5291 (M059) </o:SecurityTokenReference>
5292 (M060) <sc:Nonce>OEu+WEEUxPFRQK7SCFAnEQ==</sc:Nonce>
5293 (M061) </sc:DerivedKeyToken>
5294 (M062) <e:ReferenceList>
5295 (M063) <e:DataReference URI="#_4"/>
5296 (M064) </e:ReferenceList>
5297 (M065) <!-- encrypted SAML assertion -->
5298 (M066) <e:EncryptedData Id="_3"
5299 (M067) Type="http://www.w3.org/2001/04/xmlenc#Element">
5300 (M068) <e:EncryptionMethod Algorithm=
5301 (M069) "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5302 (M070) <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5303 (M071) <!-- encrypted Key K2 -->
5304 (M072) <e:EncryptedKey>
5305 (M073) <e:EncryptionMethod Algorithm=
5306 (M074) "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgflp"/>
5307 (M075) <KeyInfo>
5308 (M076) <o:SecurityTokenReference>
5309 (M077) <o:KeyIdentifier ValueType=
5310 (M078) "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5311 1.1.xsd#ThumbprintSHA1">
5312 (M079) NQM0IBvuplAtETQvk+6gn8C13wE=
5313 (M080) </o:KeyIdentifier>
5314 (M081) </o:SecurityTokenReference>
5315 (M082) </KeyInfo>

```

```

5316 (M083) <e:CipherData>
5317 (M084) <e:CipherValue>
5318 (M085)
5319 cb7+JW2idPNSarK9quqCe9PQwmW2hoUghuyKRe+I9zOts6HaMcg73LqCWuK/jtdpvNl6
5320 (M086) GT/ZDYfcAJ7NlyMGxSiwi4DULtOShqS60TYBIKgUKiA+zXNl2koVsy7amcUhPMIT6/fo
5321 (M087) hH+6MZDA4t6jomcyhlCiW8d9IAzSWFkfg2k=
5322 (M088) </e:CipherValue>
5323 (M089) </e:CipherData>
5324 (M090) </e:EncryptedKey>
5325 (M091) </KeyInfo>
5326 (M092) <e:CipherData>
5327 (M093) <e:CipherValue>
5328 (M094) <!-- base64 encoded octets from SAML assertion encrypted
5329 (M095) with the encrypted key K2 above -->
5330 (M096) <!-- SAML assertion element is identical as received in
5331 (M097) RSTR , unencrypted form is omitted for brevity -->
5332 (M098) <!--.....-->
5333 (M099) </e:CipherValue>
5334 (M100) </e:CipherData>
5335 (M101) </e:EncryptedData>
5336 (M102)
5337 (M103) <sc:DerivedKeyToken u:Id="_9">
5338 (M104) <o:SecurityTokenReference>
5339 (M105) <o:KeyIdentifier ValueType=
5340 (M106) "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
5341 1.0#SAMLAssertionID">
5342 (M107) uuid-8222b7a2-3874-4884-bdb5-9c2ddd4b86b5-16
5343 (M108) </o:KeyIdentifier>
5344 (M109) </o:SecurityTokenReference>
5345 (M110) <sc:Offset>0</sc:Offset>
5346 (M111) <sc:Length>24</sc:Length>
5347 (M112) <sc:Nonce>pgnS/VDSzJn6SFz+Vy23JA==</sc:Nonce>
5348 (M113) </sc:DerivedKeyToken>
5349 (M114) <Signature Id="_1" xmlns="http://www.w3.org/2000/09/xmldsig#">
5350 (M115) <SignedInfo>
5351 (M116) <CanonicalizationMethod Algorithm=
5352 (M117) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5353 (M118) <SignatureMethod Algorithm=
5354 (M119) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5355 (M120) <Reference URI="#_3">
5356 (M121) <Transforms>
5357 (M122) <Transform Algorithm=
5358 (M123) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5359 (M124) </Transforms>
5360 (M125) <DigestMethod Algorithm=
5361 (M126) "http://www.w3.org/2000/09/xmldsig#sha1" />
5362 (M127) <DigestValue>eQdQVGRkVI1YfKJBw7vOYCOeLQw=</DigestValue>
5363 (M128) </Reference>
5364 (M129) <Reference URI="#_5">
5365 (M130) <Transforms>
5366 (M131) <Transform Algorithm=
5367 (M132) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5368 (M133) </Transforms>
5369 (M134) <DigestMethod Algorithm=
5370 (M135) "http://www.w3.org/2000/09/xmldsig#sha1" />
5371 (M136) <DigestValue>xxyKpp5RZ2TebKca2IGOafIgcxk=</DigestValue>
5372 (M137) </Reference>
5373 (M138) <Reference URI="#_6">
5374 (M139) <Transforms>
5375 (M140) <Transform Algorithm=
5376 (M141) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5377 (M142) </Transforms>
5378 (M143) <DigestMethod Algorithm=
5379 (M144) "http://www.w3.org/2000/09/xmldsig#sha1" />

```

```

5380 (M0145) <DigestValue>WyGDDyYbL/hQZJfE3Yx2aK3RkK8=</DigestValue>
5381 (M0146) </Reference>
5382 (M0147) <Reference URI="#_7">
5383 (M0148) <Transforms>
5384 (M0149) <Transform Algorithm=
5385 (M0150) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5386 (M0151) </Transforms>
5387 (M0152) <DigestMethod Algorithm=
5388 (M0153) "http://www.w3.org/2000/09/xmldsig#sha1" />
5389 (M0154) <DigestValue>AEOH0t2KYR8mivgqUGDrgMtxgEQ=</DigestValue>
5390 (M0155) </Reference>
5391 (M0156) <Reference URI="#_8">
5392 (M0157) <Transforms>
5393 (M0158) <Transform Algorithm=
5394 (M0159) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5395 (M0160) </Transforms>
5396 (M0161) <DigestMethod Algorithm=
5397 (M0162) "http://www.w3.org/2000/09/xmldsig#sha1" />
5398 (M0163) <DigestValue>y8n6Dxd3DbD6TR6d6H/oVWsV4yE=</DigestValue>
5399 (M0164) </Reference>
5400 (M0165) <Reference
5401 (M0166) URI="#uuid-40f5bac7-f9af-4384-80db-cfab34263849-14">
5402 (M0167) <Transforms>
5403 (M0168) <Transform Algorithm=
5404 (M0169) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5405 (M0170) </Transforms>
5406 (M0171) <DigestMethod Algorithm=
5407 (M0172) "http://www.w3.org/2000/09/xmldsig#sha1" />
5408 (M0173) <DigestValue>/Cc+bGkkeQ6j1VXZx8PGgmF6MjI=</DigestValue>
5409 (M0174) </Reference>
5410 (M0175) </SignedInfo>
5411 (M0176) <!--base64 encoded signature value -->
5412 (M0177) <SignatureValue>EyKUHUuffPUPE/ZjaFrMJJ5KLKY=</SignatureValue>
5413 (M0178) <KeyInfo>
5414 (M0179) <o:SecurityTokenReference>
5415 (M0180) <o:Reference URI="#_0" />
5416 (M0181) </o:SecurityTokenReference>
5417 (M0182) </KeyInfo>
5418 (M0183) </Signature>
5419 (M0184) <!-- signature over the primary signature above
5420 (M0185) using the key derived from the proof-key, K3,
5421 (M0186) associated with SAML assertion -->
5422 (M0187) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5423 (M0188) <SignedInfo>
5424 (M0189) <CanonicalizationMethod Algorithm=
5425 (M0190) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5426 (M0191) <SignatureMethod Algorithm=
5427 (M0192) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5428 (M0193) <Reference URI="#_1">
5429 (M0194) <Transforms>
5430 (M0195) <Transform Algorithm=
5431 (M0196) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5432 (M0197) </Transforms>
5433 (M0198) <DigestMethod Algorithm=
5434 (M0199) "http://www.w3.org/2000/09/xmldsig#sha1" />
5435 (M0200) <DigestValue>TMSmLlgeUn8cxyb60Ye5Q2nUuxY=</DigestValue>
5436 (M0201) </Reference>
5437 (M0202) </SignedInfo>
5438 (M0203) <SignatureValue>Fh4NyOpAi+NqVFiHBgHWyvzah9I=</SignatureValue>
5439 (M0204) <KeyInfo>
5440 (M0205) <o:SecurityTokenReference>
5441 (M0206) <o:Reference URI="#_9" />
5442 (M0207) </o:SecurityTokenReference>
5443 (M0208) </KeyInfo>

```

