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*
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 0.19: September 28th 2001

Security Assertions Markup Language

Version 0.19

Table Of Contents

1 XML Assertion Syntax

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

Authentication Assertion
An authentication assertion contains a statement made by the issuer that asserts the subject was authenticated by a particular means at a particular time.

Authorization Decision Assertion
An authorization decision assertion contains a statement made by the issuer that asserts 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 contains a statement made by the issuer that asserts 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(s)
The statement(s) that are 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 statements. Possible additional applications of the assertion package format include management of embedded trust roots **[XTAML]** and authorization policy information **[XACML]**.

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"
```

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

```
xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-19.xsd"
```

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"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
    targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-19.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-19.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>draft-sstc-schema-assertion-19.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` and Element `<AssertionID>`

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 `AssertionAbstractType`. The abstract `AssertionAbstractType` specifies the basic information that is common to all SAML assertions. Instances of SAML assertions have concrete types that are extensions of the base `AssertionAbstractType`.

1.2.1 Element `<Assertion>`

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

`MajorVersion` [Required]
: Each assertion MUST specify the SAML major version identifier. The identifier for this version of SAML is 1.

`MinorVersion` [Required]
: Each assertion MUST specify the SAML minor version identifier. The identifier for this version of SAML is 0.

`AssertionID` [Required]
: The `AssertionID` attribute specifies the assertion identifier.

`Issuer` [Required]
: The `Issuer` attribute specifies the issuer of the assertion by means of a string[PHB1].

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

`<Conditions>` [Optional]
: The `<Conditions>` element specifies conditions that affect the validity of the asserted statement.

`<Advice>` [Optional]
: The `<Advice>` element specifies additional information related to the assertion that may assist processing in certain situations but which can be ignored by applications that do not support its use.

Processing of the `MajorVersion` and `MinorVersion` attributes is specified in section 3 .

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

```
<element name="Assertion" type="saml:AssertionAbstractType"/>
<complexType name="AssertionAbstractType" abstract="true">
    <sequence>
        <element ref="saml:Conditions" minOccurs="0"/>
        <element ref="saml:Advice" minOccurs="0"/>
    </sequence>
    <attribute name="MajorVersion" type="integer" use="required"/>
    <attribute name="MinorVersion" type="integer" use="required"/>
    <attribute name="AssertionID" type="saml:IDType" use="required"/>
    <attribute name="Issuer" type="string" use="required"/>
    <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

1.2.2 Element <SingleAssertion>

The element <SingleAssertion> of type SingleAssertionType is used to specify a SAML assertion that contains a single statement that is one of the following:

<Statement>
A statement defined in an extension schema (see section 3.2)

<SubjectStatement>
A subject statement defined in an extension schema (see section 3.2)

<AuthenticationStatement>
An authentication statement

<AuthorizationStatement>
An authorization statement

<AttributeStatement>
An attribute statement.

The following schema specifies the <SingleAssertion> element:

```
<element name="SingleAssertion" type="saml:SingleAssertionType"/>
<complexType name="SingleAssertionType">
    <complexContent>
        <extension base="saml:AssertionAbstractType">
            <choice>
                <element ref="saml:Statement"/>
                <element ref="saml:SubjectStatement"/>
                <element ref="saml:AuthenticationStatement"/>
                <element ref="saml:AuthorizationStatement"/>
                <element ref="saml:AttributeStatement"/>
            </choice>
        </extension>
    </complexContent>
</complexType>
```

1.2.3 Element <MultipleAssertion>

The element <MultipleAssertion> of type MultipleAssertionType is used to specify a SAML assertion that contains any number of the statement elements specified in section 1.2.2 above.

The following schema specifies the <MultipleAssertion> element:

```
<element name="MultipleAssertion" type="saml:MultipleAssertionType"/>
<complexType name="MultipleAssertionType">
    <complexContent>
        <extension base="saml:AssertionAbstractType">
            <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:Statement"/>
                <element ref="saml:SubjectStatement"/>
                <element ref="saml:AuthenticationStatement"/>
                <element ref="saml:AuthorizationStatement"/>
                <element ref="saml:AttributeStatement"/>
            </choice>
        </extension>
    </complexContent>
</complexType>
```

1.2.4 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.

<AssertionID>
: Specifies an assertion by reference to the value of its unique AssertionID attribute.

<Assertion>
: Specifies an assertion by value.

<SingleAssertion>
: Specifies an assertion containing a single statement by value.

<MultipleAssertion>
: Specifies an assertion containing multiple statements by value.

