Security Assertions Markup Language

Core Assertion Architecture

Phillip Hallam-Baker	*VeriSign*
Tim Moses	*Entrust*
Bob Morgan	*University of Washington*
Carlisle Adams	*Entrust*
Charles Knouse	*Oblix*
Irving Reid	*Baltimore*
Jeff Hodges	*Oblix*
Marlena Erdos	*Tivoli*
Nigel Edwards	*Hewlett Packard*
Prateek Mishra	*Netegrity*

Draft Version 0.7: May 14th 2001

Printed on Monday, May 14, 2001

Security Assertions Markup Language

Version 0.7

Table Of Contents

Executive Summary

This document contains two sections. Section 1 contains the text proposed by the Core Assertions & Protocol group for the Core Assertions section of the SAML. Section 2 contains references to the material cited in the text.

1 XML Assertion and Request Syntax

Each SAML protocol exchange consists of a request and response. The embedding of these requests and responses in specific protocols is described in detail in the section on Bindings.

The syntax of requests and responses are closely related and so both are described here.

1.1 Namepaces

For clarity, some examples of XML are not complete documents and namespace declarations may be omitted from XML fragments. In this document, certain namespace prefixes represent certain namespaces.

All SAML protocol elements are defined using XML schema **[XML-Schema1][XML-Schema2]**. For clarity unqualified elements in schema definitions are in the XML schema namespace:

xmlns="`http://www.w3.org/2000/10/XMLSchema`."

References to Security Assertion Markup Language schema defined herein use the prefix "s0" and are in the namespace:

xmlns:s0="`http://www.oasis.org/tbs/1066-12-25/`"

This namespace is also used for unqualified elements in message protocol examples.

The SAML schema specification uses some elements already defined in the XML Signature namespace. The "XML Signature namespace" is represented by the prefix `ds` and is declared as:

xmlns:ds="`http://www.w3.org/2000/09/xmldsig#`"

The "XML Signature schema" is defined in **[XML-SIG-XSD]** and the `<ds:KeyInfo>` element (and all of its contents) are defined in **[XML-SIG]** §4.4.

1.2 SAML Assertion

SAML specifies several different types of assertion for different purposes, these are:

Authentication Assertion

Attribute Assertion

Decision Assertion

The different types of SAML assertion are encoded in a common XML package which at a minimum consists of:

Basic Information.
Each assertion MUST specify a unique identifier that serves as a name for the assertion. In addition an assertion MAY specify the date and time of issue and the time interval for which the assertion is valid.

Claims.
The claims made by the assertion. This document describes the use of assertions to make claims for Authorization and Key Delegation applications.

In addition an assertion MAY contain the following additional elements. An SAML client is not required to support processing of any element contained in an additional element **with the sole exception that an SAML client MUST reject any assertion containing a Conditions element that is not supported.**

Conditions.
The assertion status MAY be subject to conditions. The status of the assertion might be dependent on additional information from a validation service. The assertion may be dependent on other assertions being valid. The assertion may only be valid if the relying party is a member of a particular audience.

Advice.
Assertions MAY contain additional information as advice. The advice element MAY be used to specify the assertions that were used to make a policy decision.

The SAML assertion package is designed to facilitate reuse in other specifications. For this reason XML elements specific to the management of authentication and authorization data are expressed as claims. Possible additional applications of the assertion package format include management of embedded trust roots **[XTASS]** and authorization policy information **[XACML]**.

1.2.1 Element <Assertion>

The <Assertion> element is specified by the following schema:

```
<element name="Assertion">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name="Version" type="string"/>
         <element name="AssertionID" type="s0:AssertionID"/>
         <element name="Issuer" type="string"/>
         <element name="IssueInstant" type="DateTime"/>
         <element name="ValidityInterval" type="s0:ValidityInterval"/>
```

```
        <!-- Data -->
        <element name="Claims" type="s0:Claims"/>
        <element name="Conditions" type="s0:Conditions"/>
        <element name="Advice" type="s0:Advice"/>

    </sequence>
  </complexType>
</element>
```

1.3 SAML Request

SAML Assertions may be generated and exchanged using a variety of protocols. The bindings section of this document describes specific means of transporting SAML assertions using existing widely deployed protocols.