```

5444 (M0209)          </Signature>
5445 (M0210)          </o:Security>
5446 (M0211)          </s:Header>
5447 (M0212)          <s:Body u:Id="_3">
5448 (M0213)          <e:EncryptedData Id="_4"
5449 (M0214)              Type="http://www.w3.org/2001/04/xmlenc#Content">
5450 (M0215)          <e:EncryptionMethod Algorithm=
5451 (M0216)              "http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
5452 (M0217)          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5453 (M0218)          <o:SecurityTokenReference >
5454 (M0219)              <o:Reference URI="#_2"/>
5455 (M0220)          </o:SecurityTokenReference>
5456 (M0221)          </KeyInfo>
5457 (M0222)          <e:CipherData>
5458 (M0223)          <e:CipherValue>
5459 (M0224)              <!-- base64 encoded octets of encrypted body content-->
5460 (M0225)          </e:CipherValue>
5461 (M0226)          </e:CipherData>
5462 (M0227)          </e:EncryptedData>
5463 (M0228)          </s:Body>
5464 (M0229)          </s:Envelope>

```

5465 The message above is a request from the Client to the Service using the tokens provided by the STS  
5466 above.

5467 Lines (M022)-(M0210) contain the **WS-Security o:Security header** for this request.

5468 Lines (M028)-(M045) contain an **e:EncryptedKey** that contains the **ephemeral key, K2**, for the Client-  
5469 Service communication. The service must use its X509T3 (M034)-(M037) to decrypt this Client-generated  
5470 ephemeral key, K2.

5471 Lines (M046)-(M054) contain a **WS-SecureConversation sc:DerivedKeyToken**, that contains the  
5472 information required to **generate the signing key, DKT1(K2)** [**WS\_SECURE\_CONV**], including an  
5473 sc:Nonce (M053), and a WS-Security SecurityTokenReference that contains a direct reference (M049) to  
5474 the e:EncryptedKey, K2 (M042). This derived key, DKT1(K2), is used to sign the message in the  
5475 message signature element (M0114)-(M0183).

5476 Lines (M055)-(M054) provide similar **sc:DerivedKeyToken** constructs the **generate the encrypting key**  
5477 **DKT2(K2)**. This derived key, DKT2(K2), is used to encrypt the message body, resulting in the  
5478 EncryptedData element in the s:Body, on lines (M0213)-(M0227).

5479 Lines (M062)-(M064) provide an **e:ReferenceList to reference the e:EncryptedData in the s:Body**  
5480 (M0213), which contains the Client request for the Service. The s:Body payload data in this Client request  
5481 is not of interest to the security properties of this example and will not be shown in decrypted form.

5482 Lines (M066)-(M0101) contain an **e:EncryptedData element that contains the saml:Assertion**  
5483 described in the STS response above. This e:EncryptedData contains a dsig KeyInfo, which, in turn,  
5484 contains an e:EncryptedKey element (M072)-(M090), which contains the Client-generated ephemeral  
5485 key, K2, which was used by the Client to directly encrypt the saml:Assertion. The encryption key, K2, may  
5486 be decrypted, again using the Service public X509 certificate, again identified by its ThumbprintSHA1,  
5487 (M077)-(M080).

5488 Lines (M0103)-(M0113) contain a **third sc:DerivedKeyToken** that contains the information necessary for  
5489 the Service to **generate the endorsing signing key, DKT3(K2)**.

5490 Lines (M0114)-(M0183) contains the **message signature**. It is **signed using** the key identified in the dsig  
5491 KeyInfo (M0178)-(M0182), which contains a reference to the **derived signing key, DKT1(K2)**, which may  
5492 be constructed by the service using the sc:DerivedKeyToken (M046) described above.

5493 Lines (M0187)-(M0209) contain the **message endorsing signature**. It is **signed using the client proof**  
5494 **key, K3**, that was generated by the STS. Note that the Service should compare this key, K3, with the one  
5495 in the saml:SubjectConfirmation in the decrypted saml:Assertion to verify that the Client is using the same  
5496 proof key, K3, that is contained in the saml:Assertion that authenticates this request.

5497 Lines (M0213)-(M0227) contain the **SOAP s:Body message payload, e:EncryptedData**, which may be  
5498 decrypted using the sc:DerivedKeyToken (M055) to generate the decryption key DKT2(K2).

5499

5500 Here is an example response from the Service to the Client:

5501

```
5502 (R001) <s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
5503 (R002)   xmlns:a=http://www.w3.org/2005/08/addressing
5504 (R003)   xmlns:e=http://www.w3.org/2001/04/xmlenc#
5505 (R004)   xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5506 (R005)   1.1.xsd"
5507 (R005)   xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5508 (R006)   wssecurity-secext-1.0.xsd"
5509 (R006)   xmlns:sc="http://docs.oasis-open.org/ws-sx/ws-
5510 (R007)   secureconversation/200512"
5511 (R007)   xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
5512 (R008)   wssecurity-utility-1.0.xsd">
5513 (R008)   <s:Header>
5514 (R009)     <a:Action s:mustUnderstand="1" u:Id="_6">
5515 (R010)       http://example.org/PingResponse
5516 (R011)     </a:Action>
5517 (R012)     <a:RelatesTo u:Id="_7">
5518 (R013)       urn:uuid:a859eb17-1855-4d4f-8f73-85e4cba3e423
5519 (R014)     </a:RelatesTo>
5520 (R015)     <a:To s:mustUnderstand="1" u:Id="_8">
5521 (R016)       http://www.w3.org/2005/08/addressing/anonymous
5522 (R017)     </a:To>
5523 (R018)     <o:Security s:mustUnderstand="1">
5524 (R019)       <u:Timestamp u:Id="uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5525 (R020)         <u:Created>2005-10-25T00:47:38.921Z</u:Created>
5526 (R021)         <u:Expires>2005-10-25T00:52:38.921Z</u:Expires>
5527 (R022)       </u:Timestamp>
5528 (R023)       <sc:DerivedKeyToken u:Id="_0" >
5529 (R024)         <o:SecurityTokenReference
5530 (R025)           k:TokenType="http://docs.oasis-open.org/wss/
5531 (R026)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5532 (R027)           xmlns:k="http://docs.oasis-open.org/wss/
5533 (R028)             oasis-wss-wssecurity-secext-1.1.xsd">
5534 (R029)             <o:KeyIdentifier ValueType=
5535 (R030)             "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5536 (R031)             1.1.xsd#EncryptedKeySHA1"
5537 (R031)             >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5538 (R032)           </o:SecurityTokenReference>
5539 (R033)           <sc:Offset>0</sc:Offset>
5540 (R034)           <sc:Length>24</sc:Length>
5541 (R035)           <sc:Nonce>KFjylGb73BubLul0ZGgx+w==</sc:Nonce>
5542 (R036)         </sc:DerivedKeyToken>
5543 (R037)       <sc:DerivedKeyToken u:Id="_3">
5544 (R038)         <o:SecurityTokenReference
5545 (R039)           k:TokenType="http://docs.oasis-open.org/wss/
5546 (R040)             oasis-wss-soap-message-security-1.1#EncryptedKey"
5547 (R041)           xmlns:k="http://docs.oasis-open.org/wss/
5548 (R042)             oasis-wss-wssecurity-secext-1.1.xsd">
5549 (R043)           <o:KeyIdentifier ValueType=
5550 (R044)           "http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
5551 (R045)           1.1.xsd#EncryptedKeySHA1"
5552 (R045)           >pDJlrSjLzIqi+AcQLUB4GjUuRLs=</o:KeyIdentifier>
5553 (R046)         </o:SecurityTokenReference>
5554 (R047)         <sc:Nonce>omyh+Eg6XIa8q3V5IkHiXg==</sc:Nonce>
5555 (R048)       </sc:DerivedKeyToken>
5556 (R049)       <e:ReferenceList>
5557 (R050)         <e:DataReference URI="#_5"/>
5558 (R051)       </e:ReferenceList>
5559 (R052)       <k:SignatureConfirmation u:Id="_1"
5560 (R053)         Value="EyKUHUuffPUPE/ZjaFrMJJ5KLKY="/>
5561 (R054)       <k:SignatureConfirmation u:Id="_2"
```

