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 0.8: June 14th 2001

Security Assertions Markup Language

Version 0.8

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

1.1 Namespaces

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/2001/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/"[PHB1]
```

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
        targetNamespace="http://www.oasis.org/tbs/1066-12-25/"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:s0="http://www.oasis.org/tbs/1066-12-25/"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        elementFormDefault="unqualified">
```

1.2 SAML Assertion

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

Authentication Assertion
An authentication assertion asserts that the issuer has authenticated the specified subject.

Attribute Assertion
An attribute assertion asserts that the specified subject has the specified attribute(s). Attributes may be specified by means of a URI or through an extension schema that defines structured attributes.

Decision Assertion
A decision assertion reports the result of the specified authorization request.

Authorization Assertion
An authorization assertion asserts that a subject has been granted specific permissions to access one or more resources.

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 `<SAMLAssertionPackage>`

The `<SAMLAssertionPackage>` element is specified by the following schema:

```
<element name="SAMLAssertionPackage" type="S0:SAMLAssertionPackageType">

<complexType name="SAMLAssertionPackageType">
   <!-- Basic Information -->
   <attribute name="Version"           type="string"/>
   <attribute name="AssertionID"       type="uriReference"/>
   <attribute name="Issuer"            type="string"/>
   <attribute name="IssueInstant"      type="timeInstant"/>
   <attribute name="NotBefore"         type="timeInstant"/>
   <attribute name="NotOnOrAfter"      type="timeInstant"/>

   <element name="Claims"     type="s0:Claims"      minOccurs="0"/>
   <element name="Conditions" type="s0:Conditions"  minOccurs="0"/>
   <element name="Advice"     type="s0:Advice"      minOccurs="0"/>
</complexType>
```

Six basic information attributes are defined; a protocol version identifier, a unique assertion identifier, an issuer identifier, the time instant of issue, the bounds of the validity interval.

1.2.1.1 Attribute `Version`

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

1.2.1.2 Attribute `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*. For the purposes of the SAML protocol 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 but applications MAY specify a URL as the assertion identifier that MAY resolve to the assertion.

1.2.1.3 Attribute `Issuer`

The `Issuer` attribute specifies the issuer of the assertion by means of a URI.

1.2.1.4 Attribute `IssueInstant`

The `IssueInstant` attribute specifies the time instant of issue in Universal Coordinated Time (UTC).

1.2.1.5 Attribute `NotBefore` and `NotOnOrAfter`

The `NotBefore` and `NotOnOrAfter` attributes specify limits on the validity of the assertion.

The NotBefore attribute specifies the time instant at which the validity interval begins. The NotOnOrAfter attribute specifies the time instant at which the validity interval has ended

The NotBefore and NotOnOrAfter attributes are optional. If the value is either omitted or equal to the start of the epoch it is unspecified. If the NotBefore attribute is unspecified the assertion is valid at any time before the time instant specified by the NotOnOrAfter attribute. If the NotOnOrAfter attribute is unspecified the assertion is valid from the time instant specified by the NotBefore attribute with no expiry. If neither attribute 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.

1.3 Claims

1.3.1 Element <Claims>

The <Claims> element contains one or more SAML assertion claims elements of type <AbstractClaimType>.

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="AbstractClaim" type="s0:AbstractClaimType"
               minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
  </complexType>
</element>

<complexType name="AbstractClaimType" abstract="true">
   <sequence>
      <element name="AssertionRef"     type="s0:AssertionRef"/>
      <!-- To add conditions on a per claim basis add :
      <element name="Conditions" type="s0:Conditions"  minOccurs="0"/>
          -->
   </sequence>
</complexType>
```

1.3.2 Element <AssertionRef>

The <AssertionRef> element specifies the assertion identifier of a prior assertion that has been used to generate the assertion in which the <AssertionRef> element occurs.

The primary purpose of `<AssertionRef>` elements is to permit auditing of SAML applications. As such an `<AssertionRef>` element is advisory only and does not mandate any specific action on the part of the application (such as tracking validity dependencies).

The following elements may include `<AssertionRef>` elements:

AbstractClaimType
Advises that the specified claim was derived from the specified assertion.

Subject
Advises that the Subject definition of the claim was derived from the specified assertion.

Advice
Advises that the referenced assertion was used to derive some unspecified portion of the assertion.

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

