Web Services Security:
SAML Interop 1 Scenarios
Working Draft 04, Jan 29, 2004

Document identifier:
wss-saml-interop1-draft-04.doc

Location:
http://www.oasis-open.org/committees/wss/

Editor:
Rich Levinson, Netegrity <rlevinson@netegrity.com>
Hal Lockhart, BEA Systems <hlockhart@bea.com>

Contributors:
Prateek Mishra, Netegrity <pmishra@netegrity.com>
Ron Monzillo, Sun Microsystems <ronald.monzillo@sun.com>

Abstract:
This document documents the four scenarios to be used for the WSS-SAML Interoperability Event.

Status:
Committee members should send comments on this specification to the wss@lists.oasis-open.org list. Others should subscribe to and send comments to the wss-comment@lists.oasis-open.org list. To subscribe, send an email message to wss-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of Contents

Introduction

This document describes message exchanges to be tested during the SAML interoperability event of the WSS TC. All use the Request/Response Message Exchange Pattern (MEP) with no intermediaries. All invoke the same simple application.

These scenarios are intended to test the interoperability of different implementations performing common operations and to test the soundness of the various specifications and clarity and mutual understanding of their meaning and proper application.

THESE SCENARIOS ARE NOT INTENDED TO REPRESENT REASONABLE OR USEFUL PRACTICAL APPLICATIONS OF THE SPECIFICATIONS. THEY HAVE BEEN DESIGNED PURELY FOR THE PURPOSES INDICATED ABOVE AND DO NOT NECESSARILY REPRESENT EFFICIENT OR SECURE MEANS OF PERFORMING THE INDICATED FUNCTIONS. IN PARTICULAR, THESE SCENARIOS ARE KNOWN TO VIOLATE SECURITY BEST PRACTICES IN SOME RESPECTS AND IN GENERAL HAVE NOT BEEN EXTENSIVELY VETTED FOR ATTACKS.

1.1 Terminology

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in **[RFC2119]**.

2 Test Application

All scenarios use the same simple application.

The Requester sends a Ping element with a value of a string. The value should be the name of the organization that has developed the software and the number of the scenario, e.g. “Acme Corp. – Scenario #1”.

The Responder returns a PingResponse element with a value of the same string.

3 Scenario #1 – Sender-Vouches: Unsigned

The request contains a minimal sender-vouches SAML assertion with no optional elements included. There are no signatures or certificates required. The response does not contain a security header.

3.1 Agreements

This section describes the agreements that must be made, directly or indirectly, between the parties who wish to interoperate.

3.1.1 USERNAME-LIST

This is a list of usernames associated with the test users that participate in the test scenario.

3.1.2 ISSUERNAME-LIST

This is a list of trusted issuers of SAML assertions.

3.2 Parameters

This section describes parameters that are required to correctly create or process messages, but not a matter of mutual agreement.

No parameters required.

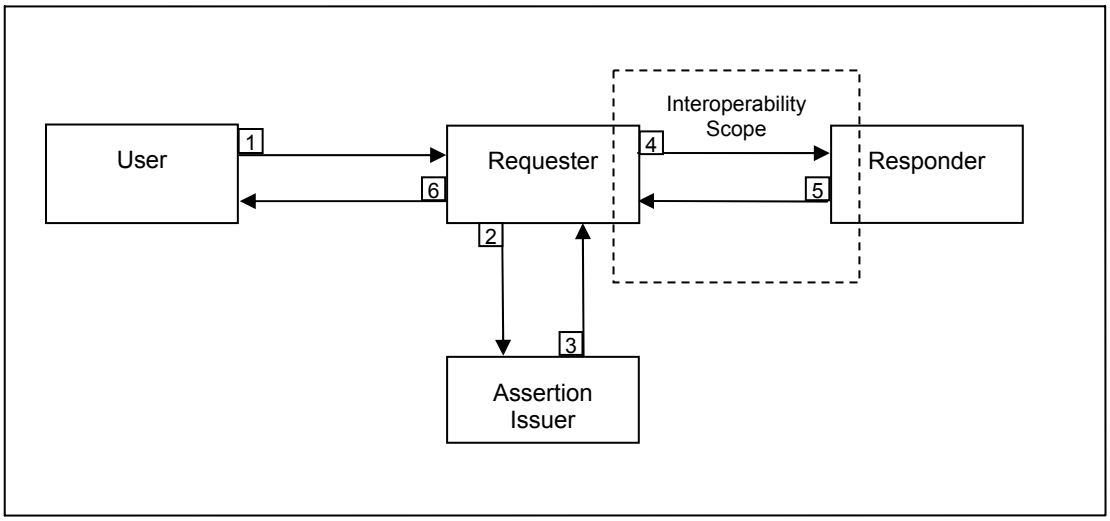
3.3 General Message Flow

This section provides a general overview of the flow of messages.

This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used. As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string may be used. The recipient SHOULD ignore the value. The request contains a plain text SAML assertion which contains one valid SubjectStatement of any valid type. The Responder checks the issuer name in the SAML assertion and determines if it is valid against the local directory of trusted issuers. If no errors are detected it returns the response without any security mechanisms.

3.3.1 Message exchange overview

This section contains a high level diagram of the scenario including the actors and the basis of trust. Interoperability for all scenarios is between the Requester and Responder. For each scenario, a hypothetical set of actions that take place prior to the Requester sending the request will be described in order to give some context for the assembly of the request and to show where the basis of trust lies for the Responder. However, the interoperability aspect of each scenario consists solely of the Request that the Requester sends to the Responder and the Response that the Responder returns to the Requester.



