Web Services Security: SAML Token Profile 1.1

OASIS Standard, 1 February 2006

Document Identifier:
wss-v1.1-spec-os-SAMLTokenProfile

OASIS Identifier:
{*WSS: SOAP Message Security* }-{SAMLTokenProfile}-{*1.1*} (OpenOffice) (PDF) (HTML)

Location:
This Version: http://docs.oasis-open.org/wss/oasis-wss-SAMLTokenProfile-1.1

Previous Version:http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0

Technical Committee:
OASIS Web Services Security (WSS) TC

Chairs:

Kelvin Lawrence	IBM
Chris Kaler	Microsoft

Editors

Ronald Monzillo	Sun
Chris Kaler	Microsoft
Anthony Nadalin	IBM
Phillip Hallem-Baker	VeriSign

Abstract:

This document describes how to use Security Assertion Markup Language (SAML) V1.1 and V2.0 assertions with the Web Services Security (WSS): SOAP Message Security V1.1 specification.

With respect to the description of the use of SAML V1.1, this document subsumes and is totally consistent with the Web Services Security: SAML Token Profile 1.0 and includes all corrections identified in the 1.0 errata.

Status:

This document is an OASIS Standard. It was approved by the OASIS membership on 1 February 2006. Check the location noted below for possible errata to this specification.

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 www.oasis-open.org/committees/wss.

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 (www.oasis-open.org/committees/wss/ipr.php).

The non-normative errata for this specification is located at www.oasis-open.org/committees/wss.

Notices

Table of Contents

1 Introduction

The WSS: SOAP Message Security specification defines a standard set of SOAP extensions that implement SOAP message authentication and encryption. This specification defines the use of Security Assertion Markup Language (SAML) assertions as security tokens from the `<wsse:Security>` header block defined by the WSS: SOAP Message Security specification.

1.1 Goals

The goal of this specification is to define the use of SAML V1.1 and V2.0 assertions in the context of WSS: SOAP Message Security including for the purpose of securing SOAP messages and SOAP message exchanges. To achieve this goal, this profile describes how:

1. SAML assertions are carried in and referenced from `<wsse:Security>` Headers.
2. SAML assertions are used with XML signature to bind the subjects and statements of the assertions (i.e., the claims) to a SOAP message.

1.1.1 Non-Goals

The following topics are outside the scope of this document:

1. Defining SAML statement syntax or semantics.
2. Describing the use of SAML assertions other than for SOAP Message Security.
3. Describing the use of SAML V1.0 assertions with the Web Services Security (WSS): SOAP Message Security specification.

2 Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

The keywords "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.

This document uses the notational conventions defined in the WS-Security SOAP Message Security document.

Namespace URIs (of the general form "some-URI") represent some application-dependent or context-dependent URI as defined in RFC2396.

This specification is designed to work with the general SOAP message structure and message processing model, and should be applicable to any version of SOAP. The current SOAP 1.2 namespace URI is used herein to provide detailed examples, but there is no intention to limit the applicability of this specification to a single version of SOAP.

Readers are presumed to be familiar with the terms in the Internet Security Glossary.

2.2 Namespaces

The appearance of the following [XML-ns] namespace prefixes in the examples within this specification should be understood to refer to the `corresponding` namespaces (from the following table) whether or not an XML namespace declaration appears in the example:

Prefix	Namespace
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
ds	http://www.w3.org/2000/09/xmldsig#
xenc	http://www.w3.org/2001/04/xmlenc
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsse11	http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
saml	**urn: oasis:names:tc:SAML:1.0:assertion**
saml2	**urn: oasis:names:tc:SAML:2.0:assertion**
samlp	**urn: oasis:names:tc:SAML:1.0:protocol**
xsi	http://www.w3.org/2001/XMLSchema-instance

Table-1 Namespace Prefixes

2.3 Terminology

This specification employs the terminology defined in the WSS: SOAP Message Security specification. The definitions for additional terminology used in this specification appear below.

Attesting Entity – the entity that provides the confirmation evidence that will be used to establish the correspondence between the subjects and claims of SAML statements (in SAML assertions) and SOAP message content.

Confirmation Method Identifier – the value within a SAML SubjectConfirmation element that identifies the subject confirmation process to be used with the corresponding statements.

Subject Confirmation – the process of establishing the correspondence between the subject and claims of SAML statements (in SAML assertions) and SOAP message content by verifying the confirmation evidence provided by an attesting entity.

SAML Assertion Authority - A *system entity* that issues *assertions*.

Subject – A representation of the entity to which the claims in one or more SAML statements apply.

3 Usage

This section defines the specific mechanisms and procedures for using SAML assertions as security tokens.

3.1 Processing Model

This specification extends the token-independent processing model defined by the WSS: SOAP Message Security specification.

When a receiver processes a `<wsse:Security>` header containing or referencing SAML assertions, it selects, based on its policy, the signatures and assertions that it will process. It is assumed that a receiver's signature selection policy MAY rely on semantic labeling[1] of `<wsse:SecurityTokenReference>` elements occurring in the `<ds:KeyInfo>` elements within the signatures. It is also assumed that the assertions selected for validation and processing will include those referenced from the `<ds:KeyInfo>` and `<ds:SignedInfo>` elements of the selected signatures.

As part of its validation and processing of the selected assertions, the receiver MUST[2] establish the relationship between the subject and claims of the SAML statements (of the referenced SAML assertions) and the entity providing the evidence to satisfy the confirmation method defined for the statements (i.e., the attesting entity). Two methods for establishing this correspondence, `holder-of-key` and `sender-vouches` are described below. Systems implementing this specification MUST implement the processing necessary to support both of these subject confirmation methods.

3.2 SAML Version Differences

The following sub-sections describe the differences between SAML V1.1 and V2.0 that apply to this specification.

3.2.1 Assertion Identifier

In SAML V1.1 the name of the assertion identifier attribute is "AssertionID". In SAML v2.0 the name of the assertion identifier attribute is "ID". In both versions the type of the identifier attribute is **`xs:ID`**.

3.2.2 Relationship of Subjects to Statements

A SAML assertion contains a collection of 0 or more statements. In SAML V1.1, a separate subject with separate subject confirmation methods may be specified for each statement of an assertion. In SAML V2.0, at most one subject and at most one set of subject confirmation methods may be specified for all the statements of the assertion. These distinctions are described in more detail by the following paragraphs.

A SAML V1.1 statement that contains a `<saml:Subject>` element (i.e., a subject statement) may contain a `<saml:SubjectConfirmation>` element that defines the rules for confirming the subject and claims of the statement. If present, the `<saml:SubjectConfirmation>` element occurs within the subject element, and defines one or more methods (i.e., `<saml:ConfirmationMethod>` elements) by which the statement may be confirmed and will include a `<ds:KeyInfo>`[3] element when any of the specified methods are based on demonstration of a confirmation key. The

---

[1] The optional `Usage` attribute of the `<wsse:SecurityTokenReference>` element MAY be used to associate one of more semantic usage labels (as URIs) with a reference and thus use of a Security Token. Please refer to WSS: SOAP Message Security for the details of this attribute.