```
<complexType name="URIReferenceType">
   <attribute name="id" type="uriReference">
</complexType>

<element name="AssertionRef" type="s0:URIReferenceType">
```

1.3.3 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.

The following forms of subject name are supported:

Element	**Description**
`<CommonName>`	An unstructured text string, for example “Alice Aardvark”.
`<NameID>`	A URI that specifies the principal by means of a machine-readable identifier.
`<Authenticator>`	Specifies credentials and an authentication protocol by which the subject may be identified.
`<AssertionRef>`	Specifies that the contents of the `<Subject>` element were derived from the specified assertion.
`<AbstractSubject>`	Extension schema…

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="s0:URIReferenceType"/>
         <element name="Authenticator"    type="s0:Authenticator"/>
         <element name="AssertionRef"     type="s0:AssertionRef"/>
         <element name="AbstractSubject"
                                         type="s0:AbstractSubjectType"/>
     </sequence>
  </complexType>
</element>

<complexType name="AbstractSubjectType" abstract="true"/>
```

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

1.3.5 Element `<DecisionClaim>`

The `<DecisionClaim>` element asserts that the access permissions specified in the request identified by the corresponding RequestID were either permitted, denied or could not be determined.

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

```
<complexType name="DecisionClaim">
```

```
   <complexContent>
      <extension base="s0:AbstractClaimType">
         <attribute name="Decision" type="s0:DecsionType"/>
      </extension>
   </complexContent>
</complexType>

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

1.3.6 Element `<AuthenticationClaim>`

The `<AuthenticationClaim>` element asserts that the specified subject has been authenticated.[PHB2]

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

```
<complexType name="AuthenticationClaim">
   <complexContent>
      <extension base="s0:AbstractClaimType">
         <sequence>
            <element name="Subject"          type="s0:Subject"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

1.3.7 Element `<AttributeClaim>`

The `<AttributeClaim>` element asserts that a specified subject has the specified attribute(s) specified by a URI.

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

```
<complexType name="AttributeClaim">
   <complexContent>
      <extension base="s0:AbstractClaimType">
         <sequence>
            <element name="Subject" type="s0:Subject"/>
            <element name="AttributeID" type="s0:URIReferenceType"
                     minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

1.3.8 Element `<ExtendedAttributeClaim>`

The `<ExtendedAttributeClaim>` element asserts a relationship between the specified subject and a collection of attributes specified by means of an extension schema.

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

```
<complexType name="ExtendedAttributeClaim">
```

```
   <complexContent>
      <extension base="s0:AbstractClaimType">
         <sequence>
            <element name="Subject" type="s0:Subject"/>
            <element name="Attribute" type="s0:AbstractAttributeType"
                     minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<complexType name="AbstractAttributeType" abstract="true"/>
```

1.3.9 Element `<AuthorizationClaim>`

The `<AuthorizationClaim>` element asserts that the specified subject is authorized to perfom the specified operation(s)on the specified resource(s).

Defined permissions are Read, Write, Execute, Delete and Control. Additional permissions may be specified by URI through an `<ExtendedPermissions>` element.

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

```
<complexType name="AuthorizationClaim">
   <complexContent>
      <extension base="s0:AbstractClaimType">
         <sequence>
            <element name="Subject"          type="s0:Subject"/>
            <element name="Authorization"    type="s0:Authorization"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<element name="Authorization">
   <complexType>
      <sequence>
         <element name="Resource" type="uriReference"
               minOccurs="0" maxOccurs="unbounded"/>
         <element name="Permission" type="s0:PermissionType"
               minOccurs="0" maxOccurs="unbounded"/>
         <element name="ExtendedPermission" type="s0:URIReferenceType"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>

