Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)

Document identifier: draft-sstc-core-21

Location: http://www.oasis-open.org/committees/security/docs

Publication date: 10 December 2001

Status: Interim draft. Send comments to the editors.

Contributors:[TBD][elm1]

Carlisle Adams, Entrust
Nigel Edwards, Hewlett-Packard
Marlena Erdos, Tivoli
Phillip Hallam-Baker, VeriSign, editor (pbaker@verisign.com)
Jeff Hodges, Oblix
Charles Knouse, Oblix
Chris McLaren, Netegrity
Prateek Mishra, Netegrity
RL “Bob” Morgan, University of Washington
Eve Maler, Sun Microsystems, editor (eve.maler@sun.com)
Tim Moses, Entrust
David Orchard, BEA
Irving Reid, Baltimore

1. Introduction

This specification defines the syntax and semantics for XML-encoded SAML assertions, protocol requests, and protocol responses. These constructs are typically embedded in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML specification for bindings and profiles **[SAMLBind]** provides frameworks for this embedding and transport. Files containing just the SAML assertion schema **[SAML-XSD]** and protocol schema **[SAMLP-XSD]** are available.

The following sections describe how to understand the rest of this specification.

1.1. Notation

This specification uses schema documents conforming to W3C XML Schema **[Schema1]** and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC2119]**:

> *"they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)"*

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

```
Listings of SAML schemas appear like this.
```

```
Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is present in the example:

- The prefix `saml:` stands for the SAML assertion namespace.
- The prefix `samlp:` stands for the SAML request-response protocol namespace.
- The prefix `ds:` stands for the W3C XML Signature namespace.
- The prefix `xsd:` stands for the W3C XML Schema namespace in example listings. In schema listings, this is the default namespace and no prefix is shown.

This specification uses the following typographical conventions in text: `<SAMLElement>`, `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

1.2. Schema Organization and Namespaces

The SAML assertion structures are defined in a schema **[SAML-XSD]** associated with the following XML namespace:

```
http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd
```

The SAML request-response protocol structures are defined in a schema **[SAMLP-XSD]** associated with the following XML namespace:

```
http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd
```

Note: The SAML namespace names are temporary and will change when SAML 1.0 is finalized.

The assertion schema is imported into the protocol schema. Also imported into both schemas is the schema for XML Signature **[XMLSig-XSD]**, which is associated with the following XML namespace[elm2]:

```
http://www.w3.org/2000/09/xmldsig#
```

The XML Signature element `<ds:KeyInfo>`, defined in **[XMLSig]** §4.4, is of particular interest in SAML.

1.3. SAML Concepts (Non-Normative)

This section is informative only and is superseded by any contradicting information in the normative text in Sections 1.2 and following. A glossary of SAML terms and concepts **[SAMLGloss]** is available.

[TBD][elm3]

2. Assertions

An assertion is a package of information that supplies one or more statements made by an issuer. SAML allows issuers to make three different kinds of assertion statement:

- **Authentication:** The specified subject was authenticated by a particular means at a particular time.
- **Authorization decision:** A request to allow the specified subject to access the specified object has been granted or denied.
- **Attribute:** The specified subject is associated with the supplied attributes.

Assertions have a nested structure. A series of inner elements representing authentication statements, authorization decision statements, and attribute statements contains the specifics, while an outer generic assertion element provides information that is common to all the statements.

2.1. Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the assertion schema:

```
<schema
   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-21.xsd"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-21.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-21.xsd</documentation>
   </annotation>
…
</schema>
```

2.2. Simple Types

The following sections define the SAML assertion-related simple types.

2.2.1. Simple Type IDType

The **IDType** simple type is used to declare and reference identifiers to assertions, requests and responses.

Values of attributes declared to be of type **IDType** MUST satisfy the following properties:

- Any party that assigns an identifier 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 declares that is has a particular identifier, 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}.

It is OPTIONAL for an identifier based on **IDType** to be resolvable in principle to some resource. In the case that the identifier is resolvable in principle (for example, the identifier is in the form of a URI reference), it is OPTIONAL for the identifier to be dereferenceable.

The following schema fragment defines the **IDType** type:

```
    <simpleType name="IDType">
        <restriction base="string"/>
    </simpleType>
```

2.2.2. Simple Type DecisionType

The **DecisionType** simple type defines the possible values to be reported as the status of an authorization decision statement.

`Permit`
The specified action is permitted.

`Deny`
The specified action is denied.

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

The following schema fragment defines the **DecisionType** type:

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

2.3. Assertions

The following sections define the SAML constructs that contain assertion information.

2.3.1. Element <AssertionSpecifier>

The `<AssertionSpecifier>` element specifies an assertion either by reference or by value. It contains one of the following subelements:

`<AssertionID>`
Specifies an assertion by reference to the value of the assertion's `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 fragment defines the `<AssertionSpecifier>` element and its **AssertionSpecifierType** complex type:

```
    <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>
```

2.3.2. Element <AssertionID>

The `<AssertionID>` element specifies a reference to the identifier of a SAML assertion.

The following schema fragment defines the `<AssertionID>` element and **IDType** type:

```
    <element name="AssertionID" type="saml:IDType"/>
```

2.3.3. Element <Assertion>

A SAML assertion is specified by an XML element whose type is derived from the abstract XML type **AssertionAbstractType**. This type specifies the basic information that is common to all assertions. Instances of SAML assertions have concrete types that are extensions of the base **AssertionAbstractType**.

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 as an integer in this attribute. The identifier for this version of SAML is 1. Processing of this attribute is specified in Section 4.

`MinorVersion` [Required]
Each assertion MUST specify the SAML minor version identifier as an integer in this attribute. The identifier for this version of SAML is 0. Processing of this attribute is specified in Section 4.

`AssertionID` [Required]
The identifier for this assertion.

`Issuer` [Required]
The issuer of the assertion. The name of the issuer is provided as a string[PHB4].

`IssueInstant` [Required]
The time instant of issue.

`<Conditions>` [Optional]
Conditions that MUST be taken into account in assessing the validity of the assertion.

`<Advice>` [Optional]
Additional information related to the assertion that assists processing in certain situations but which MAY be ignored by applications that do not support its use.

The following schema fragment defines the `<Assertion>` element and its **AssertionAbstractType** complex 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>
```

2.3.3.1. Element <Conditions>

If an assertion contains a `<Conditions>` element, the validity of the assertion is dependent on the conditions provided. Each condition evaluates to a status of `Valid`, `Invalid`, or `Indeterminate`. The validity status of an assertion is the conjunction of the validity of each of the conditions it contains, 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 conditions evaluate to `Valid`, the assertion status is `Valid`.

The `<Conditions>` element MAY be extended to define additional conditions. If an element contained within a `<Conditions>` element is encountered that is not understood, the status of the condition MUST be evaluated to `Indeterminate`.

The `<Conditions>` element contains the following element and attributes:

`NotBefore` [Optional]
Specifies the earliest time instant at which the assertion is valid.

`NotOnOrAfter` [Optional]
Specifies the time instant at which the assertion has expired.

`<Condition>` [Zero or more]
Provides an extension point allowing extension schemas to define new conditions.

`<AudienceRestrictionCondition>` [Any Number]
Specifies that the assertion is addressed to a particular audience.

The following schema fragment defines the `<Conditions>` element and its **ConditionsType** complex type:

```
<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>
```

2.3.3.1.1 Attributes NotBefore and NotOnOrAfter

The `NotBefore` and `NotOnOrAfter` attributes specify time 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

If the value for either `NotBefore` or `NotOnOrAfter` is omitted or is equal to the start of the epoch it is considered unspecified. If the `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid at any time.

The NotBefore and NotOnOrAfter attributes are defined to have the **dateTime** simple type that is built in to the W3C XML Schema Datatypes specification **[Schema2]**. All time instants are interpreted to be in Universal Coordinated Time (UTC) unless they explicitly indicate a time zone.

Implementations MUST NOT generate time instants that specify leap seconds.

2.3.3.1.2 Element <Condition>

The <Condition> element serves as an extension point for new conditions.

The following schema fragment defines the <Condition> element and its **ConditionAbstractType** complex type:

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

2.3.3.1.3 Element <AudienceRestrictionCondition>

The <AudienceRestrictionCondition> element specifies that the assertion is addressed to one or more specific audiences. Although a party that is outside the audiences 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. The URI MAY identify a document that describes the terms and conditions of audience membership.