[2] When the confirmation method is urn:`oasis:names:tc:SAML:1.0:cm:bearer`, proof of the relationship between the attesting entity and the subject of the statements in the assertion is implicit and no steps need be taken by the receiver to establish this relationship.

[3] When a `<ds:KeyInfo>` element is specified, it identifies the key that applies to all the key confirmed methods of the confirmation element.

<saml:SubjectConfirmation> element also provides for the inclusion of additional information to be applied in the confirmation method processing via the optional <saml:SubjectConfirmationData> element. The following example depicts a SAML V1.1 assertion containing two subject statements with different subjects and different subject confirmation elements.

```
<saml:Assertion xmlns:saml="..." xmlns:ds="..."
   MajorVersion="1" MinorVersion="1" >
      ...
  <saml:SubjectStatement>
     <saml:Subject>
        <saml:NameIdentifier>
         ...
        </saml:NameIdentifier>
        <saml:SubjectConfirmation>
           <saml:ConfirmationMethod>
              urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
           </saml:ConfirmationMethod>
           <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:SubjectStatement>
  <saml:SubjectStatement>
     <saml:Subject>
        <saml:NameIdentifier>
         ...
        </saml:NameIdentifier>
        <saml:SubjectConfirmation>
           <saml:ConfirmationMethod>
              urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
           </saml:ConfirmationMethod>
         </saml:SubjectConfirmation>
     </saml:Subject>
            ... .
   </saml:SubjectStatement>
   …
</saml:Assertion>
```

A SAML V2.0 assertion may contain a single <saml2:Subject> that applies to all the statements of the assertion. When a subject is included in A SAML V2.0 assertion, it may contain any number of <saml2:SubjectConfimation> elements, satisfying any of which is sufficient to confirm the subject and all the statements of the assertion. Each <saml2:SubjectConfirmation> element identifies a single confirmation method (by attribute value) and may include an optional <saml2:SubjectConfirmationData> element that is used to specify optional confirmation method independent condition attributes and to define additional method specific confirmation data. In the case of a key dependent confirmation method, a complex schema type, saml2:KeyInfoConfirmationDataType, that includes 1 or more <ds:KeyInfo> elements, can be specified as the xsi:type of the <saml2:SubjectConfirmationData> element. In this case, each <ds:KeyInfo> element identifies a key that may be demonstrated to confirm the assertion. The following example depicts a SAML V2.0 assertion containing a subject with multiple confirmation elements that apply to all the statements of the assertion.

```
 <saml2:Assertion xmlns:saml2="..." xmlns:ds="..." xmlns:xsi="...">
   <saml2:Subject>
      <saml2:NameID>
         …
     </saml2:NameID>
     <saml2:SubjectConfirmation
```

```
         Method=”urn:oasis:names:tc:SAML:2.0:cm:sender-vouches”>
         <saml2:SubjectConfirmationData>
          Address=”129.148.9.42”
         </saml2:SubjectConfirmationData>
      </saml2:SubjectConfirmation>
      <saml2:SubjectConfirmation
         Method=”urn:oasis:names:tc:SAML:2.0:cm:holder-of-key”>
         <saml2:SubjectConfirmationData
            xsi:type="saml2:KeyInfoConfirmationDataType">
            <ds:KeyInfo>
               <ds:KeyValue>…</ds:KeyValue>
            </ds:KeyInfo>
         </saml2:SubjectConfirmationData>
      </saml2:SubjectConfirmation>
   </saml2:Subject>
             ….
   <saml2:Statement>
   …
   </saml2:Statement>

   <saml2:Statement>
   …
   </saml2:Statement>
   …

</saml2:Assertion>
```

3.2.3 Assertion URI Reference Replaces AuthorityBinding

SAML V1.1 defines the (deprecated) `<saml:AuthorityBinding>` element so that a relying party can locate and communicate with an assertion authority to acquire a referenced assertion.

The `<saml:AuthorityBinding>` element was removed from SAML V2.0. [SAMLBindV2] requires that an assertion authority support a URL endpoint at which an assertion will be returned in response to an HTTP request with a single query string parameter named ID.

For example, if the documented endpoint at an assertion authority is:

```
https://saml.example.edu/assertion-authority
```

then the following request will cause the assertion with ID “abcde” to be returned:

```
https://saml.example.edu/assertion-authority?ID=abcde
```

3.2.4 Attesting Entity Identifier

The `<saml2:SubjectConfirmation>` element of SAML V2.0 provides for the optional inclusion of an element (i.e., `NameID`) to identify the expected attesting entity as distinct from the subject of the assertion.

```
<saml2:SubjectConfirmation xmlns:saml2="..."
  Method=”urn:oasis:names:tc:SAML:2.0:cm:sender-vouches”>
   <NameID>
      gateway
   </NameID>
   <saml2:SubjectConfirmationData>
      Address=”129.148.9.42”
   </saml2:SubjectConfirmationData>
</saml2:SubjectConfirmation>
```

3.3 Attaching Security Tokens

SAML assertions are attached to SOAP messages using WSS: SOAP Message Security by placing assertion elements or references to assertions inside a `<wsse:Security>` header. The following example illustrates a SOAP message containing a bearer confirmed SAML V1.1 assertion in a `<wsse:Security>` header.

```
<S12:Envelope xmlns:S12="...">
  <S12:Header>
    <wsse:Security xmlns:wsse="...">
     <saml:Assertion xmlns:saml="..."
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1">
       <saml:AuthenticationStatement>
          <saml:Subject>
            <saml:NameIdentifier
              NameQualifier="www.example.com"
              Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName">
              uid=joe,ou=people,ou=saml-demo,o=baltimore.com
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:bearer
              </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
          </saml:Subject>
        </saml:AuthenticationStatement>

      </saml:Assertion>

 </wsse:Security>
  </S12:Header>
  <S12:Body>
    . . .
  </S12:Body>
</S12:Envelope>
```

3.4 Identifying and Referencing Security Tokens

The WSS: SOAP Message Security specification defines the `<wsse:SecurityTokenReference>` element for referencing security tokens. Three forms of token references are defined by this element and the element schema includes provision for defining additional reference forms should they be necessary. The three forms of token references defined by the `<wsse:SecurityTokenReference>` element are defined as follows:

- A key identifier reference – a generic element (i.e., `<wsse:KeyIdentifier>)` that conveys a security token identifier as an `wsse:EncodedString` and indicates in its attributes (as necessary) the key identifier type (i.e., the `ValueType)`, the identifier encoding type (i.e., the `EncodingType)`, and perhaps other parameters used to reference the security token.