In the **Sender Vouches: Unsigned** scenario there is **no technical basis for trust**, because the messages are sent in the clear with no content or channel protection. This scenario is intended only to demonstrate message structure interoperability and is not intended for production use. In typical scenarios the Requester and Assertion Issuer may be the same party however this is not a requirement and does not impact the interoperability characteristics of this scenario.

In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a potential before and after context for the operability test which only includes steps 4 and 5.

1. User sends a SOAP request to the Requester
2. Requester requests a SAML assertion for this User from the Assertion Issuer.
3. Assertion Issuer returns a Sender-Vouches SAML assertion for the User to the Requester.
4. Requester inserts the Assertion to a wsse:Security header in the SOAP message as described in the “First Message” section below and sends the Request to the Responder.
5. Responder processes Request as described in “Message Processing” section below and returns Response to Requester as described in “Second Message” section below.
6. Requester returns Response to User.

3.4 First Message - Request

3.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand=”1” attribute. Items marked mandatory MUST be generated and processed. Items marked as optional MAY be generated and MUST be processed if generated. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Issuer	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
Body	Mandatory

3.4.2 Message Creation

3.4.2.1 Timestamp

The Created element within the Timestamp SHOULD contain the current local time at the sender expressed in the UTC time zone.

3.4.2.2 Security

The Security element MUST contain the mustUnderstand="1" atttibute.

3.4.2.3 Assertion

The Assertion element MUST contain an Issuer attribute, whose value MUST match an Issuer value in the ISSUERNAME-LIST.

3.4.2.4 SubjectStatement

The Assertion element MUST contain one SubjectStatement, which may be any of the standard SAML SubjectStatement extensions.

3.4.2.4.1 ConfirmationMethod

The Subject element MUST also contain a SubjectConfirmation element which MUST contain a ConfirmationMethod with a value of "urn:oasis:names:tc:SAML:cm:sender-vouches".

3.4.2.5 Body

The Body is not signed or encrypted in any way.

3.4.3 Message Processing

This section describes the processing performed by the Responder. If an error is detected, the Responder MUST cease processing the message. If an error is detected, a SOAP Fault MAY be returned and, if so, it MUST have a value of InvalidSecurityToken.

3.4.3.1 Security

3.4.3.2 Timestamp

The Timestamp element MUST be ignored.

3.4.3.3 Assertion

The Assertion element MUST be validated. The value of the Issuer attribute MUST match an entry in the ISSUERNAME-LIST.

3.4.3.4 SubjectStatement

The Responder MUST check that at least one of the standard SAML SubjectStatement elements is included in the Assertion. Any elements or attributes in the statement other than those explicitly described here SHOULD be ignored.

3.4.3.4.1 ConfirmationMethod

The value of the ConfirmationMethod element MUST be “urn:oasis:names:tc:SAML:cm:sender-vouches”.

3.4.3.5 Body

The Body is passed to the application without modification.

3.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
  xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp>
        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
      </wsu:Timestamp>
```

```
      <saml:Assertion
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
          MajorVersion="1" MinorVersion="0"
          AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
          Issuer="www.example.com"
          IssueInstant="2002-06-19T16:58:33.173Z">
        <saml:AuthenticationStatement
         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
         AuthenticationInstant="2002-06-19T16:57:30.000Z">
          <saml:Subject>
            <saml:NameIdentifier
                NameQualifier="www.example.com"
                Format="">
              uid=joe,ou=people,ou=saml-demo,o=example.com
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
              </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
          </saml:Subject>
        </saml:AuthenticationStatement>
      </saml:Assertion>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </Ping>
 </soap:Body>
</soap:Envelope>
```

3.5 Second Message - Response

3.5.1 Message Elements and Attributes

Items not listed in the following table MUST NOT be created or processed. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Body	Mandatory

3.5.2 Message Creation

The response message MUST NOT contain a <wsse:Security> header. Any other header elements MUST NOT be labeled with a mustUnderstand=”1” attribute.

3.5.3 Message Processing

The Body is passed to the application without modification.

3.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PingResponse xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </PingResponse>
 </soap:Body>
</soap:Envelope>
```

3.6 Other processing

This section describes processing that occurs outside of generating or processing a message.

3.6.1 Requester

No additional processing is required.

3.6.2 Responder

No additional processing is required.

3.7 Expected Security Properties

Use of the service is restricted to Requesters that can obtain and include a SAML assertion with an Issuer name that is on the Responder's ISSUERNAMELIST.

4 Scenario #2 – Sender-Vouches: Unsigned: SSL

The request contains a sender-vouches SAML assertion. There are no signatures required. This scenario MUST be tested over SSL and certificates are required to support SSL at the transport layer. The response does not contain a security header.

4.1 Agreements

This section describes the agreements that must be made, directly or indirectly, between the parties who wish to interoperate.

4.1.1 ISSUERNAME-LIST

This is a list of trusted issuers of SAML assertions.

4.1.2 Signature Trust Root

This refers generally to agreeing on at least one trusted key and any other certificates and sources of revocation information sufficient to validate certificates sent for the purpose of verifying the identity of the Requester.

4.1.3 REQUESTER-CERT-VALUE

This is an opaque identifier indicating the X.509 certificate to be used by the Requester. The Responder MUST have the necessary trusted certificates in the Signature Trust Root to validate the Requester certificate.

4.2 Parameters

This section describes parameters that are required to correctly create or process messages, but not a matter of mutual agreement.

No parameters required.