The condition evaluates to Valid 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 fragment defines the <AudienceRestrictionCondition> element and its **AudienceRestrictionConditionType** complex 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"/>
```

2.3.3.2. Element <Advice>

The <Advice> element contains any additional information that the issuer wishes to provide. This information MAY be ignored by applications without affecting either the semantics or the validity of the assertion. The <Advice> element serves as an extension point for specialized kinds of advice.

Following are some potential uses of the <Advice> element:

- 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 fragment defines the `<Advice>` element and its **AdviceType** complex type:

```
<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"/>
```

2.3.4. Element <SingleAssertion>

The `<SingleAssertion>` element specifies a single statement. It contains one of the following:

`<Statement>`
A statement defined in an extension schema.

`<SubjectStatement>`
A subject statement defined in an extension schema.

`<AuthenticationStatement>`
An authentication statement.

`<AuthorizationStatement>`
An authorization decision statement.

`<AttributeStatement>`
An attribute statement.

The following schema fragment defines the `<SingleAssertion>` element and its **SingleAssertionType** complex type:

```
<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>
```

2.3.5. Element <MultipleAssertion>

The `<MultipleAssertion>` element specifies a series of zero or more statements. It can contain the same statement elements that are allowed in `<SingleAssertion>`. Multiple statements in a <MultipleAssertion> element MUST be interpreted identically to several <SingleAssertion> elements with the same common information that contain the statements individually.

The following schema fragment defines the <MultipleAssertion> element and its **MultipleAssertionType** complex type:

```
    <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>
```

2.4. Statements

The following sections define the SAML constructs that contain statement information.

2.4.1. Element <Statement>

The <Statement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. Its **StatementAbstractType** complex type is abstract; extension elements MUST use the xsi:type attribute to indicate the derived type.

The following schema fragment defines the <Statement> element and its **StatementAbstractType** complex type:

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

2.4.2. Element <SubjectStatement>

The <SubjectStatement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. It contains a <Subject> element that allows an issuer to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends **StatementAbstractType**, is abstract; extension elements MUST use the xsi:type attribute to indicate the derived type.

The following schema fragment defines the <SubjectAssertion> element and its **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>
```

2.4.2.1. Element <Subject>

The `<Subject>` element specifies one or more subjects. It contains a mixture of one or more of the following elements:

`<NameIdentifier>`
An identification of a subject by its name and security domain.

`<SubjectConfirmation>`
Information that allows the subject to be authenticated.

`<AssertionSpecifier>`
An identification of a subject by reference to or containment of an assertion.

If a `<Subject>` element contains more than one subject specification, the issuer is asserting that the surrounding 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 subjects being identified. A `<Subject>` element SHOULD NOT identify more than one principal.

The following schema fragment defines the `<Subject>` element and its **SubjectType** complex type:

```
    <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>
```

2.4.2.2. Element <NameIdentifier>

The `<NameIdentifier>` element specifies a subject by a combination of a name and a security domain. It has the following attributes:

`SecurityDomain`
The security domain governing the name of the subject.

`Name`
The name of the subject.

The interpretation of the security domain and the name are left to individual implementations, including issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties.

The following schema fragment defines the `<NameIdentifier>` element and its **NameIdentifierType** complex type:

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

2.4.2.3. Element <SubjectConfirmation>

The `<SubjectConfirmation>` element specifies a subject by supplying data that allows the subject to be authenticated. It contains the following elements in order:

`<ConfirmationMethod>` [One or more]
A URI that identifies a protocol to be used to authenticate the subject. URIs identifying common authentication protocols are specified in Section 6.

`<SubjectConfirmationData>` [Zero or more]
Additional authentication information to be used by a specific authentication protocol.

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

The following schema fragment defines the `<SubjectConfirmation>` element and its **SubjectConfirmationType** complex type:

```
    <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>
```

2.4.2.3.1 Element <ConfirmationMethod>

The following schema fragment defines the `<ConfirmationMethod>` element:

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

2.4.2.3.2 Element <SubjectConfirmationData>

The following schema fragment defines the `<SubjectConfirmationData>` element:

```
            <element name="SubjectConfirmationData" type="string" minOccurs="0"/>
```

2.4.3. Element <AuthenticationStatement>

The `<AuthenticationStatement>` element supplies a statement by the issuer that its subject was authenticated by a particular means at a particular time. It is of type **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the addition of the following element and attributes:

`AuthenticationMethod` [Required]
Specifies the type of authentication that took place.

`AuthenticationInstant` [Required]
Specifies the time at which the authentication took place.