  When a key identifier is used to reference a SAML assertion, it MUST contain as its element value the corresponding SAML assertion identifier. The key identifier MUST also contain a `ValueType` attribute and the value of this attribute MUST be the value from Table 2 corresponding to the version of the referenced assertion. The key identifier MUST NOT include an `EncodingType`[4] attribute and the element content of the key identifier MUST be encoded as `xs:string`.

  When a key identifier is used to reference a V1.1 SAML assertion that is not contained in the same message as the key identifier, a `<saml:AuthorityBinding>` element MUST be contained in the `<wsse:SecurityTokenReference>` element containing the key identifier. The contents of the `<saml:AuthorityBinding>` element MUST contain values sufficient for the intended recipients of

[4] "The Errata for Web Services Security: SOAP Message Security Version 1.0" (at http://www.oasis-open.org/committees/wss) removed the default designation from the #Base64Binary value for the `EncodingType` attribute of the `KeyIdentifier` element. Therefore, omitting a value for `EncodingType` and requiring that Base64 encoding not be performed, as specified by this profile, is consistent with the WS-Security Specification (including V1.1).

the `<wsse:SecurityTokenReference>` to acquire the identified assertion from the intended Authority. To this end, the value of the `AuthorityKind` attribute of the `<saml:AuthorityBinding>` element MUST be "`samlp:AssertionIdReference`".

When a key Identifier is used to reference a SAML assertion contained in the same message as the key identifier, a `<saml:AuthorityBinding>` element MUST NOT be included in the `<wsse:SecurityTokenReference>` containing the key identifier.

A key identifier MUST NOT be used to reference a SAML V2.0 assertion if the assertion is NOT contained in the same message as the key identifier.

- A Direct or URI reference – a generic element (i.e., `<wsse:Reference>`) that identifies a security token by URI. If only a fragment identifier is specified, then the reference is to the security token within the document whose local identifier (e.g., `wsu:Id` attribute) matches the fragment identifier. Otherwise, the reference is to the (potentially external) security token identified by the URI.

  A reference to a SAML V2.0 assertion that is NOT contained in the same message MUST be a Direct or URI reference. In this case, the value of the URI attribute must conform to the URI syntax defined in section 3.7.5.1 of [SAMLBindV2]. That is, an HTTP or HTTPS request with a single query string parameter named ID. The reference MUST also contain a `wsse11:TokenType` attribute and the value of this attribute MUST be the `value` from Table 3 identifying the assertion as a SAML V2.0 security token. When a Direct reference is made to a SAML V2.0 Assertion, the Direct reference SHOULD NOT contain a `ValueType` attribute.

  This profile does not describe the use of Direct or URI references to reference V1.1 SAML assertions.

- An Embedded reference – a reference that encapsulates a security token.

  When an Embedded reference is used to encapsulate a SAML assertion, the SAML assertion MUST be included as a contained element within a `<wsse:Embedded>` element within a `<wsse:SecurityTokenReference>`.

This specification describes how SAML assertions may be referenced in four contexts:

- A SAML assertion may be referenced directly from a `<wsse:Security>` header element. In this case, the assertion is being conveyed by reference in the message.
- A SAML assertion may be referenced from a `<ds:KeyInfo>` element of a `<ds:Signature>` element in a `<wsse:Security>` header. In this case, the assertion contains a `SubjectConfirmation` element that identifies the key used in the signature calculation.
- A SAML assertion reference may be referenced from a `<ds:Reference>` element within the `<ds:SignedInfo>` element of a `<ds:Signature>` element in a `<wsse:Security>` header. In this case, the doubly-referenced assertion is signed by the containing signature.
- A SAML assertion reference may occur as encrypted content within an `<xenc:EncryptedData>` element referenced from a `<xenc:DataReference>` element within an `<xenc:ReferenceList>` element. In this case, the assertion reference (which may contain an embedded assertion) is encrypted.

In each of these contexts, the referenced assertion may be:

- local – in which case, it is included in the `<wsse:Security>` header containing the reference.
- remote – in which case it is not included in the `<wsse:Security>` header containing the reference, but may occur in another part of the SOAP message or may be available at the location identified by the reference which may be an assertion authority.

A SAML key identifier reference MUST be used for all (local and remote) references to SAML 1.1 assertions. All (local and remote) references to SAML V2.0 assertions SHOULD be by Direct reference and all remote references to V2.0 assertions MUST be by Direct reference URI. A key identifier reference MAY be used to reference a local V2.0 assertion. To maintain compatibility with Web Services Security: SOAP Message Security 1.0**,** the practice of referencing local SAML 1.1 assertions by Direct `<wsse:SecurityTokenReference>` reference is not defined by this profile.

Every key identifier, direct, or embedded reference to a SAML assertion SHOULD contain a `wsse11:TokenType` attribute and the value of this attribute MUST be the `value` from Table 3 that identifies the type and version of the referenced security token. When the referenced assertion is a SAML V2.0 Assertion the reference MUST contain a `wsse11:TokenType` attribute (as described above).

Assertion Version	Value
V1.1	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID
V2.0	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID

Table-2 Key Identifier ValueType Attribute Values

Assertion Version	Value
V1.1	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1
V2.0	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0

Table-3 `TokenType` Attribute Values

The following subsections define the SAML assertion references that MUST be supported by conformant implementations of this profile. A conformant implementation may choose to support the reference forms corresponding to either or both V1.1 or V2.0 SAML assertions.

3.4.1 SAML Assertion Referenced from Header or Element

All conformant implementations MUST be able to process SAML assertion references occurring in a `<wsse:Security>` header or in a header element other than a signature to acquire the corresponding assertion. A conformant implementation MUST be able to process any such reference independent of the confirmation method of the referenced assertion.

A SAML assertion may be referenced from a `<wsse:Security>` header or from an element (other than a signature) in the header. The following example demonstrates the use of a key identifier in a `<wsse:Security>` header to reference a local SAML V1.1 assertion.

```
<S12:Envelope xmlns:S12="...">
  <S12:Header>
    <wsse:Security xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="...">
      <saml:Assertion xmlns:saml="..."
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2003-04-17T00:46:02Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1">
      </saml:Assertion>
      <wsse:SecurityTokenReference wsu:Id="STR1"
        wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1">
        <wsse:KeyIdentifier wsu:Id="…"
          ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID">
          _a75adf55-01d7-40cc-929f-dbd8372ebdfc
        </wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
    </wsse:Security>
  </S12:Header>
  <S12:Body>
    . . .
  </S12:Body>
</S12:Envelope>
```

The following example depicts the use of a key identifier reference to reference a local SAML V2.0 assertion.

```
<wsse:SecurityTokenReference
    xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
   wsu:Id="STR1"
    wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0">
   <wsse:KeyIdentifier wsu:Id="…"
      ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLID">
        _a75adf55-01d7-40cc-929f-dbd8372ebdfc
   </wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
```