SAML aware clients may in addition use the request protocol defined by the `<SAMLQuery>` and `<SAMLQueryResponse>` elements described in this section.

1.3.1 Element `<SAMLQuery>`

The query specifies the principal and the resources for which access is requested by use of the claim element syntax. The information requested in the response is specified by means of the `<Respond>` element described in section 1.3.3.

The `<SAMLQuery>` element is defined by the following schema:

```
<element name="SAMLQuery">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name="RequestID" type="s0:AssertionID"/>
         <element name="AssertionID" type="s0:AssertionID"/>
         <element name="ValidityInterval" type="s0:ValidityInterval"/>

         <!-- Data -->
         <element name="Query"       type="s0:Claims"/>
         <element name="Conditions"  type="s0:Conditions"/>
         <element name="Advice"      type="s0:Advice"/>

         <element name="Respond"     type="s0:Respond"/>
     </sequence>
  </complexType>
</element>
```

1.3.2 Element `<RequestID>`

The `RequestID` element defines a unique identifier for the assertion request. If an assertion query specifies a `RequestID` value the same value MUST be returned in the response unless a Respond element of `Static` is specified.

The `<RequestID>` element is defined by the following schema:

```
<element name="RequestID" type="string"/>
```

1.3.3 Element <Respond>

The <Respond> element in the request specifies one or more strings included in the request that specify data elements to be provided in the response.

The Service SHOULD return a requested data element if it is available. The Service MAY return additional data elements that were not requested. In particular, the service MAY return data elements specified in the request with the response.

Defined identifiers include:

Identifier	Description
Decision	Return the result of the Query (True/False).
Static	Specifies that the response may return any data element thus allowing the responder to return a static pre-signed assertion.
Assertion	Return any assertion element.
ValidityInterval	Return the ValidityInterval element
Conditions	Return the assertion conditions
Claims	Return all assertion claims matching the query.
Claims/Object/Attribute	Return all assertion claims matching the query whose object is an attribute.
Claims/Object/Role	Return all assertion claims matching the query whose object is a role.
Claims/Object/Resource	Return all assertion claims matching the query whose object is a resource.
Advice	Return additional advice elements
ResourceWildcard	Responses MAY identify a set of resources by means of a restricted wildcard.
XML Schema URI	If a URI is specified the response may contain Claims, Conditions and Advice elements specified by the corresponding XML schema.

The <Respond> element is defined by the following schema:

```
<element name="Respond" >
   <complexType>
      <sequence>
         <element name="string" type="string"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

1.3.4 Element <SAMLQueryResponse>

The response to a <SAMLQuery> is a <SAMLQueryResponse> element. This returns the <RequestID> specified in the response together with a <Decision> element and/or an <Assertion> element. The information returned in the response is controlled by the <Respond> element of the request.

The <SAMLQueryResponse> element is defined by the following schema:

```
<element name="SAMLQueryResponse">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name="RequestID"   type="s0:AssertionID"/>
         <element name="Decision"    type="s0:Decision"/>
         <element name="Assertion"   type="s0:Assertion"/>
     </sequence>
  </complexType>
</element>
```

1.3.5 Element <Decision>

The <Decision> element in the request specifies an authorization decision and has three possible values: Permit, Deny and Indeterminate.

The <Respond> element is defined by the following schema:

```
<simpleType name="Decision" base="string">
   <enumeration value="Permit"/>
   <enumeration value="Deny"/>
   <enumeration value="Indeterminate"/>
</simpleType>
```

1.4 Basic Information

Four basic information elements are defined; a unique identifier, the issuer, the time instant of issue, the validity interval and the assertion status.

1.4.1 Element <Version>

Each assertion MUST specify the SAML version identifier. The identifier for this version of SAML is "1.0".

The following schema defines the <Version> element:

```
<element name="Version" type="string"/>
```

1.4.2 Element <AssertionID>

Each assertion MUST specify exactly one unique assertion identifier. All identifiers are encoded as a Uniform Resource Identifier (URI) and are specified in full (use of relative identifiers is not permitted).

The URI is used as a *name* for the assertion and not as a *locator*. It is only necessary to ensure that no two assertions share the same identifier. Provision of a service to resolve an identifier into an assertion is not a requirement.

The following schema defines the <AssertionID> element:

```
<element name="AssertionID" type="string"/>
```

1.4.3 Element <Issuer>

The Issuer element specifies the issuer of the assertion by means of a URI. It is defined by the following XML schema:

The following schema defines the <Issuer> element:

```
<element name="Issuer" type="string"/>
```

1.4.4 Element <IssueInstant>

The time instant of issue.

The following schema defines the <Version> element:The <IssueInstant> element is defined by the following schema:

```
<element name="IssueInstant" type="timeInstant"/>
```

1.4.5 Element <ValidityInterval>

The <ValidityInterval> structure specifies limits on the validity of the assertion. It contains the following elements:

Member	Type	Description
NotBefore	DateTime	Time instant at which the validity interval begins
NotOnOrAfter	DateTime	Time instant at which the validity interval has ended

The DateTime instant MUST fully specify the date.

The NotBefore and NotOnOrAfter elements are optional. If the value is either omitted or equal to the start of the epoch it is unspecified. If the NotBefore element is

unspecified the assertion is valid from the start of the epoch (0000-01-01T00:00.00) until the `NotOnOrAfter` element. If the `NotOnOrAfter` element is unspecified the assertion is valid from the `NotBefore` element with no expiry. If neither element is specified the assertion is valid at any time.

In accordance with the XML Schemas Specification, all time instances are interpreted in Universal Coordinated Time unless they explicitly indicate a time zone.Implementations MUST NOT generate time instances that specify leap seconds.

For purposes of comparison, the time interval `NotBefore` to `NotOnOrAfter` begins at the earliest time instant compatible with the specification of `NotBefore` and *has ended* at the earliest time instant compatible with the specification of `NotOnOrAfter`.

For example if the time interval specified is *day*`T12:03:02` to *day*`T12:05:12` the times `12:03:02.00` and `12:05:11.9999` are within the time interval. The time `12:05:12.0000` is outside the time interval.

The following schema defines the `<ValidityInterval>` element:

```
<complexType name="ValidityInterval">
   <sequence>
      <element name="NotBefore" type="timeInstant"/>
      <element name="NotOnOrAfter" type="timeInstant"/>
   </sequence>
</complexType>
```

1.5 Conditions

Assertion Conditions are contained in the `<Conditions>` element. SAML applications MAY define additional elements using an extension schema. If an application encounters an element contained within a `<Conditions>` element that is not understood the status of the Condition MUST be considered Indeterminate.

The following conditions are defined:

Identifier	Type	Description
`Audiences`	`URI []`	Specifies the set of audiences to which the assertion is addressed.

The following schema defines the `<Conditions>` element:

```
<element name="Conditions">
   <complexType>
      <sequence>
         <element name="Audiences" >
           <complexType >
              <sequence>
                  <element name="string" type=“string"
                           minOccurs="0" maxOccurs="unbounded"/>
```