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"/>
        <element ref="saml:SingleAssertion"/>
        <element ref="saml:MultipleAssertion"/>
    </choice>
</complexType>
```

1.2.5 Element <Statement>

The <Statement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework.

The following schema specifies the <Statement> element:

```
<element name="Statement" type="saml:StatementAbstractType"/>
<complexType name="StatementAbstractType" abstract="true"/>
```

1.3 Subject Statement

A subject statement is a statement that relates to the specific SAML subject specified by a <Subject> element.

1.3.1 Element <SubjectStatement>

The <SubjectStatement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework.

The SubjectAssertionAbstractType type is an abstract type that extends the AssertionAbstractType to include a <Subject> element.

The following schema defines the <SubjectAssertion> element and SubjectAssertionAbstractType abstract type:

```
<element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
<complexType name="SubjectStatementAbstractType" abstract="true">
    <complexContent>
        <extension base="saml:StatementAbstractType">
            <sequence>
                <element ref="saml:Subject"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

1.3.2 Element <Subject>

The <Subject> element specifies a party by any 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 the statement is true for all of the subjects specified. For example if both a <NameIdentifier> and a <SubjectConfirmation> element are present the issuer is asserting that the statement is true of both parties.

The definition of a <Subject> element that intentionally identifies more than one principal is deprecated.

The following schema defines the <Subject> element:

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

1.3.2.1 Element <SubjectConfirmation>

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

<ConfirmationMethod>[Any number]
Each <AuthenticationMethod> element specifies a URI that identify a protocol that may be used to authenticate the subject.

<SubjectConfirmationData>[Optional]
Each <SubjectConfirmationData> 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 5 .

The following schema defines the <SubjectConfirmation> element:

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
    <sequence>
        <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
        <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
        <element ref="ds:KeyInfo" minOccurs="0"/>
    </sequence>
    <!-- Need to modify this element-->
</complexType>
```

1.3.2.2 Element <NameIdentifier>

The <NameIdentifier> element 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.[PHB2]

The following schema defines the <NameIdentifier> element:

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

1.3.2.3 Element <ConfirmationMethod>

The <ConfirmationMethod> element specifies the type of Authentication that took place.

The following schema defines the <ConfirmationMethod> element:

```
<element name="ConfirmationMethod" type="anyURI"/>
```

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.

1.4.1 Element <AuthenticationStatement>

The <AuthenticationStatement> element is of type AuthenticationStatementType, which extends the SubjectStatementAbstractType with the addition of the following elements:

AuthenticationMethod [Required]
: The AuthenticationMethod attribute specifies the type of Authentication that took place.

AuthenticationInstant [Required]
: The AuthenticationInstant attribute specifies the time at which the authentication took place.

<AuthenticationLocality> [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:

```
<element name="AuthenticationStatement"
          type="saml:AuthenticationStatementType"/>
<complexType name="AuthenticationStatementType">
   <complexContent>
      <extension base="saml:SubjectStatementAbstractType">
         <sequence>
            <element ref="saml:AuthenticationLocality" minOccurs="0"/>
         </sequence>
         <attribute name="AuthenticationMethod" type="anyURI"/>
         <attribute name="AuthenticationInstant" type="dateTime"/>
      </extension>
   </complexContent>
</complexType>
```

1.4.1.1 Element <AuthenticationLocality>

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

IPAddress [Optional]
: The IPAddress attribute of the system entity that performed the authentication

DNSAddress [Required]
: The DNSAddress attribute 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 <AuthenticationLocality> type:

```
<element name="AuthenticationLocality"
         type="saml:AuthenticationLocalityType"/>
<complexType name="AuthenticationLocalityType">
   <attribute name="IPAddress" type="string" use="optional"/>
   <attribute name="DNSAddress" type="string" use="optional"/>
</complexType>
```

1.5 Authorization Decision Statement

An authorization decision assertion contains a statement that assert 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 Element <AuthorizationStatement>

The <AuthorizationStatement> element extends the SubjectStatementAbstractType with the addition of the following elements:

Resource [Optional]
The Resource attribute specifies the resource by means of a URI.

<Actions> [Required]
The <Actions> element specifies the set of actions authorized for the specified resource.

Decision [Optional]
The <Decision> attribute specifies the decision with respect to the specified object.

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

If the <Namespace> element is not specified the namespace specified in section 5.2.1 is specified by default.

The following schema defines the <AuthorizationStatement> element:

```
<element name="AuthorizationStatement"
         type="saml:AuthorizationStatementType"/>
<complexType name="AuthorizationStatementType">
   <complexContent>
      <extension base="saml:SubjectStatementAbstractType">
         <sequence>
            <element ref="saml:Actions"/>
            <element ref="saml:Evidence"
                     minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="Resource" type="anyURI" use="optional"/>
         <attribute name="Decision"
                  type="saml:DecisionType" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

1.5.1.1 Element <Actions>

The Actions element specifies a set of actions within a specified action namespace.