```

5562 (R055) Value="Fh4NyOpAi+NqVFihBgHWyvzah9I="/>
5563 (R056) <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
5564 (R057) <SignedInfo>
5565 (R058) <CanonicalizationMethod Algorithm=
5566 (R059) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5567 (R060) <SignatureMethod Algorithm=
5568 (R061) "http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
5569 (R062) <Reference URI="#_4">
5570 (R063) <Transforms>
5571 (R064) <Transform Algorithm=
5572 (R065) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5573 (R066) </Transforms>
5574 (R067) <DigestMethod Algorithm=
5575 (R068) "http://www.w3.org/2000/09/xmldsig#sha1" />
5576 (R069) <DigestValue>y/oItF5TcTOFan7SavhZTTTv48M=</DigestValue>
5577 (R070) </Reference>
5578 (R071) <Reference URI="#_6">
5579 (R072) <Transforms>
5580 (R073) <Transform Algorithm=
5581 (R074) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5582 (R075) </Transforms>
5583 (R076) <DigestMethod Algorithm=
5584 (R077) "http://www.w3.org/2000/09/xmldsig#sha1" />
5585 (R078) <DigestValue>X4UIaLWnaAWTriw4UJ/SFDgm090=</DigestValue>
5586 (R079) </Reference>
5587 (R080) <Reference URI="#_7">
5588 (R081) <Transforms>
5589 (R082) <Transform Algorithm=
5590 (R083) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5591 (R084) </Transforms>
5592 (R085) <DigestMethod Algorithm=
5593 (R086) "http://www.w3.org/2000/09/xmldsig#sha1" />
5594 (R087) <DigestValue>vqy8/D4CDCaI1nnd4wl1Qjyp+qM=</DigestValue>
5595 (R088) </Reference>
5596 (R089) <Reference URI="#_8">
5597 (R090) <Transforms>
5598 (R091) <Transform Algorithm=
5599 (R092) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5600 (R093) </Transforms>
5601 (R094) <DigestMethod Algorithm=
5602 (R095) "http://www.w3.org/2000/09/xmldsig#sha1" />
5603 (R096) <DigestValue>H1lvLAr5g8pbZ6jfZ+2WNYiNjiM=</DigestValue>
5604 (R097) </Reference>
5605 (R098) <Reference
5606 (R099) URI="#uuid-24adda3a-247a-4fec-b4f7-fb3827496cee-16">
5607 (R100) <Transforms>
5608 (R101) <Transform Algorithm=
5609 (R102) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5610 (R103) </Transforms>
5611 (R104) <DigestMethod Algorithm=
5612 (R105) "http://www.w3.org/2000/09/xmldsig#sha1" />
5613 (R106) <DigestValue>dr0g6hycoc884i+BD8FYCJGbbbE=</DigestValue>
5614 (R107) </Reference>
5615 (R108) <Reference URI="#_1">
5616 (R109) <Transforms>
5617 (R110) <Transform Algorithm=
5618 (R111) "http://www.w3.org/2001/10/xml-exc-c14n#" />
5619 (R112) </Transforms>
5620 (R113) <DigestMethod Algorithm=
5621 (R114) "http://www.w3.org/2000/09/xmldsig#sha1" />
5622 (R115) <DigestValue>Rv3N7VNfAqpn0khr3F/qQZmE/l4=</DigestValue>
5623 (R116) </Reference>
5624 (R117) <Reference URI="#_2">
5625 (R118) <Transforms>

```