A SAML V1.1 assertion that exists outside of a `<wsse:Security>` header may be referenced from the `<wsse:Security>` header element by including (in the `<wsse:SecurityTokenReference>`) a `<saml:AuthorityBinding>` element that defines the location, binding, and query that may be used to acquire the identified assertion at a SAML assertion authority or responder.

```
<wsse:SecurityTokenReference
    xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
    wsu:Id="STR1"
   wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1">
  <saml:AuthorityBinding xmlns:saml="..."
    Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
    Location="http://www.opensaml.org/SAML-Authority"
    AuthorityKind= "samlp:AssertionIdReference"/>
  <wsse:KeyIdentifier
    wsu:Id="…"
    ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#SAMLAssertionID">
    _a75adf55-01d7-40cc-929f-dbd8372ebdfc
  </wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
```

The following example depicts the use of a Direct or URI reference to reference a SAML V2.0 assertion that exists outside of a `<wsse:Security>` header.

```
<wsse:SecurityTokenReference
     xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
    wsu:Id="…"
    wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0">
  <wsse:Reference
    wsu:Id="…"
    URI="https://saml.example.edu/assertion-authority?ID=abcde">
  </wsse:Reference>
</wsse:SecurityTokenReference>
```

3.4.2 SAML Assertion Referenced from KeyInfo

All conformant implementations MUST be able to process SAML assertion references occurring in the `<ds:KeyInfo>` element of a `<ds:Signature>` element in a `<wsse:Security>` header as defined by the holder-of-key confirmation method.

The following example depicts the use of a key identifier to reference a local V1.1 assertion from `<ds:KeyInfo>`.

```
<ds:KeyInfo xmlns:ds="...">
  <wsse:SecurityTokenReference
     xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
     wsu:Id="STR1"
    wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1">
    <wsse:KeyIdentifier wsu:Id="…"
```

```
      ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#SAMLAssertionID”>
      _a75adf55-01d7-40cc-929f-dbd8372ebdfc
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

A local, V2.0 assertion may be referenced by replacing the values of the Key Identifier `ValueType` and reference `TokenType` attributes with the values defined in tables 2 and 3 (respectively) for SAML V2.0 as follows:

```
<ds:KeyInfo xmlns:ds="...">
  <wsse:SecurityTokenReference
    xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
    wsu:Id=”STR1”
    wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0”>
    <wsse:KeyIdentifier wsu:Id=”…”
      ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLID”>
      _a75adf55-01d7-40cc-929f-dbd8372ebdfc
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

The following example demonstrates the use of a `<wsse:SecurityTokenReference>` containing a key identifier and a `<saml:AuthorityBinding>` to communicate information (location, binding, and query) sufficient to acquire the identified V1.1 assertion at an identified SAML assertion authority or responder.

```
<ds:KeyInfo xmlns:ds="...">
  <wsse:SecurityTokenReference
    xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
    wsu:Id=”STR1”
    wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1”>
    <saml:AuthorityBinding xmlns:saml="..."
      Binding=”urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding”
      Location=”http://www.opensaml.org/SAML-Authority”
      AuthorityKind= “samlp:AssertionIdReference”/>
    <wsse:KeyIdentifier wsu:Id=”…”
      ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#SAMLAssertionID”>
_a75adf55-01d7-40cc-929f-dbd8372ebdfc
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Remote references to V2.0 assertions are made by Direct reference URI. The following example depicts the use of a Direct reference URI to reference a remote V2.0 assertion from `<ds:KeyInfo>`.

```
<ds:KeyInfo xmlns:ds="...">
  <wsse:SecurityTokenReference
       xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..."
       wsu:id=”STR1”
       wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0”>
    <wsse:Reference
       wsu:id=”…”
       URI=”https://saml.example.edu/assertion-authority?ID=abcde”>
     </wsse:Reference>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

`<ds:KeyInfo>` elements may also occur in `<xenc:EncryptedData>` and `<xenc:EncryptedKey>` elements where they serve to identify the encryption key. `<ds:KeyInfo>` elements may also occur in SAML `SubjectConfirmation` elements where they identify a key that MUST be demonstrated to confirm the subject of the corresponding statement(s).

Conformant implementations of this profile are NOT required to process SAML assertion references occurring within the `<ds:KeyInfo>` elements within `<xenc:EncryptedData>`, `<xenc:EncryptedKey>`, or SAML `SubjectConfirmation` elements.

3.4.3 SAML Assertion Referenced from SignedInfo

Independent of the confirmation method of the referenced assertion, all conformant implementations MUST be able to process SAML assertions referenced by `<wsse:SecurityTokenReference>` from `<ds:Reference>` elements within the `<ds:SignedInfo>` element of a `<ds:Signature>` element in a `<wsse:Security>` header. Embedded references may be digested directly, thus effectively digesting the encapsulated assertion. Other `<wsse:SecurityTokenReference>` forms must be dereferenced for the referenced assertion to be digested.

The core specification, WSS: SOAP Message Security, defines the STR Dereference transform to cause the replacement (in the digest stream) of a `<wsse:SecurityTokenReference>` with the contents of the referenced token. To digest any SAML assertion that is referenced by a non-embedded `<wsse:SecurityTokenReference>`, the STR Dereference transform MUST be specified and applied in the processing of the <ds:Reference>. Conversly, the STR Dereference transform MUST NOT be specified or applied when the `<wsse:SecurityTokenReference>`, not the referenced assertion, is to be digested.

The following example demonstrates the use of the STR Dereference transform to dereference a reference to a SAML V1.1 Assertion (i.e., Security Token) such that the digest operation is performed on the security token not its reference.

```
<wsse:SecurityTokenReference
   xmlns:wsse="..." xmlns:wsu="..." xmlns:wsse11="..." wsu:Id=”STR1”
   wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1”>
  <saml:AuthorityBinding xmlns:saml="..."
    Binding=”urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding”
    Location=”http://www.opensaml.org/SAML-Authority”
    AuthorityKind= “samlp:AssertionIdReference”/>
  <wsse:KeyIdentifier wsu:Id=”…”
    ValueType=”http://docs.oasis-open.org/wss/oasis-2004XX-wss-saml-token-
profile-1.0#SAMLAssertionID”>
     _a75adf55-01d7-40cc-929f-dbd8372ebdfc
  </wsse:KeyIdentifier>
</wsse:SecurityTokenReference

<ds:SignedInfo xmlns:ds="..." xmlns:wsse="...">
  <ds:CanonicalizationMethod
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <ds:SignatureMethod
    Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <ds:Reference URI="#STR1">
    <Transforms>
      <ds:Transform
        Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#STR-Transform”>
        <wsse:TransformationParameters>
          <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </wsse:TransformationParameters>
      </ds:Transform>
    </Transforms>
    <ds:DigestMethod
      Algorithm= "http://www.w3.org/2000/09/xmldsig#sha1"/>

 <ds:DigestValue>...</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>
```

Note that the URI appearing in the `<ds:Reference>` element identifies the `<wsse:SecurityTokenReference>` element by its `wsu:Id` value. Also note that the STR Dereference transform MUST contain (in `<wsse:TransformationParameters>`) a `<ds:CanonicalizationMethod>` that defines the algorithm to be used to serialize the input node set (of the referenced assertion).