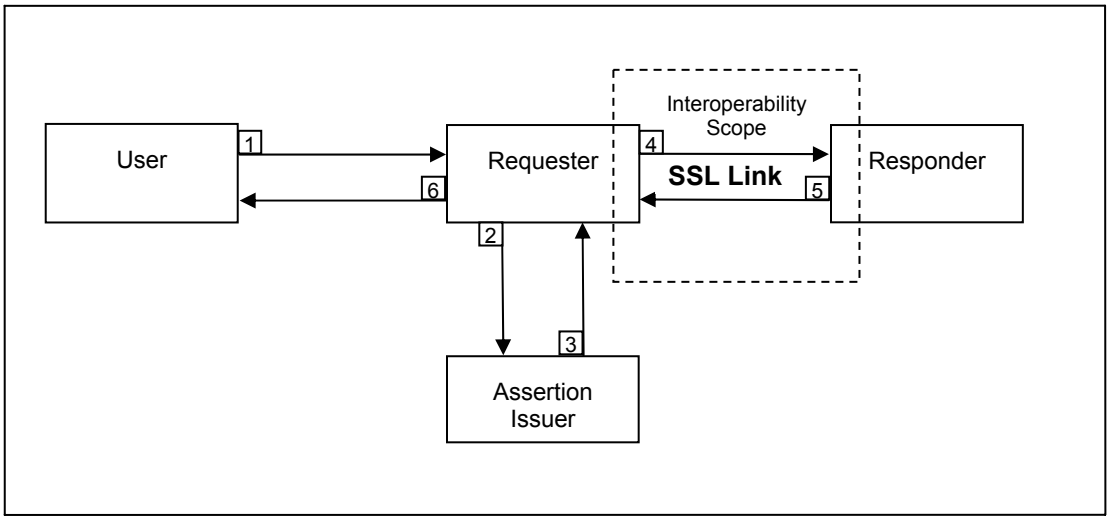
4.3 General Message Flow

This section provides a general overview of the flow of messages.

This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used. As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string may be used. The recipient SHOULD ignore the value. The request contains a plain text SAML assertion which contains one valid SubjectStatement of any valid type. The Responder checks the issuer name in the SAML assertion and determines if it is valid against the local directory of trusted issuers. If no errors are detected it returns the response without any security mechanisms.

4.3.1 Message exchange overview

This section contains a high level diagram of the scenario including the actors and the basis of trust. Interoperability for all scenarios is between the Requester and Responder. For each scenario, a hypothetical set of actions that take place prior to the Requester sending the request will be described in order to give some context for the assembly of the request and to show where the basis of trust lies for the Responder. However, the interoperability aspect of each scenario consists solely of the Request that the Requester sends to the Responder and the Response that the Responder returns to the Requester.



In the **Sender-Vouches: SSL** scenario the basis of trust is the Requester's client certificate used to establish the SSL link. The Responder relies on the Requester who vouches for the contents of the User message and the SAML Assertion. In typical scenarios the Requester and Assertion Issuer may be the same party however this is not a requirement and does not impact the interoperability characteristics of this scenario.

In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a potential before and after context for the operability test which only includes steps 4 and 5.

1. User sends a SOAP request to the Requester
2. Requester requests a SAML assertion for this User from the Assertion Issuer.
3. Assertion Issuer returns a Sender-Vouches SAML assertion for the User to the Requester.
4. Requester inserts the Assertion to a wsse:Security header in the SOAP message as described in the "First Message" section below and sends the Request to the Responder.
5. Responder processes Request as described in "Message Processing" section below and returns Response to Requester as described in "Second Message" section below.
6. Requester returns Response to User.

4.4 First Message - Request

4.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed. Items marked as optional MAY be generated and MUST be processed if generated. Items MUST appear in the order specified, except as noted.

Name	**Mandatory?**
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Issuer	Mandatory
Conditions	Mandatory
NotBefore	Mandatory
NotOnOrAfter	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
Body	Mandatory

4.4.2 Message Creation

4.4.2.1 Security

The Security element MUST contain the mustUnderstand="1" atttibute.

4.4.2.2 Timestamp

The Created element within the Timestamp SHOULD contain the current local time at the sender expressed in the UTC time zone.

4.4.2.3 Assertion

The Assertion element MUST contain an Issuer attribute, whose value MUST match an Issuer value in the ISSUERNAME-LIST.

4.4.2.3.1 Conditions

The Conditions element MUST be present and contain valid values for the NotBefore and NotOnOrAfter attributes.

4.4.2.3.2 SubjectStatement

The Assertion element MUST contain one SubjectStatement, which may be any of the standard SAML SubjectStatement extensions.

4.4.2.3.3 ConfirmationMethod

The Subject element MUST also contain a SubjectConfirmation element which MUST contain a ConfirmationMethod with a value of “urn:oasis:names:tc:SAML:cm:sender-vouches”.

4.4.2.4 Body

The Body is not signed or encrypted in any way.

4.4.3 Message Processing

This section describes the processing performed by the Responder. If an error is detected, the Responder MUST cease processing the message. If an error is detected, a SOAP Fault MAY be returned and, if so, it MUST have a value of InvalidSecurityToken.

4.4.3.1 Security

4.4.3.2 Timestamp

The Timestamp element MUST be ignored.

4.4.3.3 Assertion

The Assertion element MUST be validated. The value of the Issuer attribute MUST match an entry in the ISSUERNAME-LIST.

4.4.3.3.1 Conditions

As part of validating the Assertion element, the NotBefore and NotOnOrAfter attributes MUST be consistent with the current UTC time.

4.4.3.3.2 SubjectStatement

The Responder MUST check that at least one of the standard SAML SubjectStatement elements is included in the Assertion. Any elements or attributes in the statement other than those explicitly described here SHOULD be ignored.

4.4.3.3.3 ConfirmationMethod

The value of the ConfirmationMethod element MUST be “urn:oasis:names:tc:SAML:cm:sender-vouches”.

4.4.3.4 Body

The Body is passed to the application without modification.