```
                </sequence>
             </complexType>
          </element>
       </sequence>
    </complexType>
</element>
```

1.5.1 Element <Audiences>

Assertions MAY be addressed to a specific audience. Although a party that is outside the audience specified is capable of drawing conclusions from an assertion, the issuer explicitly makes no representation as to accuracy or trustworthiness to such a party.

- Require users of an assertion to agree to specific terms (rule book, liability caps, relying party agreement)
- Prevent clients inadvertently relying on data that does not provide a sufficient warranty for a particular purpose
- Enable sale of per-transaction insurance services.

An audience is identified by a URI that identifies to a document that describes the terms and conditions of audience membership.

Each client is configured with a set of URIs that identify the audiences that the client is a member of, for example:

`http://cp.verisign.test/cps-2000`
Client accepts the VeriSign Certification Practices Statement

`http://rule.bizexchange.test/bizexchange_ruebook`
Client accepts the provisions of the *bizexchange* rule book.

An assertion MAY specify a set of audiences to which the assertion is addressed. If the set of audiences is the empty set there is no restriction and all audiences are addressed. Otherwise the client is not entitled to rely on the assertion unless it is addressed to one or more of the audiences that the client is a member of. For example:

`http://cp.verisign.test/cps-2000/part1`
Assertion is addressed to clients that accept the provisions of a specific part of the VeriSign CPS.

In this case the client accepts a superset of the audiences to which the assertion is addressed and may rely on the assertion.

The following schema defines the <Audiences> element:

```
<element name="Audiences" >
   <complexType >
      <sequence>
```

```
         <element name="string" type="string"
                  minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

1.6 Claims

The <Claims> element contains one or more SAML assertion claims. At present only one type of claim is defined, the <Authority> element. Additional types of claims may be defined in future revisions of the SAML specification or by means of an extension schema.

In each case if more than one assertion claim element is specified the validity of each claim is asserted jointly and severally, that is the semantics of a single assertion containing two claims are identical to the semantics of two separate assertions each of which contain one of the claims.

The following schema defines the <Claims> element:

```
<element name="Claims">
   <complexType>
      <sequence>
         <element name="Binding" type="so:Binding"
               minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
  </complexType>
</element>
```

1.6.1 Element <Binding>

The <Binding> element asserts a relationship between a specified subject and one or more objects.

The following schema defines the <Binding> element:

```
<element name="Binding">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name="Subject" type="s0:Subject"/>
         <element name="Object" type="s0:Object"/>
     </sequence>
  </complexType>
</element>
```

1.6.2 Element <Subject>

The <Subject> element specifies the subject of the binding. In every case the subject of a SAML assertion binding is a principal. A principal MAY be identified by name and/or by reference to authentication credentials.

Two forms of subject name are supported:

<CommonName>	An unstructured text string, for example “Alice Aardvark”.
<NameID>	A URI that specifies the principal by means of a machine readable atom.

In addition the principal MAY be specified by reference to authentication credentials by means of the <Authenticator> element.

The following schema defines the <Subject> element:

```
<element name="Subject">
   <complexType>
      <sequence>
         <element name=“CommonName”    type="string"/>
         <element name=“NameID”        type="uriReference"/>
         <element name=“Authenticator” type="s0:Authenticator"/>
      </sequence>
  </complexType>
</element>
```

1.6.3 Element <Authenticator>

The <Authenticator> element specifies a means of identifying the subject of the binding by means of their authentication credentials.

The authentication credentials MAY be specified either by means of the XML Digitial Signature <ds:KeyInfo> element or by means of the <Authdata> element. Applications SHOULD make use of the <ds:KeyInfo> element for credentials that it supports. Applications MAY use the <Authdata> element to specify other types of authentication credentials, including passwords.

The <Authenticator> element MAY specify one or more <Protocol> elements. If present the <Protocol> elements specify the authentication algorithms with which the authentication credentials MAY be used to obtain an acceptable authentication.

The following schema defines the <Authenticator> element:

```
<element name="Authenticator">
   <complexType>
      <sequence>
         <element name="Protocol" type="string"
               minOccurs="0" maxOccurs="unbounded/>
         <element name="Authdata" type="string"/>
         <element name="KeyInfo" type="ds:KeyInfo"/>
      </sequence>
  </complexType>
</element>
```

1.6.4 Element <Object>

The <Object> element specifies the object(s) of the binding and the relationship of the subject to the object. The object may be an attribute, a role or a resource. All objects are

referenced by means of URIs. A binding MAY specify multiple objects in which case the binding is asserted between the subject and each of the objects as if separate assertions had been issued for each case.