```

5626 (R0119)         <Transform Algorithm=
5627 (R0120)         "http://www.w3.org/2001/10/xml-exc-c14n#" />
5628 (R0121)         </Transforms>
5629 (R0122)         <DigestMethod Algorithm=
5630 (R0123)         "http://www.w3.org/2000/09/xmldsig#sha1" />
5631 (R0124)         <DigestValue>X2pxEnYPM8cMLrbhNqPgs8xk+a4=</DigestValue>
5632 (R0125)         </Reference>
5633 (R0126)         </SignedInfo>
5634 (R0127)         <SignatureValue>I2jQuDTWWQiNJy/ziyg8AFYO/z4=</SignatureValue>
5635 (R0128)         <KeyInfo>
5636 (R0129)         <o:SecurityTokenReference>
5637 (R0130)         <o:Reference URI="#_0" />
5638 (R0131)         </o:SecurityTokenReference>
5639 (R0132)         </KeyInfo>
5640 (R0133)         </Signature>
5641 (R0134)         </o:Security>
5642 (R0135)         </s:Header>
5643 (R0136)         <s:Body u:Id="_4">
5644 (R0137)         <e:EncryptedData Id="_5"
5645 (R0138)         Type="http://www.w3.org/2001/04/xmlenc#Content">
5646 (R0139)         <e:EncryptionMethod Algorithm=
5647 (R0140)         "http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
5648 (R0141)         <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5649 (R0142)         <o:SecurityTokenReference>
5650 (R0143)         <o:Reference URI="#_3" />
5651 (R0144)         </o:SecurityTokenReference>
5652 (R0145)         </KeyInfo>
5653 (R0146)         <e:CipherData>
5654 (R0147)         <e:CipherValue>
5655 (R0148)         <!--base64 encoded octets of encrypted body content-->
5656 (R0149)         </e:CipherValue>
5657 (R0150)         </e:CipherData>
5658 (R0151)         </e:EncryptedData>
5659 (R0152)         </s:Body>
5660 (R0153)         </s:Envelope>

```

5661 The message above is a response from the Service to the Client using the tokens based on the  
5662 saml:Assertion IssuedToken from the STS.

5663 Lines (R018)-(R0134) contain the WS-Security o:Security header in the SOAP s:Header.

5664 Lines (R023)-(R036) contain an **sc:DerivedKeyToken** that may be used to **construct the derived**  
5665 **signing key DKT4(K2)**, which uses the Client-generated ephemeral key, K2, that the Service received in  
5666 the Client request (M028)-(M045) above, and is now used in the response to the client, similar to the way  
5667 the STS used K1 to respond to the Client, except that in this case the Service will use derived keys  
5668 DKT4(K2) and DKT5(K2) in addition to K2 in the response. This sc:DerivedKeyToken contains a WS-  
5669 Security o:SecurityTokenReference (R024)-(R032) that uses the WS-Security 1.1 mechanism  
5670 EncryptedKeySHA1 to identify the Client-generated ephemeral key, K2, as the key to use to derive  
5671 DKT4(K2) along with the sc:Nonce provided (R035) (and the label as described in  
5672 [\[WS\\_SECURE\\_CONV\]](#)).

5673 Lines (R037)-(R048) contain a **second sc:DerivedKeyToken** that may be used by the Client to  
5674 **construct the derived encryption key, DKT5(K2)**. The same EncryptedKeySHA1 mechanism is used  
5675 by the Client to construct DKT5(K2) as described for DKT4(K2).

5676 Lines (R049)-(R051) contain an **e:ReferenceList** containing an e:DataReference to the EncryptedData  
5677 element in the s:Body (R0136).

5678 Lines (R052)-(R055) contain 2 WS-Security 1.1 k:SignatureConfirmation elements confirming the dsig  
5679 Signatures that were in the client request above (M0203) on the endorsing signature and (M0177) on the  
5680 message signature of the Client request. The dsig SignatureValues compare respectively to assure the  
5681 Client that the data from the Client request was processed and is what is being referred to in this  
5682 response.

5683 Lines (R056)-(R0134) contain the message signature, which is signed as referenced in the dsig KeyInfo  
5684 (R0128)-(R0132) by DKT4(K2), which may be constructed as described above using the  
5685 sc:DerivedKeyToken element (R023)-(R036).

5686 Lines (R0137)-(R0151) contain the Service response payload to the Client which may be decrypted using  
5687 DKT5(K2) using the sc:DerivedKeyToken element (R037)-(R048).

## 5688 2.4 Secure Conversation Scenarios

### 5689 2.4.1 (WSS 1.0) Secure Conversation bootstrapped by Mutual 5690 Authentication with X.509 Certificates

5691 This scenario corresponds to the situation where both parties have an X.509 certificate (and public/private  
5692 key pair). Because of the volume of messages from each source, the Requestor/Initiator uses WS-  
5693 SecureConversation to establish a new session key and uses the session key for integrity and/or  
5694 confidentiality protection. This improves performance, by using less expensive symmetric key operations  
5695 and improves security by reducing the exposure of the long term secret.

5696 The recipient models this scenario by using the symmetric binding that includes a  
5697 SecureConversationToken assertion to describe the token type accepted by this endpoint. This token  
5698 assertion further contains a bootstrap policy to indicate the security binding that is used by requestors that  
5699 want a security context token issued by this service. The bootstrap policy affects the Request Security  
5700 Token Request (RST) and Request Security Token Request Response (RSTR) messages sent between  
5701 the Initiator and the Recipient to establish the security context. It is modeled by use of an asymmetric  
5702 binding assertion for this scenario because both parties mutually authenticate each other using their  
5703 X.509 certificates.

5704 The message level policies cover a different scope of the web service definition than the security binding  
5705 level policy and so appear as separate policies and are attached at Message Policy Subject. These are  
5706 shown below as input and output policies. Thus, we need a set of coordinated policies one with endpoint  
5707 subject (WSS10SecureConversation\_policy) and two (WSS10SecureConversation\_input\_policy,  
5708 WSS10SecureConversation\_output\_policy) with message subjects to achieve this use case.

5709

```
5710 (P001) <wsp:Policy wsu:Id="WSS10SecureConversation_policy">
5711 (P002)   <wsp:ExactlyOne>
5712 (P003)     <wsp:All>
5713 (P004)       <sp:SymmetricBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
5714 securitypolicy/200702">
5715 (P005)         <wsp:Policy>
5716 (P006)           <sp:ProtectionToken>
5717 (P007)             <wsp:Policy>
5718 (P008)               <sp:SecureConversationToken
5719 sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken
5720 /AlwaysToRecipient">
5721 (P009)                 <wsp:Policy>
5722 (P010)                   <sp:RequireDerivedKeys/>
5723 (P011)                   <sp:BootstrapPolicy>
5724 (P012)                   <wsp:Policy>
5725 (P013)                   <wsp:ExactlyOne>
5726 (P014)                   <wsp:All>
5727 (P015)                     <sp:AsymmetricBinding
5728 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
5729 (P016)                       <wsp:Policy>
5730 (P017)                         <sp:InitiatorToken>
5731 (P018)                           <wsp:Policy>
5732 (P019)                             <sp:X509Token
5733 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
5734 AlwaysToRecipient">
5735 (P020)                               <wsp:Policy>
5736 (P021)                                 <sp:WssX509V3Token10/>
5737 (P022)                                   </wsp:Policy>
```

```

5738 (P023) </sp:X509Token>
5739 (P024) </wsp:Policy>
5740 (P025) </sp:InitiatorToken>
5741 (P026) <sp:RecipientToken>
5742 (P027) <wsp:Policy>
5743 (P028) <sp:X509Token
5744 sp:IncludeToken="http://schemas.xmlsoap.org/ws/200512/securitypolicy/IncludeToken/
5745 AlwaysToInitiator">
5746 (P029) <wsp:Policy>
5747 (P030) <sp:WssX509V3Token10/>
5748 (P031) </wsp:Policy>
5749 (P032) </sp:X509Token>
5750 (P033) </wsp:Policy>
5751 (P034) </sp:RecipientToken>
5752 (P035) <sp:AlgorithmSuite>
5753 (P036) <wsp:Policy>
5754 (P037) <sp:TripleDesRsa15/>
5755 (P038) </wsp:Policy>
5756 (P039) </sp:AlgorithmSuite>
5757 (P040) <sp:Layout>
5758 (P041) <wsp:Policy>
5759 (P042) <sp:Strict/>
5760 (P043) </wsp:Policy>
5761 (P044) </sp:Layout>
5762 (P045) <sp:IncludeTimestamp/>
5763 (P046) <sp:OnlySignEntireHeadersAndBody/>
5764 (P047) </wsp:Policy>
5765 (P048) </sp:AsymmetricBinding>
5766 (P049) <sp:Wss10>
5767 (P050) <wsp:Policy>
5768 (P051) <sp:MustSupportRefKeyIdentifier/>
5769 (P052) </wsp:Policy>
5770 (P053) </sp:Wss10>
5771 (P054) <sp:SignedParts>
5772 (P055) <sp:Body/>
5773 (P056) <sp:Header Name="Action"
5774 Namespace="http://www.w3.org/2005/08/addressing"/>
5775 (P057) </sp:SignedParts>
5776 (P058) <sp:EncryptedParts>
5777 (P059) <sp:Body/>
5778 (P060) </sp:EncryptedParts>
5779 (P061) </wsp:All>
5780 (P062) </wsp:ExactlyOne>
5781 (P063) </wsp:Policy>
5782 (P064) </sp:BootstrapPolicy>
5783 (P065) </wsp:Policy>
5784 (P066) </sp:SecureConversationToken>
5785 (P067) </wsp:Policy>
5786 (P068) </sp:ProtectionToken>
5787 (P069) <sp:AlgorithmSuite>
5788 (P070) <wsp:Policy>
5789 (P071) <sp:Basic256/>
5790 (P072) </wsp:Policy>
5791 (P073) </sp:AlgorithmSuite>
5792 (P074) <sp:Layout>
5793 (P075) <wsp:Policy>
5794 (P076) <sp:Strict/>
5795 (P077) </wsp:Policy>
5796 (P078) </sp:Layout>
5797 (P079) <sp:IncludeTimestamp/>
5798 (P080) <sp:OnlySignEntireHeadersAndBody/>
5799 (P081) </wsp:Policy>
5800 (P082) </sp:SymmetricBinding>
5801 (P083) <sp:Trust13>

```

