Security Assertions Markup Language[pm1]

Core Assertion Architecture

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

Draft Version 0.12: July 27th 2001

Security Assertions Markup Language

Version 0.12

Table Of Contents

1 XML Assertion Syntax

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

Authentication Assertion
An authentication assertion is an assertion by the issuer that the subject was authenticated by a particular means at a particular time.

Authorization Decision Assertion
An authorization decision assertion is an assertion by the issuer that the request for access by the specified subject to the specified object has resulted in the specified decision on the basis of some optionally specified evidence.

Attribute Assertion
An attribute assertion asserts that the specified subject is associated with the specified attribute(s).

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 the version of the SAML assertion syntax, a unique identifier that serves as a name for the assertion, a unique identifier for the issuer and the time instant of issue.

The Asserted Statement
The statement that is asserted by the issuer of the assertion.

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]**.

[Class diagram to be inserted][PHB2]

1.1 Namespaces

In this document, certain namespace prefixes represent specific XML 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"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"[PHB3]
```

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

```
xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"[PHB4]
```

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.

The following schema defines the XML namespaces for the assertion schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
      xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
      xmlns="http://www.w3.org/2000/10/XMLSchema"
      elementFormDefault="unqualified">
   <import namespace=" http://www.w3.org/2000/09/xmldsig#"
         schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-schema-consensus-10.xsd</documentation>
   </annotation>
```

1.1.1 Basic Types

The types defined in this section define XML types that are considered part of the schema as a whole rather than a component of a particular element. This allows for greater consistency and avoids the need for extension schemas to redefine the same types.

1.1.1.1 Simple Type `IDType`

The `IDType` type is used for any data element that is an identifier for a specific data object that is opaque to the SAML application. In the SAML specification the `IDType` type is used declare and reference identifiers to assertions, requests and responses.

`IDType` identifiers MUST satisfy the following properties:

- Any party that assigns an `IDType` identifier to a data object MUST ensure that there is negligible probability that that party or any other party will assign the same identifier to a different data object.
- Where a data object specifies an `IDType` identifier that is to be an identifier for that object there MUST be exactly one such declaration.

The mechanism by which the application ensures that the identifier is unique is left to the implementation. In the case that a pseudorandom technique is employed the probability of two randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less than 2^{-160}.

An `IDType` identifier MAY or MAY NOT be resolvable. In the case that the identifier is resolvable (e.g. the identifier is a URL) the identifier MAY or MAY NOT resolve to the data object identified.

The `<AssertionID>` element is used to specify an identifier that references a SAML assertion.

The following schema specifies the `<AssertionID>` element and `IDType` type:

```
<element name="AssertionID" type="saml:IDType"/>
<simpleType name="IDType">
   <restriction base="string"/>
</simpleType>
```

1.1.1.2 Simple Type `DecisionType`

The `DecisionType` type reports the status of an authorization decision with respect to a specific resource.

`Permit`
The specified action is permitted.

`Deny`
The specified action is denied.

`Indeterminate`
No statement is made as to whether the specified action is permitted or denied.

The following schema specifies the `DecisionType` type:

```
<simpleType name="DecisionType">
   <restriction base="string">
```

```
            <enumeration value="Permit"/>
            <enumeration value="Deny"/>
            <enumeration value="Indeterminate"/>
        </restriction>
    </simpleType>
```

1.2 SAML Assertion

A SAML Assertion is specified by a single XML element whose type is derived from the abstract XML type `AssertionType`.[PHB5] The abstract `AssertionType` specifies the basic information that is common to all SAML assertions. Instances of SAML assertions have concrete types that are extensions of the base `AssertionType`.

1.2.1 Abstract Element `<Assertion>`

The element `<Assertion>` of abstract type `AssertionType` is used to specify a SAML assertion.

Following [XML-Schema] each instance of an `<Assertion>` element MUST specify the specific concrete type using the `xsiType` attribute:

```
    <Assertion xsiType=AttributeAssertionType>
```

The following schema specifies the `<Assertion>` element and `AssertionType` abstract type:

```
    <element name="Assertion" type="saml:AssertionType"/>
    <complexType name="AssertionType" abstract="true">
        <sequence>
            <element name="Conditions" type="saml:ConditionsType" minOccurs="0"/>
            <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
        </sequence>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="AssertionID" type="saml:IDType" use="required"/>
        <attribute name="Issuer" type="string" use="required"/>
        <attribute name="IssueInstant" type="timeInstant" use="required"/>
    </complexType>

```

The following basic information attributes are defined; a protocol version identifier, a unique assertion identifier, an issuer identifier and the time instant of issue.