<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 attribute is not specified the namespace specified in section 5.2.1 is specified by default.

The following schema defines the <Actions > type:

```
<element name="Actions" type="saml:ActionsType"/>
<complexType name="ActionsType">
    <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Namespace" type="anyURI" use="optional"/>
</complexType>
<element name="Action" type="string"/>
```

1.5.1.2 Element <Evidence>

The <Evidence> element specifies an assertion that the issuer relied upon in issuing the assertion in which the element is contained.

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 Statement

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

1.6.1 Element <AttributeStatement>

The <AttributeStatement> extends the SubjectStatementAbstractType 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:

```
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
    <complexContent>
        <extension base="saml:SubjectStatementAbstractType">
            <sequence>
                <element ref="saml:Attribute" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

1.6.1.1 Element <AttributeDesignator>

The <AttributeDesignator> element specifies an attribute name within an attribute namespace. The element is used in an attribute assertion query to request that attribute values within a specific namespace be returned. The <AttributeDesignator> element contains the following attributes:

AttributeNamespace [Required]
The <AttributeNamespace> attribute specifies the namespace in which the <AttributeName> elements are interpreted.

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

The following schema defines the <Attribute> element:

```
<element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
    <attribute name="AttributeName" type="string"/>
    <attribute name="AttributeNamespace" type="anyURI"/>
</complexType>
```

1.6.1.2 Element <Attribute>

The <Attribute> element specifies an attribute of the assertion subject. The <Attribute> element is an extension of the <AttributeDesignator> element that allows the attribute value to be specified. The <Attribute> element contains the following additional element:

<AttributeValue> [One or more]
Each <AttributeValue> element specifies a value of the attribute.

The following schema defines the <Attribute> element:

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            <sequence>
                <element ref="saml:AttributeValue"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

1.6.1.3 Type AttributeValueType

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

The following schema defines the AttributeValueType type:

```
<element name="AttributeValue" type="saml:AttributeValueType"/>
<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.

<Condition> [Any Number]
The <Condition> element provides an extension point allowing extension schemas to define new conditions.

<AudienceRestrictionCondition> [Any Number]
The <AudienceRestrictionCondition> condition specifies that the assertion is addressed to a particular audience.

NotBefore [Optional]
The NotBefore attribute specifies the earliest time instant at which the assertion is valid.

NotOnOrAfter [Optional]
The NotOnOrAfter attribute specifies the time instant at which the assertion has expired.

The following schema defines the <Conditions> element:

```
<element name="Conditions" type="saml:ConditionsType"/>
<complexType name="ConditionsType">
    <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="saml:Condition"/>
        <element ref="saml:AudienceRestrictionCondition"/>
    </choice>
    <attribute name="NotBefore" type="dateTime" use="optional"/>
    <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
```

1.7.1.1 Element <Condition>

The <Condition> element is of abstract type ConditionAbstractType and serves as an extension point for new condition definitions.

The following schema defines the <Condition> element:

```
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
```

1.7.1.2 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

The <AudienceRestrictionCondition> element specifies 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 condition evaluates to True if and only if the relying party is a member of one or more of the audiences specified.

The issuer of an assertion cannot prevent a party to whom it is disclosed making a decision on the basis of the information provided. However, the `<AudienceRestrictionCondition>` element allows the issuer to state explicitly that no warranty is provided to such a party in a machine and human readable form. While there can be no guarantee that a court would upholding such a warranty exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably improved.

The following schema defines the `<AudienceRestrictionCondition>` condition type:

```
<element name="AudienceRestrictionCondition"
       type="saml:AudienceRestrictionConditionType"/>
<complexType name="AudienceRestrictionConditionType">
    <complexContent>
        <extension base="saml:ConditionAbstractType">
            <sequence>
                <element ref="saml:Audience"
                         minOccurs="1" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

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"/>
```

```
691     <complexType name="AdviceType">
692        <sequence>
693           <choice minOccurs="0" maxOccurs="unbounded">
694              <element ref="saml:AssertionSpecifier"/>
695              <element ref="saml:AdviceElement"/>
696              <any namespace="##other" processContents="lax"/>
697           </choice>
698        </sequence>
699     </complexType>
700     <element name="AdviceElement" type="saml:AdviceAbstractType"/>
701     <complexType name="AdviceAbstractType"/>
702  </schema>
```

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 `<Request>` and `<Response>` elements described in this section. A `<Request>` from the client is followed by a `<Response>` 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-
open.org/committees/security/docs/draft-sstc-schema-protocol-15.xsd"
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-19.xsd"
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-19.xsd"
   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-19.xsd"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   elementFormDefault="unqualified">
   <import namespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-19.xsd"
      schemaLocation="draft-sstc-schema-assertion-19.xsd"/>
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-sstc-schema-protocol-19.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="Partial"/>
        <enumeration value="AllOrNone"/>
    </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[PHB3]

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 `RequestAbstractType`

All SAML requests are of types that are extensions of the `RequestAbstractType` abstract type. The `RequestAbstractType` requires that all SAML requests specify the version number of the SAML protocol and a request identifier:

`RequestID` [Required]
: 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.