```

5802 (P084) <wsp:Policy>
5803 (P085) <sp:RequireClientEntropy/>
5804 (P086) <sp:RequireServerEntropy/>
5805 (P087) </wsp:Policy>
5806 (P088) </sp:Trust13>
5807 (P089)
5808 (P090) </wsp:All>
5809 (P091) </wsp:ExactlyOne>
5810 (P092) </wsp:Policy>
5811
5812 (P093) <wsp:Policy wsu:Id="WSS10SecureConversation_input_policy">
5813 (P094) <wsp:ExactlyOne>
5814 (P095) <wsp:All>
5815 (P096) <sp:SignedParts>
5816 (P097) <sp:Header Name="Action"
5817 <sp:Header Name="To"
5818 <sp:Header Name="MessageID"
5819 <sp:Header Name="MessageID"
5820 <sp:Header Name="MessageID"
5821 <sp:Header Name="MessageID"
5822 <sp:Body/>
5823 </sp:SignedParts>
5824 </wsp:All>
5825 </wsp:ExactlyOne>
5826 </wsp:Policy>
5827 (P0105)
5828 (P0106) <wsp:Policy wsu:Id="WSS10SecureConversation_output_policy">
5829 (P0107) <wsp:ExactlyOne>
5830 (P0108) <wsp:All>
5831 (P0109) <sp:SignedParts>
5832 (P0110) <sp:Header Name="Action"
5833 <sp:Header Name="To"
5834 <sp:Header Name="To"
5835 <sp:Header Name="MessageID"
5836 <sp:Header Name="MessageID"
5837 <sp:Header Name="MessageID"
5838 <sp:Header Name="RelatesTo"
5839 <sp:Header Name="RelatesTo"
5840 <sp:Body/>
5841 </sp:SignedParts>
5842 </wsp:All>
5843 </wsp:ExactlyOne>
5844 (P0118) </wsp:Policy>

```

5845 Lines (P004) – (P082) indicate that the service uses the symmetric security binding to protect messages,  
5846 using a Security Context Token (Lines (P008) – (P066)) to sign messages in both directions. The actual  
5847 basis for the signatures should be keys derived from that Security Context Token as stated by the  
5848 RequireDerivedKey assertion in Line (P010). Messages must include a Timestamp element (Line (P079))  
5849 and the signature value must be calculated over the entire body and header elements (Line (P080)).

5850 Lines (P083) – (P088) contain the Trust13 assertion which defines the requirements for the WS-Trust  
5851 related message exchange as part of the SC bootstrap. Line (P085) indicates that the Initiator (Client) has  
5852 to provide entropy to be used as key material for the requested proof token. Line (P086) requires the  
5853 same from the recipient (Server) which results in computed key from both client and server entropy  
5854 values.

5855 Lines (P011) – (P065) contain the bootstrap policy for the initial creation of the security context between  
5856 the communication parties before it is being used. It contains an AsymmetricBinding assertion (Lines  
5857 (P015) – (P048)) which indicates that the initiator's (WS-Security 1.0 X.509 Certificate) token (Lines  
5858 (P017) – (P025)) used by the recipient to verify the signature in the RST must be included in the message  
5859 (P019). The exchange of entropy and key material in the body of the RST and RSTR messages is  
5860 protected by the EncryptedParts assertion of the bootstrap policy in lines (P058) – (P061).

5861 According to the MustSupportKeyRefIdentifier assertions in Line (P051), an X.509 Key Identifier must be  
5862 used to identify the token. Lines (P054) – (P057) require that the body and the WS-Addressing Action  
5863 header is signed on each message (RST, RSTR) within the bootstrap process.

5864 An example of an RST message sent from the initiator to the recipient according to the bootstrap policy  
5865 defined by this policy is as follows:

```
5866 (M001) <?xml version="1.0" encoding="utf-8" ?>
5867 (M002) <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
5868         xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="...">
5869 (M003)   <soap:Header>
5870 (M004)     <wsa:Action wsu:Id="action">
5871 (M005)       http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
5872 (M006)     </wsa:Action>
5873 (M007)     <wsse:Security>
5874 (M008)       <wsu:Timestamp wsu:Id="timestamp">
5875 (M009)         <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5876 (M010)         <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5877 (M011)       </wsu:Timestamp>
5878 (M012)       <wsse:BinarySecurityToken wsu:Id="clientToken"
5879 (M013)         Value="..."#X509v3" EncodingType="...#Base64Binary">
5880 (M014)         MIICZDCCAc2gAwIBAgIRALSOLzt7...
5881 (M015)       </wsse:BinarySecurityToken>
5882 (M016)       <ds:Signature xmlns:ds="...">
5883 (M017)         <ds:SignedInfo>
5884 (M018)           <ds:CanonicalizationMethod
5885 (M019)             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5886 (M020)           <ds:SignatureMethod
5887 (M021)             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
5888 (M022)           <ds:Reference URI="#action">
5889 (M023)             <ds:Transforms>
5890 (M024)               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
5891 (M025)                 exc-c14n#" />
5892 (M026)             </ds:Transforms>
5893 (M027)           <ds:DigestMethod
5894 (M028)             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
5895 (M029)           <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
5896 (M030)         </ds:Reference>
5897 (M031)         <ds:Reference URI="#timestamp">...</ds:Reference>
5898 (M032)         <ds:Reference URI="#body">...</ds:Reference>
5899 (M033)       </ds:SignedInfo>
5900 (M034)       <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
5901 (M035)       <ds:KeyInfo>
5902 (M036)         <wsse:SecurityTokenReference>
5903 (M037)           <wsse:Reference URI="#clientToken"
5904 (M038)             Value="..."#X509v3" />
5905 (M039)         </wsse:SecurityTokenReference>
5906 (M040)       </ds:KeyInfo>
5907 (M041)     </ds:Signature>
5908 (M042)     <xenc:EncryptedKey>
5909 (M043)       <xenc:EncryptionMethod Algorithm="...#rsa-1_5" />
5910 (M044)       <ds:KeyInfo>
5911 (M045)         <wsse:SecurityTokenReference>
5912 (M046)           <wsse:KeyIdentifier
5913 (M047)             Value="..."#X509v3SubjectKeyIdentifier">AtETQ...
5914 (M048)           </wsse:KeyIdentifier>
5915 (M049)         </wsse:SecurityTokenReference>
5916 (M050)       </ds:KeyInfo>
5917 (M051)       <xenc:CipherData>
5918 (M052)         <xenc:CipherValue>
5919 (M053)           <!-- encrypted key -->
5920 (M054)         </xenc:CipherValue>
5921 (M055)       </xenc:CipherData>
5922 (M056)       <xenc:ReferenceList>
5923 (M057)         <xenc:DataReference URI="#request" />
```