As depicted in the other examples of this section, this profile establishes `<wsse:SecurityTokenReference>` forms for referencing V1.1, local V2.0, and remote V2.0 assertions.

3.4.4 SAML Assertion Referenced from Encrypted Data Reference

Independent of the confirmation method of the referenced assertion, all conformant implementations MUST be able to process SAML assertion references occurring as encrypted content within the `<xenc:EncryptedData>` elements referenced by Id from the `<xenc:DataReference>` elements of `<xenc:ReferenceList>` elements. An `<xenc:ReferenceList>` element may occur either as a top-level element in a `<wsse:Security>` header, or embedded within an `<xenc:EncryptedKey>` element. In either case, the `<xenc:ReferenceList>` identifies the encrypted content.

Such references are similar in format to the references that MAY appear in the `<ds:Reference>` element within `<ds:SignedInfo>`, except the STR Dereference transform does not apply. As shown in the following example, an encrypted `<wsse:SecurityTokenReference>` (which may contain an embedded assertion) is referenced from an `<xenc:DataReference>` by including the identifier of the `<xenc:EncryptedData>` element that contains the encrypted `<wsse:SecurityTokenReference>` in the `<xenc:DataReference>`.

```
<xenc:EncryptedData xmlns:xenc="..." xmlns:ds="..." Id=”EncryptedSTR1”>
  <ds:KeyInfo>
  . . .
  </ds:KeyInfo>
  <xenc:CipherData>
    <xenc:CipherValue>...</xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>
<xenc:ReferenceList xmlns:xenc="...">
  <xenc:DataReference URI="#EncryptedSTR1"/>
</xenc:ReferenceList>
```

3.4.5 SAML Version Support and Backward Compatibility

An implementation of this profile MUST satisfy all of its requirements with respect to either or both SAML V1.1 or SAML V2.0 Assertions. An implementation that satisfies the requirements of this profile with respect to SAML V1.1 assertions MUST be able to fully interoperate with any fully compatible implementation of version 1.0 of this profile.

An implementation that does not satisfy the requirements of this profile with respect to SAML V1.1 or SAML V2.0 assertions MUST reject a message containing a `<wsse:Security>` header that references or conveys an assertion of the unsupported version. When a message containing an unsupported assertion version is detected, the receiver MAY choose to respond with an appropriate fault as defined in Section 3.6, “Error Codes”.

3.5 Subject Confirmation of SAML Assertions

The SAML profile of WSS: SOAP Message Security requires that systems support the holder-of-key and sender-vouches methods of subject confirmation. It is strongly RECOMMENDED that an XML signature be used to establish the relationship between the message and the statements of the attached assertions.

This is especially RECOMMENDED whenever the SOAP message exchange is conducted over an unprotected transport.

Any processor of SAML assertions MUST conform to the required validation and processing rules defined in the corresponding SAML specification including the validation of assertion signatures, the processing of `<saml:Condition>` elements within assertions, and the processing of `<saml2:SubjectConfirmationData>` attributes. [SAMLCoreV1] defines the validation and processing rules for V1.1 assertions, while [SAMLCoreV2] is authoritative for V2.0 assertions.

The following table enumerates the mandatory subject confirmation methods and summarizes their associated processing models:

Mechanism	RECOMMENDED Processing Rules

`Urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`
`Or`
`urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`
	The attesting entity demonstrates knowledge of a confirmation key identified in a holder-of-key `SubjectConfirmation` element within the assertion.

`Urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`
`Or`
`urn:oasis:names:tc:SAML:2.0:cm:sender-vouches`
	The attesting entity, (presumed to be) different from the subject, vouches for the verification of the subject. The receiver MUST have an existing trust relationship with the attesting entity. The attesting entity MUST protect the assertion in combination with the message content against modification by another party. See also section 4.

Note that the high level processing model described in the following sections does not differentiate between the attesting entity and the message sender as would be necessary to guard against replay attacks. The high-level processing model also does not take into account requirements for authentication of receiver by sender, or for message or assertion confidentiality. These concerns must be addressed by means other than those described in the high-level processing model (i.e., section 3.1).

3.5.1 Holder-of-key Subject Confirmation Method

The following sections describe the holder-of-key method of establishing the correspondence between a SOAP message and the subject and claims of SAML assertions added to the SOAP message according to this specification.

3.5.1.1 Attesting Entity

An attesting entity demonstrates that it is authorized to act as the subject of a holder-of-key confirmed SAML statement by demonstrating knowledge of any key identified in a holder-of-key `SubjectConfirmation` element associated with the statement by the assertion containing the statement. Statements attested for by the holder-of-key method MUST be associated, within their containing assertion, with one or more holder-of-key `SubjectConfirmation` elements.

The `SubjectConfirmation` elements MUST include a `<ds:KeyInfo>` element that identifies a public or secret key[5] that can be used to confirm the identity of the subject.

To satisfy the associated confirmation method processing to be performed by the message receiver, the attesting entity MUST demonstrate knowledge of the confirmation key. The attesting entity MAY accomplish this by using the confirmation key to sign content within the message and by including the resulting `<ds:Signature>` element in the `<wsse:Security>` header. `<ds:Signature>` elements produced for this purpose MUST conform to the canonicalization and token pre-pending rules defined in the WSS: SOAP Message Security specification.

SAML assertions that contain a holder-of-key `SubjectConfirmation` element SHOULD contain a `<ds:Signature>` element that protects the integrity of the confirmation `<ds:KeyInfo>` established by the assertion authority.

The canonicalization method used to produce the `<ds:Signature>` elements used to protect the integrity of SAML assertions MUST support the validation of these `<ds:Signature>` elements in contexts (such as `<wsse:Security>` header elements) other than those in which the signatures were calculated.

3.5.1.2 Receiver

Of the SAML assertions it selects for processing, a message receiver MUST NOT accept statements of these assertions based on a holder-of-key `SubjectConfirmation` element defined for the statements (within the assertion) unless the receiver has validated the integrity of the assertion and the attesting entity has demonstrated knowledge of a key identified within the confirmation element.

If the receiver determines that the attesting entity has demonstrated knowledge of a subject confirmation key, then the subjects and claims of the SAML statements confirmed by the key MAY be attributed to the attesting entity and any content of the message whose integrity is protected by the key MAY be considered to have been provided by the attesting entity.