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`

The `AssertionID` attribute specifies the assertion identifier.

1.2.1.3 Attribute `Issuer`

The `Issuer` attribute specifies the issuer of the assertion by means of a string[PHB6].

1.2.1.4 Attribute `IssueInstant`

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

1.2.2 Element `<AssertionSpecifier>`

The `<AssertionSpecifier>` element specifies an assertion either by reference or by value. An assertion is specified by reference to the value of its AssertionID attribute. An assertion is specified by value by including the entire assertion.

The following schema specifies the `<AssertionSpecifier>` element:

```
<element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
<complexType name="AssertionSpecifierType">
   <choice>
      <element ref="saml:AssertionID" />
      <element ref="saml:Assertion"/>
   </choice>
</complexType>
```

1.3 Subject Assertion

A Subject Assertion is an assertion by the issuer that concerns a SAML subject specified by a `<Subject>` element.

1.3.1 Abstract Type `SubjectAssertionType`

The `SubjectAssertionType` type is an abstract type that extends the `AssertionType` to include a `<Subject>` element.

The following schema defines the `SubjectAssertionType` abstract type:

```
<complexType name="SubjectAssertionType" abstract="true">
   <complexContent>
      <extension base="saml:AssertionType">
         <sequence>
            <element ref="saml:Subject"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

1.3.2 Element `<Subject>`

The `<Subject>` element specifies a party by one of the following means:

- A name.
- By information that allows the party to be authenticated.
- By reference to another assertion or by containment of another assertion.

If a `<Subject>` element contains more than one subject specification the issuer is asserting that all the subject specifications present specify the same subject. For example if both a `<NameIdentifier>` and a `<Authenticator>` element are present the issuer is asserting that the authentication data authenticates the party with the specified name.

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

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
   <choice maxOccurs="unbounded">
      <element ref="saml:NameIdentifier"
            minOccurs="0" maxOccurs="unbounded"/>
      <element ref="saml:Authenticator"
            minOccurs="0" maxOccurs="unbounded"/>
      <element ref="saml:AssertionSpecifier"
            minOccurs="0" maxOccurs="unbounded"/>
   </choice>
</complexType>
```

1.3.2.1 Element `<Authenticator>`

The <Authenticator> element specifies a subject by specifying data that authenticates the subject.

`<Protocol>` `[Required]`
Each `<Protocol>` element specifies a URI that identify a protocol that may be used to authenticate the subject.

`<AuthData>` `[Optional]`
Each `<AuthData>` element specifies additional authentication information used by a specific authentication protocol.

`<ds:KeyInfo>` `[Optional]`
An XML Signature `<ds:KeyInfo>` element that specifies a cryptographic key held by the subject.

URIs identifying common authentication protocols are specified in Section 4 .

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

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

1.3.2.2 Element `<NameIdentifier>`

The NameIdentifier type specifies a subject by a combination of a name and a security domain.

The interpretation of the security domain and the name are left to individual implementations.[PHB7]

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

```
   <element name="NameIdentifier" type="saml:NameIdentifierType"/>
   <complexType name="NameIdentifierType">
      <sequence>
         <element name="SecurityDomain" type="string"/>
         <element name="Name" type="string"/>
      </sequence>
   </complexType>
```

1.4 Authentication Assertion

An authentication assertion is an assertion by the issuer that the subject was authenticated by a particular means at a particular time.[PHB8]

1.4.1 Assertion Type `AuthenticationAssertionType`

The `AuthenticationAssertionType` extends the `SubjectAssertionType` with the addition of the following elements:

`<AuthenticationCode>` [Required]
The `<AuthenticationCode>` element specifies the type of Authentication that took place.

`<AuthenticationInstant>` [Required]
The `<AuthenticationInstant>` element specifies the time at which the authentication took place.

`<AuthenticationLocale>` [Optional]
The `<AuthenticationLocale>` element specifies the DNS domain name and IP address for the system entity that performed the authentication.

The following schema defines the `<AuthenticationAssertion>` assertion type:

```
   <complexType name="AuthenticationAssertionType">
      <complexContent>
         <extension base="saml:SubjectAssertionType">
            <sequence>
               <element ref="saml:AuthenticationCode"/>
               <element name="AuthenticationInstant" type="timeInstant"/>
               <element name="AuthLocale"
                     type="saml:AuthLocaleType" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
```

1.4.1.1 Element `<AuthenticationCode>`

The `<AuthenticationCode>` element specifies the type of Authentication that took place.[PHB9]

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

```
   <element name="AuthenticationCode" type="saml:AuthenticationCodeType"/>
   <simpleType name="AuthenticationCodeType">
      <restriction base="string"/>
   </simpleType>
```

1.4.1.2 Type `AuthLocale`

The `<AuthenticationCode>` element specifies the DNS domain name and IP address for the system entity that performed the authentication.