`<AuthenticationLocality>` [Optional]
Specifies the DNS domain name and IP address for the system entity that performed the authentication.

The following schema fragment defines the `<AuthenticationStatement>` element and its **AuthenticationStatementType** complex 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>
```

2.4.3.1. Element <AuthenticationLocality>

The `<AuthenticationLocality>` element specifies the DNS domain name and IP address for the system entity that was authenticated. It has the following attributes:

`IPAddress` [Optional]
The IP address of the system entity that that was authenticated.

`DNSAddress` [Required]
The DNS address of the system entity that that was authenticated.

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

The following schema fragment defines the `<AuthenticationLocality>` element and its **AuthenticationLocalityType** complex 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>
```

2.4.4. Element <AuthorizationStatement>

The `<AuthorizationStatement>` element supplies a statement by the issuer that the request for access by the specified subject to the specified resource has resulted in the specified decision on the basis of some optionally specified evidence. It is of type **AuthorizationStatementType**, which extends **SubjectStatementAbstractType** with the addition of the following elements (in order) and attributes:

`Resource` [Optional]
A URI identifying the resource to which access authorization is sought.

`Decision` [Optional]
The decision rendered by the issuer with respect to the specified object. The value is of the **DecisionType** simple type.

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

`<Evidence>` [Zero or more]
A set of assertions that the issuer relied on in making the decision.

The following schema fragment defines the `<AuthorizationStatement>` element and its **AuthorizationStatementType** complex type:

```
<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>
```

2.4.4.1. Elements <Actions> and <Action>

The `<Actions>` element specifies the set of actions on the specified resource for which permission is sought. It has the following element and attribute:

`Namespace` [Optional]
A URI representing the namespace in which the names of specified actions are to be interpreted. If this element is absent, the namespace specified in section [TBD][elm5] is in effect by default.

`<Action>` [One or more]
An action sought to be performed on the specified resource.

The following schema fragment defines the `<Actions>` element, its **ActionsType** complex type, and the `<Action>` element:

```
    <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"/>
```

2.4.4.2. Element <Evidence>

The `<Evidence>` element contains an assertion that the issuer relied on in issuing the authorization decision. It has the **AssertionSpecifierType** complex type.

The provision of an assertion as evidence MAY affect the reliance agreement between the client and the 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 fragment defines the `<Evidence>` element:

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

2.4.5. Element <AttributeStatement>

The `<AttributeStatement>` element supplies a statement by the issuer that the specified subject is associated with the specified attributes. It is of type **AttributeStatementType**, which extends **SubjectStatementAbstractType** with the addition of the following element:

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

The following schema fragment defines the `<AttributeStatement>` element and its **AttributeStatementType** complex 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>
```

2.4.5.1. Elements <AttributeDesignator> and <Attribute>

The `<AttributeDesignator>` element identifies an attribute name within an attribute namespace. It has the **AttributeDesignatorType** complex type. It 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 fragment defines the `<AttributeDesignator>` element and its **AttributeDesignatorType** complex type:

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

The `<Attribute>` element supplies the value for an attribute of an assertion subject. It has the **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following element:

`<AttributeValue>` [Required]
The value of the attribute.

The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

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

2.4.5.1.1 Element <AttributeValue>

The `<AttributeValue>` element supplies the value of the specified attribute. It is of the **AttributeValueType** complex type, which allows the inclusion of any element in any namespace and specifies that lax schema validation is in effect.

The following schema fragment defines the `<AttributeValue>` element and its **AttributeValueType** complex type:

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

3. Protocol

SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings and profiles specification for SAML **[SAMLBind]** describes specific means of transporting assertions using existing widely deployed protocols.

SAML-aware clients MAY in addition use the SAML request-response protocol defined by the `<Request>` and `<Response>` elements. The client sends a `<Request>` element to a SAML service, and the service generates a `<Response>` element, as shown in Figure 1.

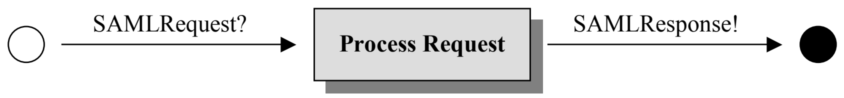

Figure 1: SAML Request-Response Protocol

3.1. Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the protocol schema.