`MajorVersion` [Required]
: Each request MUST specify the SAML major version identifier. The identifier for this version of SAML is 1.

`MinorVersion` [Required]
: Each request MUST specify the SAML minor version identifier. The identifier for this version of SAML is 0.

Processing of the `MajorVersion` and `MinorVersion` attributes is specified in section 3 .

The following schema defines the `RequestAbstractType` abstract type:

```
<complexType name="RequestAbstractType" abstract="true">
   <attribute name="RequestID" type="saml:IDType" use="required"/>
   <attribute name="MajorVersion" type="integer" use="required"/>
   <attribute name="MinorVersion" type="integer" use="required"/>
</complexType>
```

2.2.2 Element `<Request>`

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

`<Query>`
: The `<Query>` element is an extension point that allows extension schemas to define new types of query.

`<SubjectQuery>`
: The `<SubjectQuery>` element is an extension point that allows extension schemas to define new types of query that specify a single SAML subject.

`<AuthenticationQuery>`
: The `<AuthenticationQuery>` element specifies an authentication query.

`<AttributeQuery>`
: The `<AttributeQuery>` element specifies an attribute query.

817 `<AuthorizationQuery>`
818 The `<AuthorizationQuery>` element specifies an authorization query.

819 `<AssertionID>`
820 The `<AssertionID>` element requests an assertion by its assertion identifier.

821 `<AssertionArtifact>`
822 The `<AssertionArtifact>` element requests an assertion by means of an
823 assertion artifact that relates to it.

824 The following schema defines the `<Request>` element:

```
825 <element name="Request" type="samlp:RequestType"/>
826 <complexType name="RequestType">
827     <complexContent>
828         <extension base="samlp:RequestAbstractType">
829             <choice>
830                 <element ref="samlp:Query"/>
831                 <element ref="samlp:SubjectQuery"/>
832                 <element ref="samlp:AuthenticationQuery"/>
833                 <element ref="samlp:AttributeQuery"/>
834                 <element ref="samlp:AuthorizationQuery"/>
835                 <element ref="saml:AssertionID" maxOccurs="unbounded"/>
836                 <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
837             </choice>
838         </extension>
839     </complexContent>
840 </complexType>
841 <element name="AssertionArtifact" type="string"/>
```

842 2.2.3 Element `<Query>`

843 The `<Query>` element is an extension point that allows new SAML queries to be
844 defined. The Query `<Query>` element is of `QueryAbstractType` abstract type,
845 which is the base type from which all SAML request query elements are derived. The
846 `QueryAbstractType` abstract type contains no elements or attributes.

847 The following schema defines the `<Query>` element:

```
848 <element name="Query" type="samlp:QueryAbstractType"/>
849 <complexType name="QueryAbstractType" abstract="true"/>
```

850 2.2.4 Element `<SubjectQuery>`

851 The `<SubjectQuery>` element is an extension point that allows new SAML queries
852 that specify a single SAML subject to be defined. The `<SubjectQuery>` element is of
853 `SubjectQueryAbstractType` type extends the `Query` type to specify a query with
854 a specific subject as its principal.

855 The following schema defines the `<SubjectQuery>` element:

```
856 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
857 <complexType name="SubjectQueryAbstractType" abstract="true">
858     <complexContent>
859         <extension base="samlp:QueryAbstractType">
860             <sequence>
861                 <element ref="saml:Subject"/>
862             </sequence>
863         </extension>
864     </complexContent>
```

```
    </complexType>
```

2.3 Authentication Query

The <AuthenticationQuery> element is used to make the query "What authentication assertions are available for this Subject?"

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

2.3.1 Element <AuthenticationQuery>

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

<ConfirmationMethod> [Optional]
The <ConfirmationMethod> 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 ConfirmationMethod?"

A SAML processor will return a certain number of Authentication Assertions in response to an authentication query. The <Subject> of the returned assertions MUST be identical to the <Subject> element of the Query. If the <ConfirmationMethod> field is present in the Query at least one <ConfirmationMethod> field in the response MUST match. There is no implication that all such assertions must be returned.