The relationship between the subject and the object is implicit for attributes and roles. In the case of an attribute the relationship is that the subject has the specified object. In the case of a role the relationship is that the subject has the specified role.

In the case where the object is a resource the relationship between the subject and object is the set of permissions assigned to the subject over the specified resource. Multiple permissions and multiple resources MAY be specified in the same binding in which case it is asserted that the subject is authorized to perform any of the specified actions on any of the specified resources.

These relationships are summarized in the following table:

Element	Relationship of Subject to Object
Attribute	Subject has the specified attribute `Has`
Role	Subject is assigned the specified role `Is`
Authorization	As specified by the `<Permission>` element(s). `Read`, `Write`, `Execute`, `Delete`, `Control`, or by URI

The following schema defines the `<Object>` element:

```
<element name="Object">
   <complexType>
      <sequence>
         <element name="Attribute" type="string"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <sequence>
         <element name="Role" type="string"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <sequence>
         <element name="Authorization" type="s0:Authorization"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

1.6.5 Element `<Authorization>`

The following schema defines the <Authorization> element:

```
<element name="Authorization">
   <complexType>
```

```
      <sequence>
         <element name="Resource" type="uriReference"
               minOccurs="0" maxOccurs="unbounded"/>
         <element name="Permission" type="string"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

1.6.6 Structured Entitlement

SAML applications MAY specify highly structured authority data in an `<Authority>` claim by means of an extension schema. The details of such schemas are outside the scope of SAML.

1.7 Advice

The Advice element is a general container for any additional information that does not affect the semantics or validity of the assertion itself.

1.7.1 Element `<Advice>`

The `<Advice>` element permits evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions.

The following schema defines the `<Advice>` element:

```
<element name="Advice">
   <complexType>
      <sequence>
         <element name="Assertion" type="Assertion"
               minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
  </complexType>
</element>
```

[An alternative use for the Advice element that is exploited in XTASS 1.0a is for specifying reissue information. This is not employed in SAML but is the reason for the change of name since the last version in case people were wondering.]

2 References

[Kerberos] *TBS*

[SAML-USE] *TBS*

[PKCS1] Kaliski, B., *PKCS #1: RSA Encryption Version 2.*0, RSA Laboratories, also IETF RFC 2437, October 1998.

[RFC-2104] Krawczyk, H., Bellare, M. and R. Canetti, *HMAC: Keyed Hashing for Message Authentication*, IETF RFC 2104, February 1997.

[SOAP] D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S Thatte, D. Winer. *Simple Object Access Protocol (SOAP) 1.1*, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP

[WSSL] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.0* September 25, 2000, http://msdn.microsoft.com/xml/general/wsdl.asp

[XACML] *TBS*

[XTASS] P. Hallam-Baker, *XML Trust Axiom Service Specification 1.0*, VeriSign Inc. January 2001. http://www.xmltrustcenter.org/

[XML-SIG] D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. *XML-Signature Syntax and Processing*, World Wide Web Consortium. http://www.w3.org/TR/xmldsig-core/

[XML-SIG-XSD] XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd

[XML-Enc] *XML Encryption Specification*, In development.

[XML-Schema1] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn. *XML Schema Part 1: Structures*, W3C Working Draft 22 September 2000, http://www.w3.org/TR/2000/WD-xmlschema-1-20000922/, latest draft at http://www.w3.org/TR/xmlschema-1/

[XML-Schema2] P. V. Biron, A. Malhotra, *XML Schema Part 2: Datatypes*; W3C Working Draft 22 September 2000, http://www.w3.org/TR/2000/WD-xmlschema-2-20000922/, latest draft at http://www.w3.org/TR/xmlschema-2/