4.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-01.xsd"
  xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1">
      <wsu:Timestamp>
        <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
      </wsu:Timestamp>
      <saml:Assertion
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
          MajorVersion="1" MinorVersion="0"
          AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
          Issuer="www.example.com"
          IssueInstant="2002-06-19T16:58:33.173Z">
        <saml:Conditions
            NotBefore="2002-06-19T16:53:33.173Z"
            NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
        <saml:AuthenticationStatement
         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
         AuthenticationInstant="2002-06-19T16:57:30.000Z">
          <saml:Subject>
            <saml:NameIdentifier
                NameQualifier="www.example.com"
                Format="">
              uid=joe,ou=people,ou=saml-demo,o=example.com
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
              </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
          </saml:Subject>
        </saml:AuthenticationStatement>
```

```
      </saml:Assertion>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </Ping>
 </soap:Body>
</soap:Envelope>
```

4.5 Second Message - Response

4.5.1 Message Elements and Attributes

Items not listed in the following table MUST NOT be created or processed. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Body	Mandatory

4.5.2 Message Creation

The response message MUST NOT contain a <wsse:Security> header. Any other header elements MUST NOT be labeled with a mustUnderstand="1" attribute.

4.5.3 Message Processing

The Body is passed to the application without modification.

4.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PingResponse xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </PingResponse>
 </soap:Body>
</soap:Envelope>
```

4.6 Other processing

This section describes processing that occurs outside of generating or processing a message.

4.6.1 Requester

No additional processing is required.

4.6.2 Responder

No additional processing is required.

4.7 Expected Security Properties

Use of the service is restricted to Requesters that can obtain and include a SAML assertion with an Issuer name that is on the Responder's ISSUERNAMELIST. In addition, the Requester must use a client certificate on an SSL link that the Responder can validate.

5 Scenario #3 – Sender-Vouches: Signed

The request contains a sender-vouches SAML assertion. The Assertion and the Body elements are signed. A reference to the certificate used to verify the signature is provided in the header. The response does not contain a security header.

5.1 Agreements

This section describes the agreements that must be made, directly or indirectly between parties who wish to interoperate.

5.1.1 ISSUERNAME-LIST

This is a list of trusted issuers of SAML assertions. (This is optional in the signed scenario since the basis for trust is the Requester certificate that is used for signing.)

5.1.2 Signature Trust Root

This refers generally to agreeing on at least one trusted key and any other certificates and sources of revocation information sufficient to validate certificates sent for the purpose of signature verification.

5.1.3 REQUESTER-CERT-VALUE

This is an opaque identifier indicating the X.509 certificate to be used by the Requester. The Responder MUST have the necessary trusted certificates in the Signature Trust Root to find and validate the Requester certificate.

5.2 Parameters

This section describes parameters that are required to correctly create or process messages, but not a matter of mutual agreement.

No parameters required.

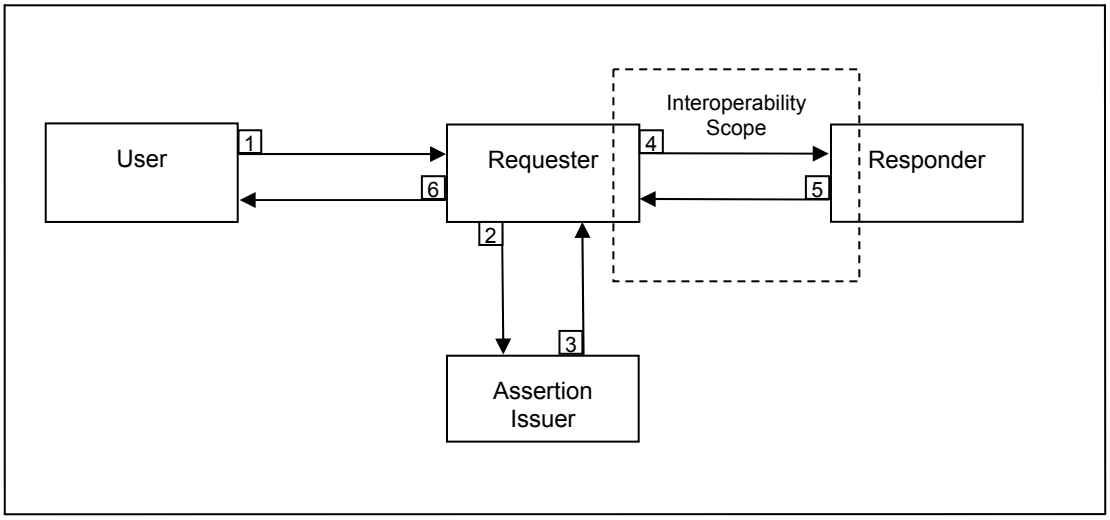
5.3 General Message Flow

This section provides a general overview of the flow of messages.

This contract covers a request/response MEP over the HTTP binding. SOAP 1.1 MUST be used. As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string may be used. The recipient SHOULD ignore the value. The request contains a plain text unsigned SAML assertion. The Responder checks the issuer name in the SAML assertion, validates the Requester certificate referenced in the signature, and verifies the signature. If no errors are detected, the message is delivered to the application and the application returns the response without any security mechanisms.

5.3.1 Message exchange overview

This section contains a high level diagram of the scenario including the actors and the basis of trust. Interoperability for all scenarios is between the Requester and Responder. For each scenario, a hypothetical set of actions that take place prior to the Requester sending the request will be described in order to give some context for the assembly of the request and to show where the basis of trust lies for the Responder. However, the interoperability aspect of each scenario consists solely of the Request that the Requester sends to the Responder and the Response that the Responder returns to the Requester.