Note: This element is entirely advisory, since both these fields are quite easily "spoofed" but current practice appears to require its inclusion.

The following schema defines the `<AuthLocale>` type:

```
   <complexType name="AuthLocaleType">
      <sequence>
         <element name="IP" type="string" minOccurs="0"/>
         <element name="DNS_Domain" type="string" minOccurs="0"/>
      </sequence>
   </complexType>
```

1.5 Authorization Decision Assertion

An authorization decision assertion is an assertion by the issuer that the request for access by the specified subject to the specified object has resulted in the specified decision on the basis of some optionally specified evidence.

1.5.1 Assertion Type `AuthorizationDecisionAssertionType`

The `AuthorizationDecisionAssertionType` extends the `SubjectAssertionType` with the addition of the following elements:

`<Object>` [Required]
The `<Object>` element specifies a set of actions on a specified resource

`<Answer>` [Required]
The `<Answer>` element specifies the decision with respect to the specified object.

`<Evidence>` [Optional]
The `<Evidence>` element specifies a set of assertions that the issuer relied upon in making the decision.

The following schema defines the `<AuthorizationDecisionAssertionType >` type:

```
   <complexType name="AuthorizationDecisionAssertionType">
      <complexContent>
         <extension base="saml:SubjectAssertionType">
            <sequence>
               <element ref="saml:Object"/>
```

```
                <element name="Answer" type="saml:DecisionType"/>
                <element name="saml:Evidence"
                      minOccurs="0" maxOccurs="unbounded"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
```

1.5.1.1 Element `<Object>`

The `<Object>` element specifies an authorization object that consists of a set of actions on a specified resource. The `<Object>` element contains the following elements:

`<Resource>` [Required]
The `<Resource>` element specifies the resource by means of a URI.

`<Namespace>` [Optional]
The `<Namespace>` element specifies the namespace in which the specified action elements are to be interpreted.

`<Action>` [One or more]
The `<Action>` element specifies the set of actions on the specified resource.

If the `<Namespace>` element is not specified the namespace specified in section 4.2.1 is specified by default.

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

```
    <element name="Object" type="saml:ObjectType"/>
    <complexType name="ObjectType">
       <sequence>
          <element name="Resource" type="xsd:uriReference"/>
          <element name="Namespace" type="uriReference" minOccurs="0"/>
          <element name="Action" type="string" maxOccurs="unbounded"/>
       </sequence>
    </complexType>
```

1.5.1.2 Element `<Evidence>`

The `<Evidence>` element specifies a set of assertions that the issuer relied upon in issuing the assertion.

The statement of an assertion as evidence MAY affect the reliance agreement between the client and service. For example in the case that the client presented an assertion to the service in a request the service MAY use that assertion as evidence in making its response without endorsing the assertion as valid either to the client or any third party.

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

```
    <element name="Evidence" type="saml:AssertionSpecifierType"/>
```

1.6 Attribute Assertion

An attribute assertion asserts that the specified subject is associated with the specified attribute(s)

1.6.1 Assertion Type `AttributeAssertionType`

The `AttributeAssertionType` extends the `SubjectAssertionType` with the addition of the following element:

`<Attribute>` [One or More]
The `<Attribute>` element specifies an attribute of the assertion subject.

The following schema defines the `AttributeAssertionType` assertion type:

```
<complexType name="AttributeAssertionType">
   <complexContent>
      <extension base="saml:SubjectAssertionType">
         <sequence>
            <element ref="saml:Attribute" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

1.6.1.1 Element `<Attribute>`

The `<Attribute>` element specifies an attribute of the assertion subject. An attribute is identified by a name that is interpreted within a particular namespace. The `<Attribute>` element contains the following elements:

`<AttributeNamespace>` [Optional]
The `<AttributeNamespace>` element specifies the namespace in which the `<AttributeName>` elements are interpreted.

`<AttributeName>` [Required]
The `<AttributeName>` element specifies the name of the attribute.

`<AttributeValue>` [Any number]
Each `<AttributeValue>` element specifies a value of the attribute.

If no `<AttributeNamespace>` element is specified the interpretation of `<AttributeName>` elements is left to the implementation.

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

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
   <sequence>
      <element name="AttributeName" type="string"/>
      <element name="AttributeNamespace"
            type="uriReference" minOccurs="0"/>
      <element name="AttributeValue" type="saml:AttributeValueType"
            minOccurs="0" maxOccurs="unbounded"/>
```