The following schema defines the <AuthenticationQuery> type:

```
    <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
    <complexType name="AuthenticationQueryType">
        <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                <sequence>
                    <element ref="saml:ConfirmationMethod" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

2.4 Attribute Query

An Attribute Query is used to make the query "Return the requested attributes for this Subject"

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

2.4.1 Element <AttributeQuery>

An <AttributeQuery> element contains all the elements and attributes of a <SubjectQuery> element 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 `<AttributeQuery>` element:

```
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:AttributeDesignator"
                         minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="CompletenessSpecifier"
                   type="samlp:CompletenessSpecifierType" use="required"/>
        </extension>
    </complexContent>
</complexType>
```

2.5 Authorization Query

An Authorization Query is used to make 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 Element `<AuthorizationQuery>`

An `<AuthorizationQuery>` element contains all the elements and attributes of a `<SubjectQuery>` 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 `<AuthorizationQuery>` type:

```
<element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
<complexType name="AuthorizationQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:Actions"/>
                <element ref="saml:Evidence"
                         minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="Resource" type="anyURI"/>
        </extension>
    </complexContent>
</complexType>
```

2.6 Response

2.6.1 Abstract Type `ResponseAbstractType`

The response to a Request is of a type that extends the `ResponseAbstractType` abstract type. The `ResponseAbstractType` 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.

`ResponseID` [Required]
: The `ResponseID` attribute defines a unique identifier for the assertion response.

`InResponseTo` [Required]
: The `InResponseTo` attribute specifies the request to which the response corresponds. The `RequestID` element MUST match the `InResponseTo` element in the corresponding response.

`MajorVersion` [Required]
: Each response MUST specify the SAML major version identifier. The identifier for this version of SAML is 1.

`MinorVersion` [Required]
: Each response MUST specify the SAML minor version identifier. The identifier for this version of SAML is 0.

Processing of the `MajorVersion` and `MinorVersion` attributes is specified in section 3 .

The following schema defines the `ResponseAbstractType` abstract type:

```
<complexType name="ResponseAbstractType" abstract="true">
   <attribute name="ResponseID" type="saml:IDType" use="required"/>
   <attribute name="InResponseTo" type="saml:IDType" use="required"/>
   <attribute name="MajorVersion" type="integer" use="required"/>
   <attribute name="MinorVersion" type="integer" use="required"/>
</complexType>
```

2.6.2 Element `<Response>`

The `<Response>` element is extends the `ResponseAbstractType` 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 `<Response>` element:

```
<element name="Response" type="samlp:ResponseType"/>
<complexType name="ResponseType">
   <complexContent>
      <extension base="samlp:ResponseAbstractType">
         <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:Assertion"/>
            <element ref="saml:SingleAssertion"/>
            <element ref="saml:MultipleAssertion"/>
         </choice>
```

```
990                 <attribute name="StatusCode"
991                            type="samlp:StatusCodeType" use="required"/>
992             </extension>
993         </complexContent>
994     </complexType>
995 </schema>
```

3 SAML Versions and Extension Schemas

3.1 SAML Version

Version Numbers are defined in the following elements:

- Assertion
- Request
- Response

The version number of the SAML assertion is independent of the version number of the SAML protocol.

Each Version number consists of a major version number and a minor version number, both of which are integers. In accordance with industry practice a version number SHOULD be presented to the user in the form *Major.Minor*. This document describes version number 1.0 of the SAML Assertion and protocol.

The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

$$Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B = Minor_A)$$

New versions of SAML SHALL assign version numbers to assertions, requests and responses that are the same as or higher than the corresponding version number in the SAML version that immediately preceded it.

New versions of SAML SHALL assign new version numbers as follows:

Documentation change: $(Major_B = Major_A) \wedge (Minor_B = Minor_A)$
If the major and minor version numbers are unchanged, the new version *B* only introduces changes to the documentation that raise no compatibility issues with an implementation of version *A*.

Minor upgrade: $(Major_B = Major_A) \wedge (Minor_B > Minor_A)$
If the major version number of versions *A* and *B* are the same and the minor version number of *A* is higher than that of *B* the new SAML version may introduce changes to the SAML schema and semantics but any changes that are introduced in *B* SHALL be compatible with version *A*.

Major upgrade: $Major_B > Major_A$
If the major version of *B* number is higher than the major version of *A*, Version *B* may introduce changes to the SAML schema and semantics that are incompatible with *A*.

3.1.1 Assertion Version

A SAML application MUST NOT issue any assertion whose version number is not supported.

A SAML application MUST reject any assertion whose major version number is not supported.

A SAML application MAY reject any assertion whose version number is higher than the highest supported version.

3.1.2 Request Version

A SAML application SHOULD issue requests that specify the highest SAML version supported by both the sender and recipient.

If the SAML application does not know the capabilities of the recipient it should assume that it supports the highest SAML version supported by the sender.

3.1.3 Response Version

A SAML application MUST NOT issue responses that specify a higher SAML version number than the corresponding request.

A SAML application MUST NOT issue a response that has a major version number that is lower than the major version number of the corresponding request except to report the error `RequestVersionTooHigh`.

Incompatible protocol versions may cause the following errors to be reported:

`RequestVersionTooHigh`
: The protocol version specified in the request is a major upgrade from the highest protocol version supported by the responder.

`RequestVersionTooLow`
: The responder cannot respond to the particular request using the SAML version specified in the request because it is too low.

`RequestVersionDepricated`
: The responder does not respond to any requests with the protocol version specified in the request.

3.2 Schema Extension

The SAML schema is designed to support extensibility.

3.2.1 Use of sub-typing and substitution groups

XML provides two principle mechanisms for specifying an element of an extended type, sub-typing and substitution groups. For example an element of the new assertion type `NewStatementType` defined in the schema `new:` may be specified using the `xsi:type` attribute as follows:

```
<Assertion xmlns="...SAML.xsd" xmlns:new="...New.xsd" ...>
   <Statement xsi:type="new:NewStatementType">
      ...
   </Statement>
</Assertion>
```

Alternatively the new schema may define a `<NewStatment>` element of `NewStatementType` that is a member of the Statement substitution group:

```
<element "NewStatement" type="new:NewStatementType"
      substitutionGroup="saml:Statement"/>
```

The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML `<Statement>` element may be used:

```
<Assertion xmlns="...SAML.xsd" xmlns:new="...New.xsd" ...>
   <new:NewStatement>
      ...
   </new:NewStatement>
</Assertion>
```

The choice of using sub-typing or substitution groups has no effect on the semantics of the XML document but does have implications for interoperability:

Advantages of using sub-typing:

- A document can be partially interpreted by a parser that does not have access to the extension schema.
- At the time of writing some XML validating parsers do not support substitution groups whereas the xsi:type attribute is widely supported.

Advantages of using substitution groups:

- A document can be explained without the need to explain the importance of the `xsi:type` attribute.

The SAML schema is designed to permit processing of the assertion package to be separated from the processing of the statements it contains if either extension mechanism is used.

3.2.2 Extending the SAML Assertion Schema

The following elements are intended for use as extension points in an extension schema.

Element	Purpose
`<Assertion>`	Define new assertion package

<Statement>	Define new assertion statement
<SubjectStatement>	Define new assertion statement that takes a single subject as the subject.
<AttributeValue>	Define a structured attribute value
<Condition>	Define a new condition
<AdviceElement>	Define a new advice element

In addition the following elements that define specific instances of an assertion element may be extended to add specific additional functionality:

- SingleAssertion
- MultipleAssertion
- AuthenticationStatement
- AuthorizationStatement
- AttributeStatement
- AudienceRestrictionCondition

3.2.3 Extending the SAML Protocol Schema

The following elements are intended for use as extension points in an extension schema.

Element	Purpose
<Request>	Define new request type
<Query>	Define new query type
<SubjectQuery>	Define new subject query type
<Response>	Define a new response type

In addition the following elements that define specific instances of an assertion element may be extended to add specific additional functionality:

- <AuthenticationQuery>
- <AuthorizationQuery>
- <AttributeQuery>

4 References

[Kerberos] Needham, R., and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.

[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*

[XTAML] P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*, VeriSign Inc. September 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/

5 Identifiers

5.1 Authentication Protocol Identifiers

5.1.1 SAML Artifact

5.1.2 Assertion Bearer

5.1.3 User Name and Password (Pass-through)

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

5.1.5 Kerberos

5.1.6 SSL/TLS Certificate Based Client Authentication

5.1.7 Object Authenticator (SHA-1)

5.2 Action Identifiers

5.2.1 Read/Write/Execute/Delete/Control

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

5.2.3 Get/Head/Put/Post

5.2.4 UNIX file Permissions

6 Appendix

6.1 Assertion Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-19.xsd"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-19.xsd"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="unqualified">
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-sstc-schema-assertion-19.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:AssertionAbstractType"/>
   <complexType name="AssertionAbstractType" abstract="true">
      <sequence>
         <element ref="saml:Conditions" minOccurs="0"/>
         <element ref="saml:Advice" minOccurs="0"/>
      </sequence>
      <attribute name="MajorVersion" type="integer" use="required"/>
      <attribute name="MinorVersion" type="integer" use="required"/>
      <attribute name="AssertionID" type="saml:IDType" use="required"/>
      <attribute name="Issuer" type="string" use="required"/>
      <attribute name="IssueInstant" type="dateTime" use="required"/>
   </complexType>
   <element name="SingleAssertion" type="saml:SingleAssertionType"/>
   <complexType name="SingleAssertionType">
      <complexContent>
         <extension base="saml:AssertionAbstractType">
            <choice>
               <element ref="saml:Statement"/>
               <element ref="saml:SubjectStatement"/>
               <element ref="saml:AuthenticationStatement"/>
               <element ref="saml:AuthorizationStatement"/>
               <element ref="saml:AttributeStatement"/>
            </choice>
         </extension>
      </complexContent>
   </complexType>
   <element name="MultipleAssertion" type="saml:MultipleAssertionType"/>
   <complexType name="MultipleAssertionType">
      <complexContent>
         <extension base="saml:AssertionAbstractType">
            <choice minOccurs="0" maxOccurs="unbounded">
               <element ref="saml:Statement"/>
               <element ref="saml:SubjectStatement"/>
               <element ref="saml:AuthenticationStatement"/>
               <element ref="saml:AuthorizationStatement"/>
               <element ref="saml:AttributeStatement"/>
            </choice>
```

```
1223            </extension>
1224        </complexContent>
1225    </complexType>
1226    <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
1227    <complexType name="AssertionSpecifierType">
1228        <choice>
1229            <element ref="saml:AssertionID"/>