In the **Sender-Vouches: Signed** scenario the basis of trust is the Requester's certificate. The Requester private key is used to sign both the SAML Assertion and the message Body. The Responder relies on the Requester who vouches for the contents of the User message and the SAML Assertion. In typical scenarios the Requester and Assertion Issuer may be the same party however this is not a requirement and does not impact the interoperability characteristics of this scenario.

In the following sequence, steps 1,2,3,6 are for illustrative purposes only and represent a potential before and after context for the operability test which only includes steps 4 and 5.

1. User sends a SOAP request to the Requester
2. Requester requests a SAML assertion for this user from the Assertion Issuer.
3. Assertion Issuer returns a Sender-Vouches SAML assertion to the Requester.
4. Requester inserts the Assertion to a wsse:Security header in the SOAP message and signs the Assertion and message Body as described in the "First Message" section below and sends the Request to the Responder.
5. Responder processes Request as described in "Message Processing" section below and returns Response to Requester as described in "Second Message" section below.
6. Requester returns Response to User.

5.4 First Message - Request

5.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Conditions	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory

5.4.2 Message Creation

5.4.2.1 Security

The Security element MUST contain the mustUnderstand="1" attribute.

5.4.2.2 Timestamp

The Created element within the Timestamp SHOULD contain the current local time at the sender.

585 #### 5.4.2.3 Signature

586 The signature is over the entire SOAP Body plus over the SAML Assertion.

587 ##### 5.4.2.3.1 SignedInfo

588 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
589 CanonicalizationMethod MUST be Exclusive Canonicalization. There MUST be 2 Reference
590 elements, one of which MUST specify a relative URI that refers to the SOAP Body element, and
591 the other which MUST specify a relative URI that refers to the Assertion element.

592 ##### 5.4.2.3.2 SignatureValue

593 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
594 private key corresponding to the public key specified in the certificate identified by the KeyInfo
595 element.

596 ##### 5.4.2.3.3 KeyInfo

597 The KeyInfo MUST contain an X509SubjectName that identifies the Requester certificate which
598 will be used for signature verification.

599 #### 5.4.2.4 Assertion

600 The Assertion MUST be signed and be labeled with an ID (AssertionID) that is referenced in a
601 SignedInfo Reference element of the Signature in the Security element.

602 The Conditions element MUST be present and contain valid NotBefore and NotOnOrAfter
603 attributes

604 A SubjectStatement element MUST be present and MUST contain a SubjectConfirmation
605 element which MUST contain a ConfirmationMethod element with a value of
606 “urn:oasis:tc:SAML:cm:sender-vouches”.

607 #### 5.4.2.5 Body

608 The Body element MUST be signed.

609 ### 5.4.3 Message Processing

610 This section describes the processing performed by the Responder. If an error is detected, the
611 Responder MUST cease processing the message and issue a Fault with a value of
612 InvalidSecurityToken.

613 #### 5.4.3.1 Security

614 #### 5.4.3.2 Timestamp

615 The Timestamp element MUST be ignored.

5.4.3.3 Signature

The Requester certificate MUST be obtained based on the KeyInfo/SecurityTokenReference and be validated against the Signature Trust Root. The Body and Assertion MUST be verified against the signature using the specified algorithms and transforms and the retained public key.

5.4.3.4 Assertion

The Assertion element MUST be validated. The value of the Issuer attribute MAY match the Subject of the certificate in the X509SubjectName. If it does not, then the value of the Issuer attribute MUST match an entry in the ISSUERNAME-LIST.

5.4.3.5 Body

The body is passed to the application without modification.

5.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<S12:Envelope
  xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-01.xsd"
  xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
 <S12:Header>
  <wsse:Security>
   <wsu:Timestamp>
     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
   </wsu:Timestamp>
   <saml:Assertion
     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
     IssueInstant="2003-04-17T00:46:02Z"
     Issuer="www.opensaml.org"
     MajorVersion="1"
     MinorVersion="1">
    <saml:Conditions
      NotBefore="2002-06-19T16:53:33.173Z"
      NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
    <saml:AttributeStatement>
     <saml:Subject>
      <saml:NameIdentifier
        NameQualifier="www.example.com"
        Format="">
       uid=joeuser,ou=system,ou=saml-demo,o=xyz.com
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
```

```
      <saml:ConfirmationMethod>
       urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
      </saml:ConfirmationMethod>
     </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute>
     ...
    </saml:Attribute>
    ...
   </saml:AttributeStatement>
  </saml:Assertion>
  <wsse:SecurityTokenReference wsu:id="STR1">
   <wsse:KeyIdentifier wsu:id="..."
    wsse:ValueType="http://www.docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-
token-profile-1.0#SAMLAssertionID">
    _a75adf55-01d7-40cc-929f-dbd8372ebdfc
   </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
  <wsse:BinarySecurityToken
    wsu:Id="attesterCert"
    ValueType="wsse:X509V3"
    EncodingType="wsse:Base64Binary">
   MIIEZzCCA9CgAwIBAgIQEmtJZc0...
  </wsse:BinarySecurityToken>
  <ds:Signature>
   <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#STR1">
     <Transforms>
      <ds:Transform Algorithm="http://schemas.xmlsoap.org/ws/2003/06/STR-Transform"/>
      <wsse:TransformationParameters>
       <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </wsse:TransformationParameters>
     </Transforms>
     <ds:DigestMethod Algorithm= "http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>...</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#MsgBody">
     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>...</ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
   <ds:KeyInfo>
    <wsse:SecurityTokenReference wsu:id="STR2">
     <wsse:Reference wsu:id="..."
      wsse:URI="attesterCert"/>
    </wsse:SecurityTokenReference>
   </ds:KeyInfo>
  </ds:Signature>
 </wsse:Security>
</S12:Header>
<S12:Body wsu:Id="MsgBody">
```

```
  <ReportRequest>
   <TickerSymbol>SUNW</TickerSymbol>
  </ReportRequest>
 </S12:Body>
</S12:Envelope>
```

5.5 Second Message - Response

5.5.1 Message Elements and Attributes

Items not listed in the following table MUST NOT be created or processed. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Body	Mandatory

5.5.2 Message Creation

The response message must not contain a <wsse:Security> header. Any other header elements MUST NOT be labeled with a mustUnderstand="1" attribute.

5.5.3 Message Processing

The body is passed to the application without modification.

5.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PingResponse xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </PingResponse>
 </soap:Body>
</soap:Envelope>
```

5.6 Other processing

This section describes processing that occurs outside of generating or processing a message.