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

1.6.1.2 Type `AttributeValueType`

An `<AttributeValue>` element is of type `AttributeValueType`. The `AttributeValueType` allows the inclusion of any element in any namespace.[PHB10]

The following schema defines the `AttributeValueType` type:

```
    <complexType name="AttributeValueType">
        <sequence>
            <any namespace="##any" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
```

1.7 Conditions

The validity of an assertion MAY be subject to a set of conditions. Each condition evaluates to a value that is `Valid`, `Invalid` or `Indeterminate`. The validity status of an assertion is the conjunction of the validity of each of the assertion conditions as follows:

If any condition evaluates to `Invalid`.
The assertion status is `Invalid`

If no condition evaluates to `Invalid` and one or more conditions evaluate to `Indeterminate`.
The assertion status is `Indeterminate`

If no conditions are specified or all the specified assertions evaluate to `Valid`.
The assertion status is `Valid`

1.7.1 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:

```
    <complexType name="ConditionsType">
        <sequence>
            <element name="Condition" type="saml:AbstractConditionType"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="NotBefore" type="timeInstant" use="optional"/>
        <attribute name="NotOnOrAfter" type="timeInstant" use="optional"/>
    </complexType>

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

1.7.1.1 Attributes `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.

1.7.2 Condition Type `AudienceRestrictionConditionType`

- A `<Condition>` element of type `AudienceRestrictionConditionType` states that the assertion is addressed to one or more specific audience(s). Although a party that is outside the audience(s) specified is capable of drawing conclusions from an assertion, the issuer explicitly makes no representation as to accuracy or trustworthiness to such a party.

An audience is identified by a URI namespace. The URI MAY identify a document that describes the terms and conditions of audience membership.

The following schema defines the `AudienceRestrictionConditionType` condition type:

```
<complexType name="AudienceRestrictionConditionType">
    <complexContent>
        <extension base="saml:AbstractConditionType">
            <sequence>
                <element name="Audience" type="xsd:uriReference"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

1.8 Advice

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

1.8.1 Element `<Advice>`

The `<Advice>` element permits additional information to be included in an assertion that MAY be ignored by applications without affecting either the assertion semantics or validity. Advice elements MAY be specified in an extension schema. The advice element MAY be used to:

- Include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions.
- State a proof of the assertion claims.
- Specify the timing and distribution points for updates to the assertion.

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

```
    <element name="Advice" type="saml:AdviceType"/>
    <complexType name="AdviceType">
        <sequence>
            <any namespace="##any" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</schema>
```

1.9 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
`AssertionType`	Define new SAML Assertion
`SubjectAssertionType`	Define new SAML Assertion that takes a single SAML Subject as the subject.
`AbstractConditionType`	Define a new SAML Condition

In addition the `<Advice>` element permits arbitrary elements to be included without type restriction. An application is not required to understand or process any information contained within an `<Advice>` element however.[PHB11]

2 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 `<SAMLRequest>` and `<SAMLResponse>` elements described in this section. A `<SAMLRequest>` from the client is followed by a `<SAMLResponse>` from the service Figure 1.

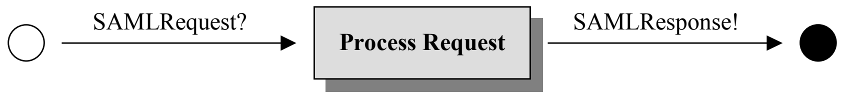

Figure 1: SAML Request/Response Protocol

2.1 Namespaces

The namespaces of the protocol schema are the same as those of the assertion schema defined in section 1.1 with the addition of the namespace for the protocol schema itself:.

```
xmlns:samlp=http://www.oasis.org/tbs/1066-12-25/protocol/
```

The following schema defines the XML namespaces for the assertion schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/protocol/"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
      xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
      xmlns:samlp="http://www.oasis.org/tbs/1066-12-25/protocol/"
      xmlns="http://www.w3.org/2000/10/XMLSchema"
      elementFormDefault="unqualified">
   <import namespace="http://www.oasis.org/tbs/1066-12-25/"
         schemaLocation="draft-schema-assertion-10.xsd"/>
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
         schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-schema-protocol-10.xsd</documentation>
   </annotation>
```

2.1.1 Basic Types

The types defined in this section define XML types that are considered part of the schema as a whole rather than a component of a particular element. This allows for greater consistency and avoids the need for extension schemas to redefine the same types.

2.1.1.1 Simple Type `CompletenessSpecifierType`

The `CompletenessSpecifierType` type is used in a request to specify how a service should respond in cases where a client makes a request and the client is not authorized to receive part of the response. The `CompletenessSpecifierType` type defines two possible values `"Any"` and `"All"`.

If `Any` is specified:
The response contains the parts of the response that the client is authorized to receive.

If `All` is specified:
The response is empty.

The following schema specifies the `<CompletenessSpecifierType>` type:

```
<simpleType name="CompletenessSpecifierType">
    <restriction base="string">
        <enumeration value="Any"/>
        <enumeration value="All"/>
    </restriction>
</simpleType>
```

2.1.1.2 Simple Type `StatusCodeType`

The type `StatusCodeType` in a response specifies the status of the request. Four status values are defined:

`Success`
The request succeeded.

`Failure`
The request could not be performed by the service.

`Error`
An error in the request prevented the service from processing it.

`Unknown`
The request failed for unknown reasons[PHB12]

The following schema specifies the `<StatusCodeType>` type:

```
<simpleType name="StatusCodeType">
    <restriction base="string">
        <enumeration value="Success"/>
        <enumeration value="Failure"/>
        <enumeration value="Error"/>
        <enumeration value="Unknown"/>
    </restriction>
</simpleType>
```

2.2 Request

2.2.1 Abstract Type `SAMLAbstractRequestType`

[PHB13]All SAML requests are extensions of the `SAMLAbstractRequestType` abstract type. The `SAMLAbstractRequestType` requires that all SAML requests specify the version number of the SAML protocol and a request identifier.

The following schema defines the SAMLAbstractRequestType abstract type:

```
    <complexType name="SAMLAbstractRequestType" abstract="true">
        <attribute name="RequestID" type="saml:IDType" use="required"/>
        <attribute name="Version" type="string" use="required"/>
    </complexType>
```

2.2.1.1 Attribute RequestID

The RequestID attribute defines a unique identifier for the assertion request. The RequestID element in a request MUST match the InResponseTo element in the corresponding response.

2.2.1.2 Attribute Version

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

2.2.2 Element <SAMLRequest>

The <SAMLRequest> element specifies a SAML request. This may contain either a query or a request for a specific assertion identified by AssertionID.

The following schema defines the <SAMLRequest> element:

```
    <element name="SAMLRequest" type="samlp:SAMLRequestType"/>
    <complexType name="SAMLRequestType">
        <complexContent>
            <extension base="samlp:SAMLAbstractRequestType">
                <choice>
                    <element name="Query" type="samlp:SAMLQueryType"/>
                    <element ref="saml:AssertionID" maxOccurs="unbounded"/>
                </choice>
            </extension>
        </complexContent>
</complexType>
```

2.2.3 Abstract Type SAMLQueryType

[PHB14]The SAMLQueryType abstract type is the base type from which all SAML request query elements are derived. The abstract type contains no elements or attributes.

The following schema defines the SAMLQueryType abstract type:

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

2.2.4 Abstract Type SubjectQueryType

[PHB15]The SubjectQueryType type extends the SAMLQuery type to specify a query with a specific subject as its principal.

The following schema defines the SubjectQueryType abstract type:

```
    <complexType name="SubjectQueryType" abstract="true">
        <complexContent>
            <extension base="samlp:SAMLQueryType">
            <sequence>
                <element ref="saml:Subject"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

2.3 Authentication Query

The AuthenticationQueryType makes the query "What authentication assertions are available for this Subject?"

The response will be in the form of an Authentication assertion.

2.3.1 Subject Query Type `AuthenticationQueryType`

An `AuthenticationQuery` contains all the elements and attributes of a `SubjectQuery` and extends them as follows:

`<AuthenticationCode>` [Optional]
The `<AuthenticationCode>` element if present can be used to as a filter for possible responses. This supports the query "What authentication assertions do you have for this Subject with the following `AuthenticationCode`?"

A SAML processor will return a certain number of Authentication Assertions in response to a `SAMLQuery` of type `AuthenticationQueryType`. The `<Subject>` and `<AuthenticationCode>` of the returned assertions MUST be identical to the subject (and optional `<AuthenticationCode>`) fields of the `SAMLQuery`. There is no implication that all such assertions must be returned.