3.5.1.3 Example V1.1

The following example illustrates the use of the holder-of-key subject confirmation method to establish the correspondence between the SOAP message and the subject of statements of the SAML V1.1 assertions in the `<wsse:Security>` header:

```
<?xml version="1.0" encoding="UTF-8"?>
<S12:Envelope xmlns:S12="..." xmlns:wsu="...">
  <S12:Header>

    <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." xmlns:ds="...">
      <saml:Assertion xmlns:saml="..."
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2005-05-27T16:53:33.173Z"
        Issuer="www.opensaml.org"
        MajorVersion="1"
        MinorVersion="1">
        <saml:Conditions
          NotBefore="2005-05-27T16:53:33.173Z"
          NotOnOrAfter="2005-05-27T16:58:33.17302Z"/>
        <saml:AttributeStatement>
          <saml:Subject>
            <saml:NameIdentifier
```

[5][SAMLCoreV1] defines `KeyInfo` of `SubjectConfirmation` as containing a "cryptographic key held by the subject". Demonstration of this key is sufficient to establish who is (or may act as the) subject. Moreover, since it cannot be proven that a confirmation key is known (or known only) by the subject whose identity it establishes, requiring that the key be held by the subject is an untestable requirement that adds nothing to the strength of the confirmation mechanism. In [SAMLCoreV2], the OASIS Security Services Technical Committee agreed to remove the phrase "held by the subject" from the definition of `KeyInfo` within `SubjectConfirmation(Data)`.

```
              NameQualifier="www.example.com"
              Format=“urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName”>
              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#rsa-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”
            wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1”>
            <wsse:KeyIdentifier wsu:Id=”…”
              ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID”>
              _a75adf55-01d7-40cc-929f-dbd8372ebdfc
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S12:Header>

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

3.5.1.4 Example V2.0

The following example illustrates the use of the holder-of-key subject confirmation method to establish the correspondence between the SOAP message and the subject of the SAML V2.0 assertion in the `<wsse:Security>` header:

```
<?xml version="1.0" encoding="UTF-8"?>
<S12:Envelope xmlns:S12="..." xmlns:wsu="...">
  <S12:Header>

    <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." xmlns:ds="...">
      <saml2:Assertion xmlns:saml2="..." xmlns:xsi="..."
        ID=”_a75adf55-01d7-40cc-929f-dbd8372ebdfc”>
      <saml2:Subject>
       <saml2:NameID>
          …
       </saml2:NameID>
       <saml2:SubjectConfirmation
            Method=”urn:oasis:names:tc:SAML:2.0:cm:holder-of-key”>
            <saml2:SubjectConfirmationData
                xsi:type="saml2:KeyInfoConfirmationDataType">
               <ds:KeyInfo>
                  <ds:KeyValue>…</ds:KeyValue>
               </ds:KeyInfo>
            </saml2:SubjectConfirmationData>
         </saml2:SubjectConfirmation>
      </saml2:Subject>
      <saml2:Statement>
       …
      </saml2:Statement>
        <ds:Signature>…</ds:Signature>
      </saml2: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#rsa-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”
            wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV2.0”>
            <wsse:KeyIdentifier wsu:Id=”…”
              ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLID”>
              _a75adf55-01d7-40cc-929f-dbd8372ebdfc
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S12:Header>

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

3.5.2 Sender-vouches Subject Confirmation Method

The following sections describe the sender-vouches method of establishing the correspondence between a SOAP message and the SAML assertions added to the SOAP message according to the SAML profile of WSS: SOAP Message Security.

3.5.2.1 Attesting Entity

An attesting entity uses the sender-vouches confirmation method to assert that it is acting on behalf of the subject of SAML statements attributed with a sender-vouches `SubjectConfirmation` element. Statements attested for by the sender-vouches method MUST be associated, within their containing assertion, with one or more sender-vouches `SubjectConfirmation` elements.

To satisfy the associated confirmation method processing of the receiver, the attesting entity MUST protect the vouched for SOAP message content such that the receiver can determine when it has been altered by another party. The attesting entity MUST also cause the vouched for statements (as necessary) and their binding to the message contents to be protected such that unauthorized modification can be detected. The attesting entity MAY satisfy these requirements by including in the corresponding `<wsse:Security>` header a `<ds:Signature>` element that it prepares by using its key to sign the relevant message content and assertions. As defined by the XML Signature specification, the attesting entity MAY identify its key by including a `<ds:KeyInfo>` element within the `<ds:Signature>` element.

A `<ds:Signature>` element produced for this purpose MUST conform to the canonicalization and token pre-pending rules defined in the WSS: SOAP Message Security specification.

3.5.2.2 Receiver

Of the SAML assertions it selects for processing, a message receiver MUST NOT accept statements of these assertions based on a sender-vouches `SubjectConfirmation` element defined for the statements (within the assertion) unless the assertions and SOAP message content being vouched for are protected (as described above) by an attesting entity who is trusted by the receiver to act as the subjects and with the claims of the statements.

3.5.2.3 Example V1.1

The following example illustrates an attesting entity’s use of the sender-vouches subject confirmation method with an associated `<ds:Signature>` element to establish its identity and to assert that it has sent the message body on behalf of the subject(s) of the V1.1 assertion referenced by “STR1”.

The assertion referenced by “STR1” is not included in the message. “STR1” is referenced by `<ds:Reference>` from `<ds:SignedInfo>`. The `ds:Reference>` includes the STR-transform to cause the assertion, not the `<SecurityTokenReference>` to be included in the digest calculation. “STR1” includes a `<saml:AuthorityBinding>` element that utilizes the remote assertion referencing technique depicted in the example of section 3.3.3.

The SAML V1.1 assertion embedded in the header and referenced by “STR2” from `<ds:KeyInfo>` corresponds to the attesting entity. The private key corresponding to the public confirmation key occurring in the assertion is used to sign together the message body and assertion referenced by “STRI”.

```
<?xml version="1.0" encoding="UTF-8"?>
<S12:Envelope xmlns:S12="..." xmlns:wsu="...">

  <S12:Header>
    <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." xmlns:ds="...">

      <saml:Assertion xmlns:saml="..."
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2005-05-27T16:53:33.173Z"
        Issuer=”www.opensaml.org”
        MajorVersion="1"
        MinorVersion="1">
        <saml:Conditions
          NotBefore="2005-05-27T16:53:33.173Z"
          NotOnOrAfter="2005-05-27T16:58:33.173Z"/>
        <saml:AttributeStatement>
          <saml:Subject>
            <saml:NameIdentifier
              NameQualifier="www.example.com"
              Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName”>
              uid=proxy,ou=system,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>
          . . .
          </saml:Attribute>
          . . .
        </saml:AttributeStatement>
      </saml:Assertion>

      <wsse:SecurityTokenReference wsu:Id=”STR1”>
        wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1”>
        <saml:AuthorityBinding xmlns:saml="..."
          Binding=”urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding”
          Location=”http://www.opensaml.org/SAML-Authority”
          AuthorityKind="samlp:AssertionIdReference”/>
        <wsse:KeyIdentifier wsu:Id=”…”
          ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID”>
          _a75adf55-01d7-40cc-929f-dbd8372ebdbe
        </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>

      <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#rsa-sha1"/>
         <ds:Reference URI="#STR1">
            <Transforms>
              <ds:Transform
                Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#STR-Transform">
                <wsse:TransformationParameters>
                  <ds:CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </wsse:TransformationParameters>
              </ds:Transform>
            </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"
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1">
            <wsse:KeyIdentifier wsu:Id="…"
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID">
              _a75adf55-01d7-40cc-929f-dbd8372ebdfc
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S12:Header>

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

3.5.2.4 Example V2.0

The following example illustrates the mapping of the preceding example to SAML V2.0 assertions.