5.6.1 Requester

No additional processing is required.

5.6.2 Responder

5.7 Expected Security Properties

Use of the service is restricted to Requesters that can obtain and include a SAML assertion with an Issuer name that is on the Responder's ISSUERNAMELIST. In addition, the Requester must sign both the assertion and the body using the Requester's private key and certificate, which will protect the message against modification if it is intercepted or replayed.

6 Scenario #4 – Holder-of-Key

The request contains a holder-of-key SAML assertion. The assertion is signed by the assertion issuer with an enveloped signature. The certificate used to verify the issuer signature is contained within the assertion signature. The message body is signed by the Requester. The certificate used to verify the Requester's signature is contained in the assertion SubjectConfirmation. The response does not contain a security header.

6.1 Agreements

This section describes the agreements that must be made, directly or indirectly between parties who wish to interoperate.

6.1.1 ISSUERNAME-LIST

This is a list of trusted issuers of SAML assertions.

6.1.2 ASSERTIONISSUER-CERT-VALUE

This is an opaque identifier indicating the X.509 certificate to be used by the Assertion Issuer. The Responder MUST have the necessary trusted certificates in the Signature Trust Root to validate the Assertion Issuer certificate, which in turn is used to validate the User certificate which is contained in the assertion.

6.1.3 Signature Trust Root

This refers generally to agreeing on at least one trusted key and any other certificates and sources of revocation information sufficient to validate certificates sent for the purpose of signature verification.

6.2 Parameters

This section describes parameters that are required to correctly create or process messages, but not a matter of mutual agreement.

No parameters are required.

6.3 General Message Flow

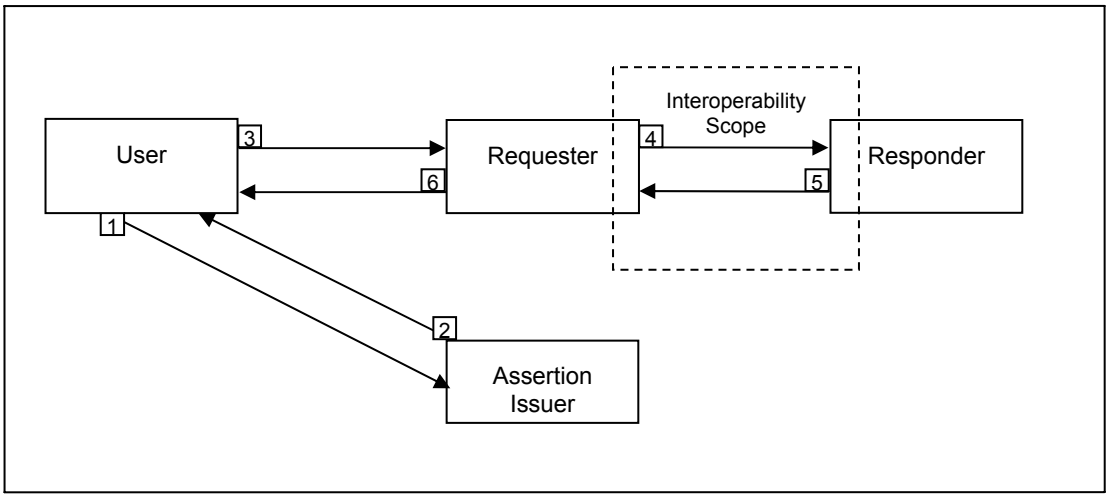
This section provides a general overview of the flow of messages.

This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used. As required by SOAP 1.1, the SOAPAction HTTP header MUST be present. Any value, including a null string may be used. The recipient SHOULD ignore the value. The request contains a plain text signed SAML assertion containing the User certificate. The Responder validates the assertion data, verifies the signature on the assertion, validates the issuer certificate contained in that signature, and verifies the signature on the Body using the user certificate/public key which is in the assertion. The order of the previous steps is not significant, although all steps MUST be

performed. If no errors are detected, the message is delivered to the application and the application returns the response without any security mechanisms.

6.3.1 Message exchange overview

This section contains a high level diagram of the scenario including the actors and the basis of trust. Interoperability for all scenarios is between the Requester and Responder. For each scenario, a hypothetical set of actions that take place prior to the Requester sending the request will be described in order to give some context for the assembly of the request and to show where the basis of trust lies for the Responder. However, the interoperability aspect of each scenario consists solely of the Request that the Requester sends to the Responder and the Response that the Responder returns to the Requester.



In the **Holder-of-Key: Signed** scenario the basis of trust is the Assertion Issuer's certificate. The Assertion Issuer's private key is used to sign the SAML Assertion for the User. The Responder relies on the Assertion Issuer to have issued the assertion to an authorized User. In typical scenarios the User and Requester may be the same party however this is not a requirement and does not impact the interoperability characteristics of this scenario.