The following schema defines the `AuthenticationQueryType` type:

```
<complexType name="AuthenticationQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryType">
            <sequence>
                <element ref="saml:AuthenticationCode" minOccurs="0"/>
                <!--do we want more than one of these?-->
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

2.4 Attribute Query

An Attribute Query makes the query "Return all of the attributes for this Subject (that I am allowed to see)?"

The response will be in the form of an Attribute Assertion.

2.4.1 Subject Query Type `SAMLAttributeQueryType`

An `SAMLAttributeQueryType` contains all the elements and attributes of a `SubjectQueryType` and extends them as follows:

`<Attribute>` [Any number]
Each `<Attribute>` element specifies an attribute that is to be returned. If no attributes are specified the scope of the query is implicit.

`<CompletenessSpecifier>` [Required]
The `<CompletenessSpecifier>` element specifies the desired behavior in the case that access to some of the requested attributes is not authorized using the `CompletenessSpecifier`.

The following schema defines the `<SAMLAttributeQueryType>` type:

```
<complexType name="AttributeQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryType">
            <sequence>
                <element ref="saml:Attribute"
                        minOccurs="0" maxOccurs="unbounded"/>
                <element name="CompletenessSpecifier"
                        type="samlp:CompletenessSpecifierType" default="All"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

2.5 Authorization Query

An Authorization Query makes the query : “Should action(s) Y on resource Z be allowed for subject S given evidence E?”

The answer comes in the form of an Authorization Decision assertion. The action(s) and resource are optionally namespace-scoped..

2.5.1 Subject Query Type `AuthorizationQueryType`

An AuthorizationQuery contains all the elements and attributes of a `SubjectQueryType` and extends them as follows:

`<Object>` [Required]
The `<Object>` element specifies the resource and action(s) for which authorization is requested.

`<Evidence>` [Any number]
Each `<Evidence>` element specifies an assertion that the service may rely upon in making its response.

The following schema defines the `AuthorizationQueryType` type:

```
<element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
<complexType name="AuthorizationQueryType">
   <complexContent>
      <extension base="samlp:SubjectQueryType">
         <sequence>
            <element ref="saml:Evidence"
                  minOccurs="0" maxOccurs="unbounded"/>
            <element ref="saml:Object"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.6 Response

2.6.1 Abstract Type `SAMLAbstractResponseType`

The response to a SAMLRequest is of a type that extends the `SAMLAbstractResponseType` abstract type. The `SAMLAbstractResponseType` specifies information that is common to all SAML responses, the SAML protocol version, the identifier of the response and the identifier of the request that is responded to.

The following schema defines the `SAMLAbstractResponseType` abstract type:

```
<complexType name="SAMLAbstractResponseType" abstract="true">
   <attribute name="ResponseID" type="saml:IDType" use="required"/>
   <attribute name="InResponseTo" type="saml:IDType" use="required"/>
   <attribute name="Version" type="string" use="required"/>
</complexType>
```

2.6.1.1 Attribute `ResponseID`

The `ResponseID` attribute specifies an identifier of type `IDType` for the response.

2.6.1.2 Attribute `InResponseTo`

The `ResponseID` attribute specifies the identifier of type `IDType` specified in the `RequestID` attribute in the SAML Request to which it is a response.

2.6.1.3 Attribute `Version`

Each response MUST specify the SAML version identifier. The identifier for this version of SAML is the string "`1.0`".[PHB16]

2.6.2 Element `<SAMLResponse>`

The `<SAMLResponse>` element is extends the `SAMLAbstractResponseType` and specifies the status of the corresponding SAML Request and a list of zero or more assertions that answer the request.

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

```
    <element name="SAMLResponse" type="samlp:SAMLResponseType"/>
    <complexType name="SAMLResponseType">
        <complexContent>
            <extension base="samlp:SAMLAbstractResponseType">
                <sequence>
                    <element ref="saml:Assertion"
                            minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
                <attribute name="StatusCode" type="samlp:StatusCodeType"
                            use="required"/>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

2.7 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:[PHB17]

Abstract Type	Purpose
`SAMLAbstractRequestType`	Specify a new SAML request other than a query.
`SAMLQueryType`	Specify a new SAML request that is a query.
`SubjectQueryType`	Specify a new SAML request that is a query concerning a single subject.
`SAMLAbstractResponseType`	Specify a new SAML response.

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

3 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/

4 Identifiers

4.1 Authentication Protocol Identifiers

4.1.1 SAML Artifact

4.1.2 Assertion Bearer

4.1.3 User Name and Password (Pass-through)

4.1.4 User Name and Password (One-Way-Function SHA-1)

4.1.5 Kerberos

4.1.6 SSL/TLS Certificate Based Client Authentication

4.2 Action Identifiers

4.2.1 Read/Write/Execute/Delete/Control

4.2.2 Read/Write/Execute/Delete/Control with Negation

4.2.3 Get/Head/Put/Post

4.2.4 UNIX file Permissions

5 Appendix

5.1 Assertion Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
       xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
       xmlns="http://www.w3.org/2000/10/XMLSchema"
       elementFormDefault="unqualified">
   <import namespace=" http://www.w3.org/2000/09/xmldsig#"
          schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-schema-consensus-10.xsd</documentation>
   </annotation>