```
<?xml version="1.0" encoding="UTF-8"?>
<S12:Envelope xmlns:S12="..." xmlns:wsu="...">
  <S12:Header>

    <wsse:Security xmlns:wsse="..." xmlns:wsse11="..." xmlns:ds="...">
      <saml2:Assertion xmlns:saml2="..." xmlns:xsi="..."

        ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
        <saml2:Subject>
         <saml2:NameID>
          …
         </saml2:NameID>
         <saml2:SubjectConfirmation
               Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
               <saml2:SubjectConfirmationData
                  xsi:type="saml2:KeyInfoConfirmationDataType">
                  <ds:KeyInfo>
                     <ds:KeyValue>…</ds:KeyValue>
                  </ds:KeyInfo>
               </saml2:SubjectConfirmationData>
            </saml2:SubjectConfirmation>
         </saml2:Subject>
         <saml2:Statement>
          …
         </saml2:Statement>
         <ds:Signature>…</ds:Signature>
      </saml2:Assertion>

      <wsse:SecurityTokenReference wsu:Id="STR1"
        wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV2.0">
        <wsse:Reference wsu:Id="…"
          URI="https://www.opensaml.org?_a75adf55-01d7-40cc-929f-
dbd8372ebdbe">
        </wsse:Reference>
      </wsse:SecurityTokenReference>

      <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#rsa-sha1"/>
          <ds:Reference URI="#STR1">
            <Transforms>
              <ds:Transform

             Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#STR-Transform">
                <wsse:TransformationParameters>
                  <ds:CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </wsse:TransformationParameters>
              </ds:Transform>
            </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”
            wsse11:TokenType=”http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV2.0”>
            <wsse:KeyIdentifier wsu:Id=”…”
              ValueType=”http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLID”>
              _a75adf55-01d7-40cc-929f-dbd8372ebdfc
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S12:Header>

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

3.5.3 Bearer Confirmation Method

This profile does NOT require message receivers to establish the relationship between a received message and the statements of any bearer confirmed (i.e., confirmation method urn:`oasis:names:tc:SAML:1.0:cm:bearer)` assertions conveyed or referenced from the message. Conformant implementations of this profile MUST be able to process references and convey bearer assertions within `<wsse:Security>` headers. Any additional processing requirements that pertain specifically to bearer confirmed assertions are outside the scope of this profile.

3.6 Error Codes

When a system that implements the SAML token profile of WSS: SOAP Message Security does not perform its normal processing because of an error detected during the processing of a security header, it MAY choose to report the cause of the error using the SOAP fault mechanism. The SAML token profile of WSS: SOAP Message Security does not require that SOAP faults be returned for such errors, and systems that choose to return faults SHOULD take care not to introduce any security vulnerabilities as a result of the information returned in error responses.

Systems that choose to return faults SHOULD respond with the error codes and fault strings defined in the WSS: SOAP Message Security specification. The RECOMMENDED correspondence between the common assertion processing failures and the error codes defined in WSS: SOAP Message Security are defined in the following table:

Assertion Processing Error	**RECOMMENDED Error(Faultcode)**
A referenced SAML assertion could not be retrieved.	`wsse:SecurityTokenUnavailable`
An assertion contains a `<saml:Condition>` element that the receiver does not understand.	`wsse:UnsupportedSecurityToken`
A signature within an assertion or referencing an assertion is invalid.	`wsse:FailedCheck`
The issuer of an assertion is not acceptable to the receiver.	`wsse:InvalidSecurityToken`
The receiver does not understand the extension schema used in an assertion.	`wsse:UnsupportedSecurityToken`
The receiver does not support the SAML version of a referenced or included assertion.	`wsse:UnsupportedSecurityToken`

The preceding table defines fault codes in a form suitable for use with SOAP 1.1. The WSS: SOAP Message Security specification describes how to map SOAP 1.1 fault constructs to the SOAP 1.2 fault constructs.

4 Threat Model and Countermeasures (non-normative)

This document defines the mechanisms and procedures for securely attaching SAML assertions to SOAP messages. SOAP messages are used in multiple contexts, specifically including cases where the message is transported without an active session, the message is persisted, or the message is routed through a number of intermediaries. Such a general context of use suggests that users of this profile must be concerned with a variety of threats.

In general, the use of SAML assertions with WSS: SOAP Message Security introduces no new threats beyond those identified for SAML or by the WSS: SOAP Message Security specification. The following sections provide an overview of the characteristics of the threat model, and the countermeasures that SHOULD be adopted for each perceived threat.

4.1 Eavesdropping

Eavesdropping is a threat to the SAML token profile of WSS: SOAP Message Security in the same manner as it is a threat to any network protocol. The routing of SOAP messages through intermediaries increases the potential incidences of eavesdropping. Additional opportunities for eavesdropping exist when SOAP messages are persisted.

To provide maximum protection from eavesdropping, assertions, assertion references, and sensitive message content SHOULD be encrypted such that only the intended audiences can view their content. This approach removes threats of eavesdropping in transit, but MAY not remove risks associated with storage or poor handling by the receiver.

Transport-layer security MAY be used to protect the message and contained SAML assertions and/or references from eavesdropping while in transport, but message content MUST be encrypted above the transport if it is to be protected from eavesdropping by intermediaries.

4.2 Replay

Reliance on authority-protected (e.g., signed) assertions with a holder-of-key subject confirmation mechanism precludes all but a holder of the key from binding the assertions to a SOAP message. Although this mechanism effectively restricts data origin to a holder of the confirmation key, it does not, by itself, provide the means to detect the capture and resubmission of the message by other parties.

Assertions that contain a sender-vouches confirmation mechanism introduce another dimension to replay vulnerability if the assertions impose no restriction on the entities that may use or reuse the assertions.

Replay attacks can be detected by receivers if message senders include additional message identifying information (e.g., timestamps, nonces, and or recipient identifiers) within origin-protected message content and receivers check this information against previously received values.

4.3 Message Insertion

The SAML token profile of WSS: SOAP Message Security is not vulnerable to message insertion attacks.

4.4 Message Deletion

The SAML token profile of WSS: SOAP Message Security is not vulnerable to message deletion attacks.

4.5 Message Modification

Messages constructed according to this specification are protected from message modification if receivers can detect unauthorized modification of relevant message content. Therefore, it is strongly RECOMMENDED that all relevant and immutable message content be signed by an attesting entity. Receivers SHOULD only consider the correspondence between the subject of the SAML assertions and the SOAP message content to have been established for those portions of the message that are protected by the attesting entity against modification by another entity.