1. User sends a request for a SAML assertion to the Assertion Issuer.
2. The Assertion Issuer authenticates the User and returns a SAML assertion containing the User certificate and signed with the Assertion Issuer private key.
3. User inserts the Assertion to a wsse:Security header in the SOAP message and signs the message Body with the User private key, and sends the message to the Requester.
4. Requester receives message from User which has form as described in the "First Message" section below and sends the Request to the Responder.
5. Responder processes Request as described in "Message Processing" section below and returns Response to Requester as described in "Second Message" section below.
6. Requester returns Response to User.

6.4 First Message - Request

6.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	**Mandatory?**
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	Mandatory
Assertion	Mandatory
Conditions	Mandatory
SubjectStatement	Mandatory
Subject	Mandatory
SubjectConfirmation	Mandatory
ConfirmationMethod	Mandatory
KeyInfo	Mandatory
Signature (Assertion)	Mandatory
SignedInfo	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Signature (Body)	Mandatory
SignedInfo	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory

824 ### 6.4.2 Message Creation

825 #### 6.4.2.1 Security

826 The Security element MUST contain the mustUnderstand="1" attribute.

827 #### 6.4.2.2 Timestamp

828 The Created element within the Timestamp SHOULD contain the current local time at the sender.

829 #### 6.4.2.3 Assertion

830 The Assertion MUST contain an ISSUER attribute, whose value MUST match an Issuer value in
831 the ISSUERNAME-LIST.

832 ##### 6.4.2.3.1 Conditions

833 The Conditions element MUST be present and contain valid values for the NotBefore and
834 NotOnOrAfter attributes.

835 ##### 6.4.2.3.2 SubjectConfirmation

836 The SubjectConfirmation element MUST contain both a ConfirmationMethod element and a
837 KeyInfo element.

838 ###### 6.4.2.3.2.1 ConfirmationMethod

839 The ConfirmationMethod element MUST have a value of
840 "urn:oasis:names:tc:SAML:1.0:cm:holder-of-key".

841 ###### 6.4.2.3.2.2 KeyInfo

842 The KeyInfo element MUST contain the user's public key, which MUST be contained in either an
843 X509Data element or a KeyValue element. (Note: because this KeyInfo is part of the SAML
844 assertion token, it is outside the direct scope of WS-Security and does not contain a
845 SecurityTokenReference.)

846 ##### 6.4.2.3.3 Signature (Assertion)

847 This Signature element is a child of the Assertion element and the signature is over the Assertion
848 element

849 ###### 6.4.2.3.3.1 SignedInfo

850 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
851 CanonicalizationMethod SHOULD be Exclusive Canonicalization. If not, the Assertion element
852 MAY need to be removed from the request before the signature is verified. There MUST be 1
853 Reference element, which MUST specify a relative URI of "" indicating the parent of the Signature
854 element. The Reference element MUST contain a Transform element with an Algorithm attribute
855 with the value "http://www.w3.org/2000/09/xmldsig#enveloped-signature".

856 ##### 6.4.2.3.3.2 SignatureValue

857 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
858 private key corresponding to the public key contained in the KeyInfo element in this Signature
859 element.

860 ##### 6.4.2.3.3.3 KeyInfo

861 This KeyInfo element MUST contain a Base 64 representation of the Issuer's certificate. The
862 certificate MUST contain the Issuer's public key, which can be used to verify the SignatureValue
863 in this Signature element. (Note: because this KeyInfo is part of the SAML assertion token, it is
864 outside the direct scope of WS-Security and does not contain a SecurityTokenReference.)

865 #### 6.4.2.4 Signature (Body)

866 This Signature element is a child of the Security element and the signature is over the entire
867 SOAP body.

868 ##### 6.4.2.4.1 SignedInfo

869 The SignatureMethod MUST be RSA-SHA1. The DigestMethod MUST be SHA-1. The
870 CanonicalizationMethod MUST be Exclusive Canonicalization. There MUST be 1 Reference
871 element which MUST specify a relative URI that refers to the SOAP Body element.

872 ##### 6.4.2.4.2 SignatureValue

873 The SignatureValue MUST be calculated as specified by the SignatureMethod element, using the
874 private key corresponding to the public key specified in the KeyInfo element that is in the
875 Assertion element.

876 ##### 6.4.2.4.3 KeyInfo

877 This KeyInfo element must contain a SecurityTokenReference that contains a KeyIdentifier
878 element that has a ValueType attribute with a value of "saml:Assertion". The KeyIdentifier MUST
879 have a value equal to the value of the AssertionID attribute of the Assertion element.

880 #### 6.4.2.5 Body

881 The body element MUST be signed.

882 ### 6.4.3 Message Processing

883 This section describes the processing performed by the Responder. If an error is detected, the
884 Responder MUST cease processing the message and issue a Fault with a value of
885 FailedAuthentication.

886 #### 6.4.3.1 Timestamp

887 The Timestamp element MUST be ignored.

6.4.3.2 Security

6.4.3.3 Assertion

The Assertion element MUST be validated. The value of the Issuer attribute SHOULD match the Subject of the certificate used to verify the signature contained in the assertion. If it does not, the not, then the value of the Issuer attribute MUST match an entry in the ISSUERNAME-LIST. The user public key or certificate MUST be obtained from the KeyInfo element of the SubjectConfirmation element and retained for verification of the Body signature.

6.4.3.3.1 Signature (Assertion)

The Assertion MUST be verified against the signature it contains, using the specified algorithms and transforms and the public key that can be obtained from the assertion issuer certificate contained in the Signature element. The issuer certificate MUST be validated against the Signature Trust Root.

6.4.3.4 Signature (Body)

The Body MUST be verified against the Signature element that is a child of the Security element using the user public key that was retained from the Assertion.

6.4.3.5 Body

The Body is passed to the application without modification.