1230            <element ref="saml:Assertion"/>
1231            <element ref="saml:SingleAssertion"/>
1232            <element ref="saml:MultipleAssertion"/>
1233        </choice>
1234    </complexType>
1235    <element name="Statement" type="saml:StatementAbstractType"/>
1236    <complexType name="StatementAbstractType" abstract="true"/>
1237    <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1238    <complexType name="SubjectStatementAbstractType" abstract="true">
1239        <complexContent>
1240            <extension base="saml:StatementAbstractType">
1241                <sequence>
1242                    <element ref="saml:Subject"/>
1243                </sequence>
1244            </extension>
1245        </complexContent>
1246    </complexType>
1247    <element name="Subject" type="saml:SubjectType"/>
1248    <complexType name="SubjectType">
1249        <choice maxOccurs="unbounded">
1250            <element ref="saml:NameIdentifier"/>
1251            <element ref="saml:SubjectConfirmation"/>
1252            <element ref="saml:AssertionSpecifier"/>
1253        </choice>
1254    </complexType>
1255    <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1256    <complexType name="SubjectConfirmationType">
1257        <sequence>
1258            <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1259            <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
1260            <element ref="ds:KeyInfo" minOccurs="0"/>
1261        </sequence>
1262        <!-- Need to modify this element-->
1263    </complexType>
1264    <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1265    <complexType name="NameIdentifierType">
1266        <attribute name="SecurityDomain" type="string"/>
1267        <attribute name="Name" type="string"/>
1268    </complexType>
1269    <element name="ConfirmationMethod" type="anyURI"/>
1270    <element name="AuthenticationStatement"
1271              type="saml:AuthenticationStatementType"/>
1272    <complexType name="AuthenticationStatementType">
1273        <complexContent>
1274            <extension base="saml:SubjectStatementAbstractType">
1275                <sequence>
1276                    <element ref="saml:AuthenticationLocality" minOccurs="0"/>
1277                </sequence>
1278                <attribute name="AuthenticationMethod" type="anyURI"/>
1279                <attribute name="AuthenticationInstant" type="dateTime"/>
1280            </extension>
1281        </complexContent>
1282    </complexType>
1283    <element name="AuthenticationLocality"
1284              type="saml:AuthenticationLocalityType"/>
1285    <complexType name="AuthenticationLocalityType">
1286        <attribute name="IPAddress" type="string" use="optional"/>
1287        <attribute name="DNSAddress" type="string" use="optional"/>
1288    </complexType>
1289    <element name="AuthorizationStatement"
1290              type="saml:AuthorizationStatementType"/>
1291    <complexType name="AuthorizationStatementType">
1292        <complexContent>
1293            <extension base="saml:SubjectStatementAbstractType">
```

```
            <sequence>
                <element ref="saml:Actions"/>
                <element ref="saml:Evidence"
                       minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="Resource" type="anyURI" use="optional"/>
            <attribute name="Decision"
                   type="saml:DecisionType" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<element name="Actions" type="saml:ActionsType"/>
<complexType name="ActionsType">
    <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Namespace" type="anyURI" use="optional"/>
</complexType>
<element name="Action" type="string"/>
<element name="Evidence" type="saml:AssertionSpecifierType"/>
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
    <complexContent>
        <extension base="saml:SubjectStatementAbstractType">
            <sequence>
                <element ref="saml:Attribute" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
    <attribute name="AttributeName" type="string"/>
    <attribute name="AttributeNamespace" type="anyURI"/>
</complexType>
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            <sequence>
                <element ref="saml:AttributeValue"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="AttributeValue" type="saml:AttributeValueType"/>
<complexType name="AttributeValueType">
    <sequence>
        <any namespace="##any" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
<element name="Conditions" type="saml:ConditionsType"/>
<complexType name="ConditionsType">
    <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="saml:Condition"/>
        <element ref="saml:AudienceRestrictionCondition"/>
    </choice>
    <attribute name="NotBefore" type="dateTime" use="optional"/>
    <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
<element name="AudienceRestrictionCondition"
       type="saml:AudienceRestrictionConditionType"/>
<complexType name="AudienceRestrictionConditionType">
    <complexContent>
        <extension base="saml:ConditionAbstractType">
            <sequence>
                <element ref="saml:Audience"
                       minOccurs="1" maxOccurs="unbounded"/>
```

```
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
<element name="Advice" type="saml:AdviceType"/>
<complexType name="AdviceType">
    <sequence>
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:AssertionSpecifier"/>
            <element ref="saml:AdviceElement"/>
            <any namespace="##other" processContents="lax"/>
        </choice>
    </sequence>
</complexType>
<element name="AdviceElement" type="saml:AdviceAbstractType"/>
<complexType name="AdviceAbstractType"/>
</schema>
```