   <element name="AssertionID" type="saml:IDType"/>
   <simpleType name="IDType">
      <restriction base="string"/>
   </simpleType>
   <simpleType name="DecisionType">
      <restriction base="string">
         <enumeration value="Permit"/>
         <enumeration value="Deny"/>
         <enumeration value="Indeterminate"/>
      </restriction>
   </simpleType>

   <element name="Assertion" type="saml:AssertionType" />
   <complexType name="AssertionType" abstract="true">
      <sequence>
         <element name="Conditions" type="saml:ConditionsType" minOccurs="0"/>
         <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
      </sequence>
      <attribute name="Version" type="string" use="required"/>
      <attribute name="AssertionID" type="saml:IDType" use="required"/>
      <attribute name="Issuer" type="string" use="required"/>
      <attribute name="IssueInstant" type="timeInstant" use="required"/>
   </complexType>

   <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
   <complexType name="AssertionSpecifierType">
      <choice>
         <element ref="saml:AssertionID" />
         <element ref="saml:Assertion"/>
      </choice>
   </complexType>

   <complexType name="SubjectAssertionType" abstract="true">
      <complexContent>
         <extension base="saml:AssertionType">
            <sequence>
               <element ref="saml:Subject"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>

   <element name="Subject" type="saml:SubjectType"/>
   <complexType name="SubjectType">
```

```
        <choice maxOccurs="unbounded">
            <element ref="saml:NameIdentifier"
                    minOccurs="0" maxOccurs="unbounded"/>
            <element ref="saml:Authenticator"
                    minOccurs="0" maxOccurs="unbounded"/>
            <element ref="saml:AssertionSpecifier"
                    minOccurs="0" maxOccurs="unbounded"/>
        </choice>
    </complexType>

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

    <element name="NameIdentifier" type="saml:NameIdentifierType"/>
    <complexType name="NameIdentifierType">
        <sequence>
            <element name="SecurityDomain" type="string"/>
            <element name="Name" type="string"/>
        </sequence>
    </complexType>

    <complexType name="AuthenticationAssertionType">
        <complexContent>
            <extension base="saml:SubjectAssertionType">
                <sequence>
                    <element ref="saml:AuthenticationCode"/>
                    <element name="AuthenticationInstant" type="timeInstant"/>
                    <element name="AuthLocale"
                            type="saml:AuthLocaleType" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="AuthenticationCode" type="saml:AuthenticationCodeType"/>
    <simpleType name="AuthenticationCodeType">
        <restriction base="string"/>
    </simpleType>

    <complexType name="AuthLocaleType">
        <sequence>
            <element name="IP" type="string" minOccurs="0"/>
            <element name="DNS_Domain" type="string" minOccurs="0"/>
        </sequence>
    </complexType>

    <complexType name="AuthorizationDecisionAssertionType">
        <complexContent>
            <extension base="saml:SubjectAssertionType">
                <sequence>
                    <element ref="saml:Object"/>
                    <element name="Answer" type="saml:DecisionType"/>
                    <element name="saml:Evidence"
                            minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="Object" type="saml:ObjectType"/>
    <complexType name="ObjectType">
        <sequence>
            <element name="Resource" type="xsd:uriReference"/>
            <element name="Namespace" type="uriReference" minOccurs="0"/>
            <element name="Action" type="string" maxOccurs="unbounded"/>
        </sequence>
    </complexType>

    <element name="Evidence" type="saml:AssertionSpecifierType"/>

    <complexType name="AttributeAssertionType">
        <complexContent>
            <extension base="saml:SubjectAssertionType">
                <sequence>
                    <element ref="saml:Attribute" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="Attribute" type="saml:AttributeType"/>
    <complexType name="AttributeType">
        <sequence>
            <element name="AttributeName" type="string"/>
            <element name="AttributeNamespace"
                    type="uriReference" minOccurs="0"/>
            <element name="AttributeValue" type="saml:AttributeValueType"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>

    <complexType name="AttributeValueType">
        <sequence>
            <any namespace="##any" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>

    <element name="Conditions" type="saml:ConditionsType"/>
    <complexType name="ConditionsType">
        <sequence>
            <element name="Condition" type="saml:AbstractConditionType"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="NotBefore" type="timeInstant" use="optional"/>
        <attribute name="NotOnOrAfter" type="timeInstant" use="optional"/>
    </complexType>