```

5924 (M052)      </xenc:ReferenceList>
5925 (M053)      </xenc:EncryptedKey>
5926 (M054)      </wsse:Security>
5927 (M055)      </soap:Header>
5928 (M056)      <soap:Body wsu:Id="body">
5929 (M057)      <xenc:EncryptedData Id="request" Type="...#Content">
5930 (M058)      <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
5931 (M059)      <xenc:CipherData>
5932 (M060)      <xenc:CipherValue>
5933 (M061)      <!-- encrypted RST request -->
5934 (M062)      <!-- ### Begin unencrypted RST request
5935 (M063)      <wst:RequestSecurityToken>
5936 (M064)      <wst:TokenType>
5937 (M065)      http://docs.oasis-open.org/ws-sx/ws-secureconversation/
5938 (M066)      200512/sct
5939 (M066)      </wst:TokenType>
5940 (M067)      <wst:RequestType>
5941 (M068)      http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
5942 (M069)      </wst:RequestType>
5943 (M070)      <wsp:AppliesTo
5944 (M071)      xmlns:wsp="http://www.w3.org/ns/ws-policy">
5945 (M071)      <wsp:EndpointReference>
5946 (M072)      <wsp:Address>
5947 (M073)      http://acme.com/ping/
5948 (M074)      </wsp:Address>
5949 (M075)      </wsp:EndpointReference>
5950 (M076)      </wsp:AppliesTo>
5951 (M077)      <wst:Entropy>
5952 (M078)      <wst:BinarySecret>cr9b0cb1wCEyY9ehaGK33e41h9s=
5953 (M079)      </wst:BinarySecret>
5954 (M080)      </wst:Entropy>
5955 (M081)      </wst:RequestSecurityToken>
5956 (M082)      ### End unencrypted RST request -->
5957 (M083)      </xenc:CipherValue>
5958 (M084)      </xenc:CipherData>
5959 (M085)      </xenc:EncryptedData>
5960 (M086)      </soap:Body>
5961 (M087) </soap:Envelope>

```

- 5962 Line (M005) indicates to the recipient that the SCT Binding of WS-Trust is used.
- 5963 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 5964 Lines (M012) – (M014) contain the initiator’s X.509 token as required by the InitiatorToken assertion’s value (“.../AlwaysToRecipient”).
- 5965
- 5966 Lines (M015) – (M035) hold the message signature.
- 5967 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).
- 5968
- 5969
- 5970
- 5971 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required by the SignedParts assertion.
- 5972
- 5973 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp assertion definition.
- 5974
- 5975 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts assertion.
- 5976
- 5977 Lines (M056) – (M086) hold the Security Token Reference pointing to the initiators X.509 certificate.
- 5978 Lines (M038) – (M058) contain the SOAP Body of the message with the encrypted RST element. Lines (M062) – (M082) show the unencrypted content of the RST request. It specifies the Token Type (Lines (M064) – (M066)) and the Request Type (Lines (M067) – (M069)). According to the Trust13 assertion, it also includes entropy provided by the initiator as indicated by Lines (M077) – (M080).
- 5979
- 5980
- 5981

5982 An example of an RSTR message sent from the recipient to the initiator according to the bootstrap policy  
 5983 defined by this policy is as follows:

```

5984 (M001)    <?xml version="1.0" encoding="UTF-8"?>
5985 (M002)    <soap:Envelope xmlns:soap="..." xmlns:wst="..." xmlns:wsc="..."
5986          xmlns:xenc="...">
5987 (M003)    <soap:Header>
5988 (M004)    <wsa:Action wsu:Id="action">
5989 (M005)    http://docs.oasis-open.org/ws-sx/ws-
5990 trust/200512/RSTRC/IssueFinal
5991 (M006)    </wsa:Action>
5992 (M007)    <wsse:Security>
5993 (M008)    <wsu:Timestamp wsu:Id="timestamp">
5994 (M009)    <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
5995 (M010)    <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
5996 (M011)    </wsu:Timestamp>
5997 (M012)    <wsse:BinarySecurityToken wsu:Id="serviceToken"
5998          ValueType="...#X509v3" EncodingType="...#Base64Binary">
5999 (M013)    MIIASDPiOASDsaöAgIRALSOLzt7...
6000 (M014)    </wsse:BinarySecurityToken>
6001 (M015)    <ds:Signature xmlns:ds="...">
6002 (M016)    <ds:SignedInfo>
6003 (M017)    <ds:CanonicalizationMethod
6004          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6005 (M018)    <ds:SignatureMethod
6006          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6007 (M019)    <ds:Reference URI="#action">
6008 (M020)    <ds:Transforms>
6009 (M021)    <ds:Transform
6010          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6011 (M022)    </ds:Transforms>
6012 (M023)    <ds:DigestMethod
6013          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6014 (M024)    <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6015 (M025)    </ds:Reference>
6016 (M026)    <ds:Reference URI="#timestamp">...</ds:Reference>
6017 (M027)    <ds:Reference URI="#body">...</ds:Reference>
6018 (M028)    </ds:SignedInfo>
6019 (M029)    <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6020 (M030)    <ds:KeyInfo>
6021 (M031)    <wsse:SecurityTokenReference>
6022 (M032)    <wsse:Reference URI="#myToken" ValueType="...#X509v3"/>
6023 (M033)    </wsse:SecurityTokenReference>
6024 (M034)    </ds:KeyInfo>
6025 (M035)    </ds:Signature>
6026 (M036)    <xenc:EncryptedKey>
6027 (M037)    <xenc:EncryptionMethod Algorithm="...#rsa-1_5"/>
6028 (M038)    <ds:KeyInfo>
6029 (M039)    <wsse:SecurityTokenReference>
6030 (M040)    <wsse:KeyIdentifier
6031          ValueType="...#X509v3SubjectKeyIdentifier">AtETQ...
6032 (M042)    </wsse:KeyIdentifier>
6033 (M043)    </wsse:SecurityTokenReference>
6034 (M044)    </ds:KeyInfo>
6035 (M045)    <xenc:CipherData>
6036 (M046)    <xenc:CipherValue>
6037 (M047)    <!-- encrypted key -->
6038 (M048)    </xenc:CipherValue>
6039 (M049)    </xenc:CipherData>
6040 (M050)    <xenc:ReferenceList>
6041 (M051)    <xenc:DataReference URI="#response"/>
6042 (M052)    </xenc:ReferenceList>
6043 (M053)    </xenc:EncryptedKey>
6044 (M054)    </wsse:Security>
6045 (M055)    </soap:Header>

```