6.2 Protocol Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
    targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-19.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-19.xsd"
    xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-19.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="unqualified">
    <import namespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-19.xsd"
        schemaLocation="draft-sstc-schema-assertion-19.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>draft-sstc-schema-protocol-19.xsd</documentation>
    </annotation>
    <simpleType name="CompletenessSpecifierType">
        <restriction base="string">
            <enumeration value="Partial"/>
            <enumeration value="AllOrNone"/>
        </restriction>
    </simpleType>
    <simpleType name="StatusCodeType">
        <restriction base="string">
            <enumeration value="Success"/>
            <enumeration value="Failure"/>
            <enumeration value="Error"/>
            <enumeration value="Unknown"/>
        </restriction>
    </simpleType>
    <complexType name="RequestAbstractType" abstract="true">
        <attribute name="RequestID" type="saml:IDType" use="required"/>
        <attribute name="MajorVersion" type="integer" use="required"/>
        <attribute name="MinorVersion" type="integer" use="required"/>
    </complexType>
    <element name="Request" type="samlp:RequestType"/>
    <complexType name="RequestType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <choice>
                    <element ref="samlp:Query"/>
                    <element ref="samlp:SubjectQuery"/>
                    <element ref="samlp:AuthenticationQuery"/>
                    <element ref="samlp:AttributeQuery"/>
                    <element ref="samlp:AuthorizationQuery"/>
```

```
                <element ref="saml:AssertionID" maxOccurs="unbounded"/>
                <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
            </choice>
        </extension>
    </complexContent>
</complexType>
<element name="AssertionArtifact" type="string"/>
<element name="Query" type="samlp:QueryAbstractType"/>
<complexType name="QueryAbstractType" abstract="true"/>
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
    <complexContent>
        <extension base="samlp:QueryAbstractType">
            <sequence>
                <element ref="saml:Subject"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
<complexType name="AuthenticationQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:ConfirmationMethod" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:AttributeDesignator"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="CompletenessSpecifier"
                    type="samlp:CompletenessSpecifierType" use="required"/>
        </extension>
    </complexContent>
</complexType>
<element name="AuthorizationQuery" type="samlp:AuthorizationQueryType"/>
<complexType name="AuthorizationQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:Actions"/>
                <element ref="saml:Evidence"
                        minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="Resource" type="anyURI"/>
        </extension>
    </complexContent>
</complexType>
<complexType name="ResponseAbstractType" abstract="true">
    <attribute name="ResponseID" type="saml:IDType" use="required"/>
    <attribute name="InResponseTo" type="saml:IDType" use="required"/>
    <attribute name="MajorVersion" type="integer" use="required"/>
    <attribute name="MinorVersion" type="integer" use="required"/>
</complexType>
<element name="Response" type="samlp:ResponseType"/>
<complexType name="ResponseType">
    <complexContent>
        <extension base="samlp:ResponseAbstractType">
            <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:Assertion"/>
                <element ref="saml:SingleAssertion"/>
                <element ref="saml:MultipleAssertion"/>
            </choice>
            <attribute name="StatusCode"
```

```
                        type="samlp:StatusCodeType" use="required"/>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Page: 7
[PHB1] Need some better text here n'est pas?

Page: 11
[PHB2]Yikes! maybe we should put an IDType in here so that the security domain is at least unique!

Page: 21
[PHB3]Need to have text for these, how exactly does failure differ from error?