<simpleType name=PermissionType>
   <restriction base="string">
      <enumeration value="Read"/>
      <enumeration value="Write"/>
      <enumeration value="Execute"/>
      <enumeration value="Delete"/>
      <enumeration value="Control"/>
   </restriction>
</simpleType>
```

1.4 Conditions

1.4.1 Element `<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 schema defines the `<Conditions>` element:

```
<element name="Conditions">
   <complexType>
      <sequence>
         <element name="AbstractCondition"
               type="s0:AbstractConditionType"
               minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
  </complexType>
</element>

<complexType name="AbstractConditionType" abstract="true"/>
```

1.4.2 Element <AudienceRestrictionCondition>

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 <AudienceRestrictionCondition> element:

```
<complexType name="AudienceRestrictionCondition">
   <complexContent>
      <extension base="s0:AbstractConditionType">
         <sequence>
            <element name="Audience" type="s0:URIReferenceType"
                       minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

1.5 Advice

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

1.5.1 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="s0:Assertion"
               minOccurs="0" maxOccurs="unbounded"/>
         <element name="AssertionRef" type="s0:AssertionRef"
               minOccurs="0" maxOccurs="unbounded"/>
         <any namespace="##any" processContents="Skip">
     </sequence>
  </complexType>
</element>
```

1.6 SAML Protocol

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.

1.6.1 Element `<SAMLQuery>`

[PHB3]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 1.6.2.

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

```
<element name="SAMLQuery">
   <complexType>
      <sequence>
         <attribute name="RequestID" type="s0:AssertionID"/>
         <element name="QueryTemplate"
                     type="s0:SAMLAssertionPackageType"/>
         <element name="Respond"     type="s0:Respond"/>
     </sequence>
  </complexType>
</element>
```

1.6.1.1 Attribute **`<RequestID>`**

The `RequestID` attribute 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.

1.6.2 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**
`Static`	Specifies that the response may return any data element thus allowing the responder to return a static pre-signed assertion.
`DecisionClaim`	Specifies that the response may return an assertion that contains a `<DecisionClaim>` element
`AttributeClaim`	Specifies that the response may return an assertion that contains a

	`<AttributeClaim>` element
`ExtendedAttributeClaim`	Specifies that the response may return an assertion that contains a `<ExtendedAttributeClaim>` element
`AuthorizationClaim`	Specifies that the response may return an assertion that contains a `<AuthorizationClaim>` element
`AuthenticationClaim`	Specifies that the response may return an assertion that contains a `<DecisionClaim>` element
`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="Accept" type="string"
               minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

1.6.3 Element `<SAMLQueryResponse>`

The response to a `<SAMLQuery>` is a `<SAMLQueryResponse>` element. This returns the `RequestID` specified in the response and a `<SAMLAssertionPackage>` 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 -->
         <attributename="RequestID"   type="s0:uriReference"/>
         <element name="SAMLAssertionPackage"
                  type="s0:SAMLAssertionPackageType"/>
     </sequence>
  </complexType>
</element>

</schema>
```

1.7 Schema Extension

The SAML schema is designed to support extensibility by means of XML abstract types. Extension schemas should specify the purpose of extension elements by defining them as extensions of the appropriate abstract types.

The following abstract types are defined in the schema:

Abstract Type	Purpose
`AbstractClaimType`	Specify a new claim element.
`AbstractSubjectType`	Specify a new element for identifying the subject of a claim.
`AbstractAttributeType`	Specify structured attribute data.
`AbstractConditionType`	Specify a new condition element.

In addition the `<Advice>` element permits arbitrary elements to be included without type restriction.

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/

Page: 3
[PHB1] We have to align with the OASIS convention here.

Page: 9
[PHB2] I really think that this assertion is bogus as specified. An authentication assertion must have an object to make sense.

Page: 13
[PHB3] asking for it to be created.