```
<schema
   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-21.xsd"
   xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-21.xsd"
   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-21.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-21.xsd"
      schemaLocation="draft-sstc-schema-assertion-21.xsd"/>
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="xmldsig-core-schema.xsd"/>
   <annotation>
      <documentation>draft-sstc-schema-protocol-21.xsd</documentation>
   </annotation>
…
</schema>
```

3.2. Simple Types

The following sections define the SAML protocol-related simple types.

3.2.1. Simple Type CompletenessSpecifierType

The **CompletenessSpecifierType** simple type is used in an attribute query request to filter the response in cases where the requester is not authorized to receive part or all of the response. The type enumerates the following possible values:

`Partial`
If there are any parts of the response that the requester is not authorized to receive, the requester is asking for a response containing just those parts of the response that the requester is authorized to receive.

`AllOrNone`
If there are any parts of the response that the requester is not authorized to receive, the requester is asking for a response containing no assertions.

The following schema fragment defines the **CompletenessSpecifierType** simple type:

```
    <simpleType name="CompletenessSpecifierType">
        <restriction base="string">
            <enumeration value="Partial"/>
            <enumeration value="AllOrNone"/>
        </restriction>
    </simpleType>
```

3.2.2. Simple Type StatusCodeType

The **StatusCodeType** simple type is used in a response to specify the status of the request that caused the response to be generated. The type enumerates the following possible values:

`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[PHB6]

The following schema fragment defines the **StatusCodeType** simple type:

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

3.3. Requests

The following sections define the SAML constructs that contain request information.

3.3.1. Complex Type RequestAbstractType

All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type. This type defines common attributes that are associated with all SAML requests:

`RequestID` [Required]
An identifier for the request. The values of the `RequestID` attribute in a request and the `InResponseTo` attribute in the corresponding response MUST match.

`MajorVersion` [Required]
Each request MUST specify the SAML major version identifier. The identifier for this version of SAML is 1. Processing of this attribute is specified in Section 4.

`MinorVersion` [Required]
Each request MUST specify the SAML minor version identifier. The identifier for this version of SAML is 0. Processing of this attribute is specified in Section 4.

The following schema fragment defines the **RequestAbstractType** complex 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>
```

3.3.2. Element <Request>

The `<Request>` element specifies a SAML request. It provides either a query or a request for a specific assertion identified by `<AssertionID>` or `<AssertionArtifact>`. It has the complex type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following elements:

`<Query>`
An extension point that allows extension schemas to define new types of query.

`<SubjectQuery>`
An extension point that allows extension schemas to define new types of query that specify a single SAML subject.

`<AuthenticationQuery>`
Makes a query for authentication information.

`<AttributeQuery>`
Makes a query for attribute information.

`<AuthorizationQuery>`
Makes a query for an authorization decision.

`<AssertionID>` [One or more]
Requests an assertion by reference to its assertion identifier.

`<AssertionArtifact>` [One or more]
Requests an assertion by supplying an assertion artifact that represents it.

The following schema fragment defines the `<Request>` element and its **RequestType** complex type:

```
    <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"/>
```

3.4. Queries

The following sections define the SAML constructs that contain query information.

3.4.1. Element <Query>

The `<Query>` element is an extension point that allows new SAML queries to be defined. Its **QueryAbstractType** is abstract; extension elements MUST use the `xsi:type` attribute to indicate the derived type. **QueryAbstractType** is the base type from which all SAML query elements are derived.

The following schema fragment defines the `<Query>` element and its **QueryAbstractType** complex type:

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

3.4.2. Element <SubjectQuery>

The `<SubjectQuery>` element is an extension point that allows new SAML queries that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements MUST use the `xsi:type` attribute to indicate the derived type. **SubjectQueryAbstractType** adds the `<Subject>` element.

The following schema fragment defines the `<SubjectQuery>` element and its **SubjectQueryAbstractType** complex type:

```
<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>
```

3.4.3. Element <AuthenticationQuery>

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 assertion containing an authentication statement. This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element:

`<ConfirmationMethod>` [Optional]
A filter for possible responses. If it is present, the query made is “What authentication assertions do you have for this subject with the supplied confirmation method?”

In response to an authentication query, a responder returns assertions with authentication statements as follows: The `<Subject>` element in the returned assertions MUST be identical to the `<Subject>` element of the query. If the `<ConfirmationMethod>` element is present in the query, at least one `<ConfirmationMethod>` element in the response MUST match. It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.