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

    <complexType name="AudienceRestrictionConditionType">
        <complexContent>
            <extension base="saml:AbstractConditionType">
                <sequence>
                    <element name="Audience" type="xsd:uriReference"
                            minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="Advice" type="saml:AdviceType"/>
    <complexType name="AdviceType">
        <sequence>
            <any namespace="##any" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
</schema>
```

5.2 Protocol Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.oasis.org/tbs/1066-12-25/protocol/"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
        xmlns:saml="http://www.oasis.org/tbs/1066-12-25/"
        xmlns:samlp="http://www.oasis.org/tbs/1066-12-25/protocol/"
        xmlns="http://www.w3.org/2000/10/XMLSchema"
        elementFormDefault="unqualified">
    <import namespace="http://www.oasis.org/tbs/1066-12-25/"
            schemaLocation="draft-schema-assertion-10.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
            schemaLocation="xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>draft-schema-protocol-10.xsd</documentation>
    </annotation>

    <simpleType name="CompletenessSpecifierType">
        <restriction base="string">
            <enumeration value="Any"/>
            <enumeration value="All"/>
        </restriction>
    </simpleType>

    <simpleType name="StatusCodeType">
        <restriction base="string">
            <enumeration value="Success"/>
            <enumeration value="Failure"/>
            <enumeration value="Error"/>
            <enumeration value="Unknown"/>
        </restriction>
    </simpleType>

    <complexType name="SAMLAbstractRequestType" abstract="true">
        <attribute name="RequestID" type="saml:IDType" use="required"/>
        <attribute name="Version" type="string" use="required"/>
    </complexType>

    <element name="SAMLRequest" type="samlp:SAMLRequestType"/>
    <complexType name="SAMLRequestType">
        <complexContent>
            <extension base="samlp:SAMLAbstractRequestType">
                <choice>
                    <element name="Query" type="samlp:SAMLQueryType"/>
                    <element ref="saml:AssertionID" maxOccurs="unbounded"/>
                </choice>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="SAMLQueryType" abstract="true"/>

    <complexType name="SubjectQueryType" abstract="true">
        <complexContent>
            <extension base="samlp:SAMLQueryType">
                <sequence>
                    <element ref="saml:Subject"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="AuthenticationQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryType">
                <sequence>
                    <element ref="saml:AuthenticationCode" minOccurs="0"/>
                    <!--do we want more than one of these?-->
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="AttributeQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryType">
                <sequence>
                    <element ref="saml:Attribute"
                            minOccurs="0" maxOccurs="unbounded"/>
                    <element name="CompletenessSpecifier"
                            type="samlp:CompletenessSpecifierType" default="All"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
    <complexType name="AuthorizationQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryType">
                <sequence>
                    <element ref="saml:Evidence"
                            minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="saml:Object"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="SAMLAbstractResponseType" abstract="true">
        <attribute name="ResponseID" type="saml:IDType" use="required"/>
        <attribute name="InResponseTo" type="saml:IDType" use="required"/>
        <attribute name="Version" type="string" use="required"/>
    </complexType>

    <element name="SAMLResponse" type="samlp:SAMLResponseType"/>
    <complexType name="SAMLResponseType">
        <complexContent>
            <extension base="samlp:SAMLAbstractResponseType">
                <sequence>
                    <element ref="saml:Assertion"
                            minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
                <attribute name="StatusCode" type="samlp:StatusCodeType"
                           use="required"/>
            </extension>
        </complexContent>
    </complexType>

</schema>
```

Page: 1
[pm1]

Page: 5
[PHB2]	Insert the class diagram here

Page: 5
[PHB3]This schema is actually to the previous version since that is what Spy 3.5 accepts.

Page: 5
[PHB4]	We have to align with the OASIS convention here.

Page: 7
[PHB5]Here we need to redo the diagram to express the inheritance mechanism we choose in the end.

Page: 7
[PHB6]	Need some better text here n'est pas?

Page: 10
[PHB7]Yikes! maybe we should put an IDType in here so that the security domain is at least unique!

Page: 10
[PHB8]But by whom? It may not have been the issuer of the assertion!

Page: 10
[PHB9]	Where are these codes defined? Do we reuse the protocol elements of section 4, I think we should but in that case the schema should specify a URI.

Page: 14
[PHB10]But the current definition does not give a ground case, how about a string?

Page: 16
[PHB11]Add in here the substitution group issue.

Page: 18
[PHB12]Need to have text for these, how exactly does failure differ from error?

Page: 18
[PHB13]	asking for it to be created.

Page: 19
[PHB14]	asking for it to be created.

Page: 19
[PHB15]	asking for it to be created.

Page: 22
[PHB16]	Other options here include specifying the highest version number that the server can supply. I suspect however that in the web services context that is unnecessary since it will be taken care of by WSDL.

Page: 23
[PHB17]If we use substitution groups add in the text to specify the requirement