6.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<S12:Envelope
  xmlns:S12="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsse="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-01.xsd"
  xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <S12:Header>
  <wsse:Security>
   <wsu:Timestamp>
     <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
   </wsu:Timestamp>
   <saml:Assertion
     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
     IssueInstant="2003-04-17T00:46:02Z"
     Issuer="www.opensaml.org"
     MajorVersion="1"
     MinorVersion="1"
     xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
```

```
    <saml:Conditions
      NotBefore="2002-06-19T16:53:33.173Z"
      NotOnOrAfter="2002-06-19T17:08:33.173Z"/>
    <saml:AttributeStatement>
     <saml:Subject>
      <saml:NameIdentifier
        NameQualifier="www.example.com"
        Format="">
       uid=joe,ou=people,ou=saml-demo,o=baltimore.com
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
       <saml:ConfirmationMethod>
        urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
       </saml:ConfirmationMethod>
       <ds:KeyInfo>
        <ds:KeyValue>...</ds:KeyValue>
       </ds:KeyInfo>
      </saml:SubjectConfirmation>
     </saml:Subject>
     <saml:Attribute
       AttributeName="MemberLevel"
       AttributeNamespace="http://www.oasis-open.org/Catalyst2002/attributes">
      <saml:AttributeValue>gold</saml:AttributeValue>
     </saml:Attribute>
     <saml:Attribute
       AttributeName="E-mail"
       AttributeNamespace="http://www.oasis-
       open.org/Catalyst2002/attributes">
      <saml:AttributeValue>joe@yahoo.com</saml:AttributeValue>
     </saml:Attribute>
    </saml:AttributeStatement>
   <ds:Signature>...</ds:Signature>
  </saml:Assertion>
  <ds:Signature>
   <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#MsgBody">
     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <ds:DigestValue>GyGsF0Pi4xPU...</ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>HJJWbvqW9E84vJVQk...</ds:SignatureValue>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference wsu:id="STR1">
      <wsse:Reference wsu:id="..."
       wsse:ValueType="http://www.docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-
token-profile-1.0#SAMLAssertion-1.0"
       wsse:URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc"/>
      </wsse:SecurityTokenReference>
     </ds:KeyInfo>
    </ds:Signature>
   </wsse:Security>
```

```
 </S12:Header>
<S12:Body wsu:Id="MsgBody">
 <ReportRequest>
  <TickerSymbol>SUNW</TickerSymbol>
 </ReportRequest>
</S12:Body>
</S12:Envelope>
```

6.5 Second Message - Response

6.5.1 Message Elements and Attributes

Items not listed in the following table MUST NOT be created or processed. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Body	Mandatory

6.5.2 Message Creation

The response message must not contain a <wsse:Security> header. Any other header elements MUST NOT be labeled with a mustUnderstand="1" attribute.

6.5.3 Message Processing

The body is passed to the application without modification.

6.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PingResponse xmlns="http://xmlsoap.org/Ping">
   <text>EchoString</text>
  </PingResponse>
 </soap:Body>
</soap:Envelope>
```

6.6 Other processing

This section describes processing that occurs outside of generating or processing a message.

6.6.1 Requester

No additional processing is required.

6.6.2 Responder

6.7 Expected Security Properties

Use of the service is restricted to authorized parties that sign the Body of the request. The Body of the request is protected against modification and interception. The authorized party must also obtain a signed SAML assertion containing the authorized party's certificate and include the assertion in the request as the basis for trust of the authorized party's certificate.

7 References

7.1 Normative

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

Appendix A. Ping Application WSDL File

```
<definitions xmlns:tns="http://xmlsoap.org/Ping"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://xmlsoap.org/Ping" name="Ping">
   <types>
         <schema targetNamespace="http://xmlsoap.org/Ping"
xmlns="http://www.w3.org/2001/XMLSchema">
               <complexType name="ping">
                     <sequence>
                           <element name="text" type="xsd:string"
nillable="true"/>
                     </sequence>
               </complexType>
               <complexType name="pingResponse">
                     <sequence>
                           <element name="text" type="xsd:string"
nillable="true"/>
                     </sequence>
               </complexType>
               <element name="Ping" type="tns:ping"/>
               <element name="PingResponse" type="tns:pingResponse"/>
         </schema>
   </types>
   <message name="PingRequest">
         <part name="ping" element="tns:Ping"/>
   </message>
   <message name="PingResponse">
         <part name="pingResponse" element="tns:PingResponse"/>
   </message>
   <portType name="PingPort">
         <operation name="Ping">
               <input message="tns:PingRequest"/>
               <output message="tns:PingResponse"/>
         </operation>
   </portType>
   <binding name="PingBinding" type="tns:PingPort">
         <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
         <operation name="Ping">
               <soap:operation/>
               <input>
                     <soap:body use="literal"/>
               </input>
               <output>
                     <soap:body use="literal"/>
               </output>
         </operation>
   </binding>
   <service name="PingService">
         <port name="PingPort" binding="tns:PingBinding">
               <soap:address
location="http://localhost:8080/pingejb/Ping"/>
         </port>
   </service>
</definitions>
```

1085

Appendix B. Revision History

Rev	Date	By Whom	What
wss-saml-01	2003-10-15	Rich Levinson	Initial version
wss-saml-02	2003-10-22	Rich Levinson	Added holder-of-key use case plus minor updates to other sections.
Wss-saml-03	2003-11-7	Rich Levinson	Update w comments from Ron Monzillo, add ssl use case, remove userlist checking, use cert refs, add intro framework, clarify issuerlist plus other specific technical adjustments.
Wss-saml-04	2004-1-29	Rich Levinson	Update minor comments. Bring in synch with WS-SAML Profile (V9 27-Jan-04)

Appendix C. Notices