```

6046 (M056) <soap:Body wsu:Id="body">
6047 (M057)   <xenc:EncryptedData Id="response" Type="...#Content">
6048 (M058)     <xenc:EncryptionMethod Algorithm="...#aes256-cbc"/>
6049 (M059)     <xenc:CipherData>
6050 (M060)       <xenc:CipherValue>
6051 (M061)         <!-- encrypted RSTR -->
6052 (M062)         <!-- ### Begin unencrypted RSTR
6053 (M063)         <wst:RequestSecurityTokenResponseCollection>
6054 (M064)           <wst:RequestSecurityTokenResponse>
6055 (M065)             <wst:RequestedSecurityToken>
6056 (M066)               <wsc:SecurityContextToken>
6057 (M067)                 <wsc:Identifier>uuid:...</wsc:Identifier>
6058 (M068)                 </wsc:SecurityContextToken>
6059 (M069)             </wst:RequestedSecurityToken>
6060 (M070)           <wst:RequestedProofToken>
6061 (M071)             <xenc:EncryptedKey Id="ek049ea390c90011dbba4e00304852867e">
6062 (M072)               <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
6063 (M073)               <ds:KeyInfo>
6064 (M074)                 <wsse:SecurityTokenReference>
6065 (M075)                   <wsse:KeyIdentifier
6066 (M075)                     ValueType="...#X509SubjectKeyIdentifier"
6067 (M075)                     EncodingType="...#Base64Binary">
6068 (M076)                       Wjw2gDCBye6NJAh0lPCyUldvTN8=
6069 (M076)                     </wsse:KeyIdentifier>
6070 (M077)                   </wsse:SecurityTokenReference>
6071 (M078)                 </ds:KeyInfo>
6072 (M079)               </xenc:EncryptedKey>
6073 (M080)             </wst:RequestedProofToken>
6074 (M081)           </wst:RequestSecurityTokenResponseCollection>
6075 (M082)         </xenc:CipherData>
6076 (M083)       </xenc:EncryptedData>
6077 (M084)     </soap:Body>
6078 (M085)   </soap:Envelope>

```

- 6098 Line (M005) indicates to the initiator that this is the final response to the RST in an RSTRC.
- 6099 Lines (M008) – (M011) hold the Timestamp element as required by the IncludeTimestamp assertion.
- 6100 Lines (M012) – (M014) contain the recipient's X.509 token as required by the RecipientToken assertion's  
6101 value (".../AlwaysToInitiator").
- 6102 Lines (M015) – (M035) hold the message signature.
- 6103 Lines (M036) – (M053) hold the encrypted symmetric key used to encrypt the RSTR response in the body  
6104 of the message according to the bootstrap policy.
- 6105 Lines (M019) – (M025) indicate the WS-Addressing Action header is included in the signature as required  
6106 by the SignedParts assertion.

6107 Line (M026) indicates the Timestamp is included in the signature according to the IncludeTimestamp  
6108 assertion definition.

6109 Line (M027) indicates the SOAP Body is included in the signature as required by the SignedParts  
6110 assertion.

6111 Lines (M031) – (M033) hold the Security Token Reference pointing to the requestor's X.509 certificate.

6112 Lines (M056) – (M102) contain the SOAP Body of the message with the encrypted RSTR collection  
6113 element. Commented out is the unencrypted form of the RSTR collection in Lines (M063) – (M097), which  
6114 includes one RSTR (Lines (M064) – (M096)). Lines (M065) – (M069) contain the new Security Context  
6115 Token. The accompanying RequestedProofToken in Lines (M070) – (M084) include the encrypted secret  
6116 key (Lines (M071) – (M083)) of the security context. The key is encrypted using the initiator's public key  
6117 that was already used to verify its signature in the incoming request.

6118 According to the Trust13 assertion, the response includes entropy provided by the recipient as indicated  
6119 by Lines (M092) – (M095).

6120 An Example of an SCT-secured application message sent from the initiator to the recipient according to  
6121 the message input policy (WSS10SecureConversation\_input\_policy) is as follows:

```

6122 (M001)    <?xml version="1.0" encoding="UTF-8"?>
6123 (M002)    <soap:Envelope xmlns:soap="..." xmlns:wsse="..." xmlns:wsu="..."
6124           xmlns:wst="..." xmlns:xenc="..." xmlns:wsa="..." xmlns:wsm="...">
6125 (M003)    <soap:Header>
6126 (M004)    <wsa:To wsu:Id="to">http://acme.com/ping/</wsa:To>
6127 (M005)    <wsa:MessageID wsu:Id="msgid">
6128 (M006)    <http://acme.com/guid/12318731edh-CA47-1067-B31D-10662DA
6129 (M007)    </wsa:MessageID>
6130 (M008)    <wsa:Action wsu:Id="action">urn:Ping</wsa:Action>
6131 (M009)    <wsse:Security>
6132 (M010)    <wsu:Timestamp wsu:Id="timestamp">
6133 (M011)    <wsu:Created>2007-06-17T00:00:00Z</wsu:Created>
6134 (M012)    <wsu:Expires>2007-06-17T23:59:59Z</wsu:Expires>
6135 (M013)    </wsu:Timestamp>
6136 (M014)    <wsc:SecurityContextToken wsu:Id="SCT">
6137 (M015)    <wsc:Identifier>uuid:91f50600-60cc-11da-8e67-000000000000
6138 (M016)    </wsc:Identifier>
6139 (M017)    </wsc:SecurityContextToken>
6140 (M018)    <wsc:DerivedKeyToken wsu:Id="DKT">
6141 (M019)    <wsse:SecurityTokenReference>
6142 (M020)    <wsse:Reference URI="#SCT" ValueType="http://docs.oasis-
6143           open.org/ws-sx/ws-secureconversation/200512/sct"/>
6144 (M021)    </wsse:SecurityTokenReference>
6145 (M022)    <wsc:Offset>0</wsc:Offset>
6146 (M023)    <wsc:Length>24</wsc:Length>
6147 (M024)    <wsc:Label>
6148           WSSecure ConversationWSSecure Conversation
6149 (M025)    </wsc:Label>
6150 (M026)    <wsc:Nonce>yln04kFBJesy2U2SQL6ezI3SCak=</wsc:Nonce>
6151 (M027)    </wsc:DerivedKeyToken>
6152 (M028)    <ds:Signature xmlns:ds="...">
6153 (M029)    <ds:SignedInfo>
6154 (M030)    <ds:CanonicalizationMethod
6155           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
6156 (M031)    <ds:SignatureMethod
6157           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
6158 (M032)    <ds:Reference URI="#msgid">
6159 (M033)    <ds:Transforms>
6160 (M034)    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-
6161           exc-c14n#"/>
6162 (M035)    </ds:Transforms>
6163 (M036)    <ds:DigestMethod
6164           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
6165 (M037)    <ds:DigestValue>oZKZXftCbY43Wo4w...</ds:DigestValue>
6166 (M038)    </ds:Reference>

```