To ensure that message receivers can have confidence that received assertions have not been forged or altered since their issuance, SAML assertions appearing in or referenced from `<wsse:Security>` header elements MUST be protected against unauthorized modification (e.g., signed) by their issuing authority or the attesting entity (as the case warrants). It is strongly RECOMMENDED that an attesting entity sign any `<saml:Assertion>` elements that it is attesting for and that are not signed by their issuing authority.

Transport-layer security MAY be used to protect the message and contained SAML assertions and/or assertion references from modification while in transport, but signatures are required to extend such protection through intermediaries.

4.6 Man-in-the-Middle

Assertions with a holder-of-key subject confirmation method are not vulnerable to a MITM attack. Assertions with a sender-vouches subject confirmation method are vulnerable to MITM attacks to the degree that the receiver does not have a trusted binding of key to the attesting entity's identity.

5 References

[GLOSSARY]	Informational RFC 2828, "*Internet Security Glossary,*" May 2000.
[KEYWORDS]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," *RFC 2119*, Harvard University, March 1997
[SAMLBindV1]	Oasis Standard, E. Maler, P.Mishra, and R. Philpott (Editors), *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003.
[SAMLBindV2]	Oasis Standard, S. Cantor, F. Hirsch, J. Kemp, R. Philpott, E. Maler (Editors), *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
[SAMLCoreV1]	Oasis Standard, E. Maler, P.Mishra, and R. Philpott (Editors), *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003.
[SAMLCoreV2]	Oasis Standard, S. Cantor, J. Kemp, R. Philpott, E. Maler (Editors), *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005.
[SOAP]	W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
	W3C Working Draft, Nilo Mitra (Editor), *SOAP Version 1.2 Part 0: Primer*, June 2002.
	W3C Working Draft, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen (Editors), *SOAP Version 1.2 Part 1: Messaging Framework*, June 2002.
	W3C Working Draft, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen (Editors), *SOAP Version 1.2 Part 2: Adjuncts*, June 2002.
[URI]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," *RFC 2396*, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.
[WS-SAML]	Contribution to the WSS TC, P. Mishra (Editor), WS-Security Profile of the Security Assertion Markup Language (SAML) Working Draft 04, Sept 2002.

[WSS: SAML Token Profile] Oasis Standard, P. Hallem-Baker, A. Nadalin, C. Kaler, R. Monzillo (Editors), Web Services Security: SAML Token Profile 1.0, December 2004.

[WSS: SOAP Message Security V1.0] Oasis Standard, A. Nadalin, C.Kaler, P. Hallem-Baker, R. Monzillo (Editors), Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), August 2003.

[WSS: SOAP Message Security] Oasis Standard, A. Nadalin, C.Kaler, R. Monzillo, P. Hallem-Baker, (Editors), Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), December 2005.

[XML-ns]	W3C Recommendation, "Namespaces in XML," 14 January 1999.
[XML Signature]	W3C Recommendation, "XML Signature Syntax and Processing," 12 February 2002.
[XML Token]	Contribution to the WSS TC, Chris Kaler (Editor), WS-Security Profile for XML-based Tokens, August 2002.

1213 # Appendix A. Acknowledgments

Current Contributors:

Michael	Hu	Actional
Maneesh	Sahu	Actional
Duane	Nickull	Adobe Systems
Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Laboratory
Hal	Lockhart	BEA Systems
Denis	Pilipchuk	BEA Systems
Corinna	Witt	BEA Systems
Steve	Anderson	BMC Software
Rich	Levinson	Computer Associates
Thomas	DeMartini	ContentGuard
Merlin	Hughes	Cybertrust
Dale	Moberg	Cyclone Commerce
Rich	Salz	Datapower
Sam	Wei	EMC
Dana S.	Kaufman	Forum Systems
Toshihiro	Nishimura	Fujitsu
Kefeng	Chen	GeoTrust
Irving	Reid	Hewlett-Packard
Kojiro	Nakayama	Hitachi
Paula	Austel	IBM
Derek	Fu	IBM
Maryann	Hondo	IBM
Kelvin	Lawrence	IBM
Michael	McIntosh	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Bruce	Rich	IBM
Ron	Williams	IBM
Don	Flinn	Individual
Kate	Cherry	Lockheed Martin
Paul	Cotton	Microsoft
Vijay	Gajjala	Microsoft
Martin	Gudgin	Microsoft
Chris	Kaler	Microsoft
Frederick	Hirsch	Nokia
Abbie	Barbir	Nortel
Prateek	Mishra	Oracle
Vamsi	Motukuru	Oracle
Ramana	Turlapi	Oracle
Ben	Hammond	RSA Security
Rob	Philpott	RSA Security
Blake	Dournaee	Sarvega
Sundeep	Peechu	Sarvega
Coumara	Radja	Sarvega
Pete	Wenzel	SeeBeyond
Manveen	Kaur	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Symon	Chang	TIBCO Software
John	Weiland	US Navy
Hans	Granqvist	VeriSign

Phillip	Hallem-Baker	VeriSign
Hemma	Prafullchandra	VeriSign

Previous Contributors:

Peter	Dapkus	BEA
Guillermo	Lao	ContentGuard
TJ	Pannu	ContentGuard
Xin	Wang	ContentGuard
Shawn	Sharp	Cyclone Commerce
Ganesh	Vaideeswaran	Documentum
Tim	Moses	Entrust
Carolina	Canales-Valenzuela	Ericsson
Tom	Rutt	Fujitsu
Yutaka	Kudo	Hitachi
Jason	Rouault	HP
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Hiroshi	Maruyama	IBM
David	Melgar	IBM
Kent	Tamura	IBM
Wayne	Vicknair	IBM
Phil	Griffin	Individual
Mark	Hayes	Individual
John	Hughes	Individual
Peter	Rostin	Individual
Davanum	Srinivas	Individual
Bob	Morgan	Individual/Internet2
Bob	Atkinson	Microsoft
Keith	Ballinger	Microsoft
Allen	Brown	Microsoft
Giovanni	Della-Libera	Microsoft
Alan	Geller	Microsoft
Johannes	Klein	Microsoft
Scott	Konersmann	Microsoft
Chris	Kurt	Microsoft
Brian	LaMacchia	Microsoft
Paul	Leach	Microsoft
John	Manferdelli	Microsoft
John	Shewchuk	Microsoft
Dan	Simon	Microsoft
Hervey	Wilson	Microsoft
Jeff	Hodges	Neustar
Senthil	Sengodan	Nokia
Lloyd	Burch	Novell
Ed	Reed	Novell
Charles	Knouse	Oblix
Vipin	Samar	Oracle
Jerry	Schwarz	Oracle
Eric	Gravengaard	Reactivity
Andrew	Nash	Reactivity
Stuart	King	Reed Elsevier
Martijn	de Boer	SAP
Jonathan	Tourzan	Sony
Yassir	Elley	Sun
Michael	Nguyen	The IDA of Singapore
Don	Adams	TIBCO

Morten	Jorgensen	Vordel