```

6167 (M039) <ds:Reference URI="#to">...</ds:Reference>
6168 (M040) <ds:Reference URI="#action">...</ds:Reference>
6169 (M041) <ds:Reference URI="#timestamp">...</ds:Reference>
6170 (M042) <ds:Reference URI="#body">...</ds:Reference>
6171 (M043) </ds:SignedInfo>
6172 (M044) <ds:SignatureValue>Po9mb0Gw6hWn...</ds:SignatureValue>
6173 (M045) <ds:KeyInfo>
6174 (M046) <wsse:SecurityTokenReference>
6175 (M047) <wsse:Reference URI="#DKT" ValueType="http://docs.oasis-
6176 open.org/ws-sx/ws-secureconversation/200512/dk"/>
6177 (M048) </wsse:SecurityTokenReference>
6178 (M049) </ds:KeyInfo>
6179 (M050) </ds:Signature>
6180 (M051) </wsse:Security>
6181 (M052) </soap:Header>
6182 (M053) <soap:Body wsu:Id="body">
6183 (M054) <pns:Ping xmlns:pns="http://tempuri.org/">
6184 (M055) <pns:Text>abc</pns:Text>
6185 (M056) </pns:Ping>
6186 (M057) </soap:Body>
6187 (M058) </soap:Envelope>

```

6188

6189 Lines (M004) – (M008) contain the WS-Addressing headers according to the UsingAddressing assertion.

6190 Lines (M010) – (M013) hold the Timestamp as specified by the IncludeTimestamp assertion within the  
6191 SymmetricBinding assertion.

6192 Lines (M014) – (M017) contain the Security Context Token which has been issued by the recipient in the  
6193 previous message. Based on this token, the initiator included a Derived Key Token (Lines (M018) –  
6194 (M027)) as indicated by the RequireDerivedKey assertion in the policy.

6195 Lines (M028) – (M050) hold the message signature that uses the Derived Key Token (Lines (M046) –  
6196 (M048)) to sign the WS-Addressing headers (Lines (M032) – (M040)), the timestamp (Line (M041)) and  
6197 the body (Line (M042)) of the message, according to the SignedParts assertion of the message input  
6198 policy (WSS10SecureConversation\_input\_policy).

---

### 6199 3 Conformance

6200 This document contains non-normative examples of usage of WS-SecurityPolicy and other related  
6201 specifications.

6202 Therefore **there are no conformance statements that apply to this document.**

6203 Refer to the referenced specifications [[References](#)], which will individually contain conformance  
6204 requirements for WS-SecurityPolicy and related specifications.

6205

6206

---

## A. Acknowledgements

6207 The following individuals have participated in the creation of this specification and are gratefully  
6208 acknowledged:

6209 **Original Contributors:**

6210 Ashok Malhotra, Oracle  
6211 Prateek Mishra, Oracle  
6212 Ramana Turlapati, Oracle

6213 **Participants:**

6214 Don Adams, Tibco  
6215 Moushmi Banerjee, Oracle  
6216 Symon Chang, Oracle  
6217 Henry Chung, IBM  
6218 Paul Cotton, Microsoft  
6219 Marc Goodner, Microsoft  
6220 Martin Gudgin, Microsoft  
6221 Tony Gullotta, SOA  
6222 Jiandong Guo, Sun  
6223 Frederick Hirsch, Nokia  
6224 Chris Kaler, Microsoft  
6225 Rich Levinson, Oracle  
6226 Chunlong Liang, IBM  
6227 Hal Lockhart, Oracle  
6228 Mike Lyons, Layer 7  
6229 Ashok Malhotra, Oracle  
6230 Carlo Milono, Tibco  
6231 Prateek Mishra, Oracle  
6232 Anthony Nadalin, Microsoft  
6233 Martin Raeppe, SAP AG  
6234 Bruce Rich, IBM  
6235 Ramana Turlapati, Oracle  
6236 Greg Whitehead, Hewlett-Packard  
6237 Ching-Yun (C.Y.) Chao, IBM  
6238

6239 **TC Members during the development of this specification:**

6240 Don Adams, Tibco Software Inc.  
6241 Jan Alexander, Microsoft Corporation  
6242 Steve Anderson, BMC Software  
6243 Donal Arundel, IONA Technologies  
6244 Howard Bae, Oracle Corporation  
6245 Abbie Barbir, Nortel Networks Limited  
6246 Charlton Barreto, Adobe Systems  
6247 Michael Botha, Software AG, Inc.  
6248 Toufic Boubez, Layer 7 Technologies Inc.  
6249 Norman Brickman, Mitre Corporation  
6250 Melissa Brumfield, Booz Allen Hamilton  
6251 Geoff Bullen, Microsoft Corporation  
6252 Lloyd Burch, Novell  
6253 Scott Cantor, Internet2  
6254 Greg Carpenter, Microsoft Corporation  
6255 Steve Carter, Novell  
6256 Symon Chang, Oracle Corporation  
6257 Martin Chapman, Oracle Corporation  
Ching-Yun (C.Y.) Chao, IBM

6258 Kate Cherry, Lockheed Martin  
6259 Henry (Hyenvui) Chung, IBM  
6260 Luc Clement, Systinet Corp.  
6261 Paul Cotton, Microsoft Corporation  
6262 Glen Daniels, Sonic Software Corp.  
6263 Peter Davis, Neustar, Inc.  
6264 Duane DeCouteau, Veterans Health Administration  
6265 Martijn de Boer, SAP AG  
6266 Werner Dittmann, Siemens AG  
6267 Abdeslem DJAOUI, CCLRC-Rutherford Appleton Laboratory  
6268 Fred Dushin, IONA Technologies  
6269 Petr Dvorak, Systinet Corp.  
6270 Colleen Evans, Microsoft Corporation  
6271 Ruchith Fernando, WSO2  
6272 Mark Fussell, Microsoft Corporation  
6273 Vijay Gajjala, Microsoft Corporation  
6274 Marc Goodner, Microsoft Corporation  
6275 Hans Granqvist, VeriSign  
6276 Martin Gudgin, Microsoft Corporation  
6277 Tony Gullotta, SOA Software Inc.  
6278 Jiandong Guo, Sun Microsystems  
6279 Phillip Hallam-Baker, VeriSign  
6280 Patrick Harding, Ping Identity Corporation  
6281 Heather Hinton, IBM  
6282 Frederick Hirsch, Nokia Corporation  
6283 Jeff Hodges, Neustar, Inc.  
6284 Will Hopkins, Oracle Corporation  
6285 Alex Hristov, Otecia Incorporated  
6286 John Hughes, PA Consulting  
6287 Diane Jordan, IBM  
6288 Venugopal K, Sun Microsystems  
6289 Chris Kaler, Microsoft Corporation  
6290 Dana Kaufman, Forum Systems, Inc.  
6291 Paul Knight, Nortel Networks Limited  
6292 Ramanathan Krishnamurthy, IONA Technologies  
6293 Christopher Kurt, Microsoft Corporation  
6294 Kelvin Lawrence, IBM  
6295 Hubert Le Van Gong, Sun Microsystems  
6296 Jong Lee, Oracle Corporation  
6297 Rich Levinson, Oracle Corporation  
6298 Tommy Lindberg, Dajeil Ltd.  
6299 Mark Little, JBoss Inc.  
6300 Hal Lockhart, Oracle Corporation Mike Lyons, Layer 7 Technologies Inc.  
6301 Eve Maler, Sun Microsystems  
6302 Ashok Malhotra, Oracle Corporation  
6303 Anand Mani, CrimsonLogic Pte Ltd  
6304 Jonathan Marsh, Microsoft Corporation  
6305 Robin Martherus, Oracle Corporation  
6306 Miko Matsumura, Infravio, Inc.  
6307 Gary McAfee, IBM  
6308 Michael McIntosh, IBM  
6309 John Merrells, Sxip Networks SRL  
6310 Jeff Mischkinsky, Oracle Corporation  
6311 Prateek Mishra, Oracle Corporation  
6312 Bob Morgan, Internet2  
6313 Vamsi Motukuru, Oracle Corporation  
6314 Raajmohan Na, EDS

6315 Anthony Nadalin, IBM  
6316 Andrew Nash, Reactivity, Inc.  
6317 Eric Newcomer, IONA Technologies  
6318 Duane Nickull, Adobe Systems  
6319 Toshihiro Nishimura, Fujitsu Limited  
6320 Rob Philpott, RSA Security  
6321 Denis Pilipchuk, Oracle Corporation.  
6322 Darren Platt, Ping Identity Corporation  
6323 Martin Raepple, SAP AG  
6324 Nick Ragouzis, Enosis Group LLC  
6325 Prakash Reddy, CA  
6326 Alain Regnier, Ricoh Company, Ltd.  
6327 Irving Reid, Hewlett-Packard  
6328 Bruce Rich, IBM  
6329 Tom Rutt, Fujitsu Limited  
6330 Maneesh Sahu, Actional Corporation  
6331 Frank Siebenlist, Argonne National Laboratory  
6332 Joe Smith, Apani Networks  
6333 Davanum Srinivas, WSO2  
6334 David Staggs, Veterans Health Administration  
6335 Yakov Sverdlov, CA  
6336 Gene Thurston, AmberPoint  
6337 Victor Valle, IBM  
6338 Asir Vedamuthu, Microsoft Corporation  
6339 Greg Whitehead, Hewlett-Packard  
6340 Ron Williams, IBM  
6341 Corinna Witt, Oracle Corporation  
6342 Kyle Young, Microsoft Corporation  
6343

6344

## B. Revision History

6345

[optional; should not be included in OASIS Standards]

6346

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	<p>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.</p> <p>Completed examples: 2.1.1.2, 2.1.2.1, 2.1.3, 2.1.4, 2.2.1, and 2.5.1.</p> <p>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</p>

6347