The following schema fragment defines the `<AuthenticationQuery>` type and its **AuthenticationQueryType** complex 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>
```

3.4.4. Element <AttributeQuery>

The `<AttributeQuery>` element is used to make the query "Return the requested attributes for this subject." The response will be in the form of an assertion containing an attribute statement. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

`CompletenessSpecifier` [Required] (see Section 0)
Filters the response in cases where the requester is not authorized to receive part or all of the response. If its value is `Partial`, the query made is "Return any requested attributes for this subject that this requester is authorized to see." If its value is `AllOrNone`, the query made is "Return the requested attributes for this subject, if and only if this requester is authorized to see all of them."

`<AttributeDesignator>` [Zero or more] (see Section 2.4.5.1)
Each `<AttributeDesignator>` element specifies an attribute whose value is to be returned. If no attributes are specified, the list of desired attributes is implicit and application-specific.

The following schema fragment defines the `<AttributeQuery>` element and its **AttributeQueryType** complex type:

```
    <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>
```

3.4.5. Element <AuthorizationQuery>

The `<AuthorizationQuery>` element is used to make the query "Should these actions on this resource be allowed for this subject, given this evidence?" The response will be in the form of an assertion containing an authorization decision statement. This element is of type **AuthorizationQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following elements and attribute:

`Resource` [Required]
A URI indicating the resource for which authorization is requested.

`<Actions>` [Required]
The actions for which authorization is requested.

`<Evidence>` [Zero or more]
An assertion that the responder MAY rely on in making its response.

The following schema fragment defines the `<AuthorizationQuery>` element and its **AuthorizationQueryType** complex 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>
```

3.5. Responses

The following sections define the SAML constructs that contain response information.

3.5.1. Complex Type ResponseAbstractType

All SAML responses are of types that are derived from the abstract **ResponseAbstractType** complex type. This type defines common attributes that are associated with all SAML responses:

`ResponseID` [Required]
An identifier for the response.

`InResponseTo` [Required]
A reference to the identifier of the request to which the response corresponds. The value of this attribute MUST match the value of the corresponding `RequestID` attribute.

`MajorVersion` [Required]
Each response MUST specify the SAML major version identifier. The identifier for this version of SAML is 1. Processing of this attribute is specified in Section 4.

`MinorVersion` [Required]
Each response MUST specify the SAML minor version identifier. The identifier for this version of SAML is 0. Processing of this attribute is specified in Section 4.

The following schema fragment defines the **ResponseAbstractType** complex 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>
```

3.5.2. Element <Response>

The `<Response>` element specifies the status of the corresponding SAML request and a list of zero or more assertions that answer the request. It has the complex type **ResponseType**, which extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture) and attribute:

`StatusCode` [Required] (see Section 3.2.2)
A code representing the status of the corresponding request.

`<Assertion>` (see Section 2.3.3)
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 fragment defines the `<Response>` element and its **ResponseType** complex type:

```
    <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>
```

4. Assertion and Protocol Versioning

SAML version information appears in the following elements:

- `<Assertion>`
- `<Request>`
- `<Response>`

The version numbering of the SAML assertion is independent of the version number of the SAML request-response protocol. The version information for each 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 defines SAML Assertions 1.0 and SAML Protocol 1.0.

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

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

Each revision 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 *B* is higher than that of *A*, 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*.

4.1. 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.

4.2. 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.

4.3. 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.

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

5. Schema Extension

The SAML schemas support extensibility. An example of an application that extends SAML assertions is the XTAML system for management of embedded trust roots **[XTAML]**. The following sections explain how to use the extensibility features in SAML to create extension schemas.

Note that elements in the SAML schemas are not blocked from substitution, so that all SAML elements MAY serve as the head element of a substitution group. Also, types are not defined as `final`, so that all SAML types MAY be extended and restricted. The following sections discuss only elements that have been specifically designed to support extensibility.

5.1. Assertion Schema Extension

The SAML assertion schema is designed to permit separate processing of the assertion package and the statements it contains, if the extension mechanism is used for either part.

The following elements are intended specifically for use as extension points in an extension schema; their types are set to `abstract`, so that the use of an `xsi:type` attribute with these elements is REQUIRED:

- `<Assertion>`
- `<Statement>`
- `<SubjectStatement>`
- `<AttributeValue>`
- `<Condition>`
- `<AdviceElement>`

In addition, the following elements that are directly usable as part of SAML MAY be extended:

- `<SingleAssertion>`
- `<MultipleAssertion>`
- `<AuthenticationStatement>`
- `<AuthorizationStatement>`
- `<AttributeStatement>`
- `<AudienceRestrictionCondition>`

5.2. Protocol Schema Extension

The following elements are intended specifically for use as extension points in an extension schema; their types are set to `abstract`, so that the use of an `xsi:type` attribute with these elements is REQUIRED:

- `<Request>`
- `<Query>`
- `<SubjectQuery>`
- `<Response>`

In addition, the following elements that are directly usable as part of SAML MAY be extended:

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

5.3. Use of Type Derivation and Substitution Groups

W3C XML Schema **[Schema1]** provides two principal mechanisms for specifying an element of an extended type: type derivation and substitution groups.

For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived from **StatementType**. The following example of a SAML assertion assumes that the extension schema (represented by the `new:` prefix) has defined this new type:

```
<saml:Assertion …>
  <saml:Statement xsi:type="new:NewStatementType">
  …
  </saml:Statement>
</saml:Assertion>
```

Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a substitution group that has `<Statement>` as a head element. For the substituted element to be schema-valid, it needs to have a type that matches or is derived from the head element's type. The following is an example of an extension schema fragment that defines this new element:

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

The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML `<Statement>` element can be used. The following is an example of a SAML assertion that uses the extension element:

```
<saml:Assertion …>
   <new:NewStatement>
     …
   </new:NewStatement>
</saml:Assertion>
```

The choice of extension method has no effect on the semantics of the XML document but does have implications for interoperability.

The advantages of type derivation are as follows:

- A document can be more fully interpreted by a parser that does not have access to the extension schema because a "native" SAML element is available.
- At the time of writing, some W3C XML Schema validators do not support substitution groups, whereas the `xsi:type` attribute is widely supported.

The advantage of substitution groups is that a document can be explained without the need to explain the functioning of the `xsi:type` attribute.

6. Identifiers

The following sections define URI-based identifiers for common authentication protocols and actions.

[TBD][elm7]

6.1. Confirmation Method Identifiers

The following identifiers MAY be used in the `<ConfirmationMethod>` element (see Section 2.4.2.3.1) to refer to common authentication protocols.

SAML Artifact:
[TBD]

Assertion Bearer:
[TBD]

User Name and Password (Pass-Through):
[TBD]

User Name and Password (One-Way-Function SHA-1):
[TBD]

Kerberos **[Kerberos]**:
[TBD]

SSL/TLS Certificate Based Client Authentication:
[TBD]

Object Authenticator (SHA-1):
[TBD]

6.2. Action Namespace Identifiers

The following identifiers MAY be used in the `ActionNamespace` attribute (see Section 2.4.4.1) to refer to common sets of actions to perform on resources.

Read/Write/Execute/Delete/Control:
[TBD]

Read/Write/Execute/Delete/Control with Negation:
[TBD]

Get/Head/Put/Post:
[TBD]

UNIX File Permissions:
[TBD]

7. Schema Listings

The following sections contain complete listings of the assertion and protocol schemas for SAML. [TBD][elm8]

7.1. Assertion Schema

Following is a complete listing of the SAML assertion schema **[SAML-XSD]**.

```
<?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-21.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>
            </extension>
        </complexContent>
    </complexType>
    <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>
    <element name="Statement" type="saml:StatementAbstractType"/>
    <complexType name="StatementAbstractType" abstract="true"/>
    <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>
    <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>
    <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>
    <element name="NameIdentifier" type="saml:NameIdentifierType"/>
    <complexType name="NameIdentifierType">
        <attribute name="SecurityDomain" type="string"/>
        <attribute name="Name" type="string"/>
    </complexType>
    <element name="ConfirmationMethod" type="anyURI"/>
    <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>
    <element name="AuthenticationLocality"
                type="saml:AuthenticationLocalityType"/>
    <complexType name="AuthenticationLocalityType">
        <attribute name="IPAddress" type="string" use="optional"/>
        <attribute name="DNSAddress" type="string" use="optional"/>
    </complexType>
    <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>
    <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>
```

7.2. Protocol Schema

Following is a complete listing of the SAML protocol schema **[SAMLP-XSD]**.

```
<?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-21.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-21.xsd"
    xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-21.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-21.xsd"
        schemaLocation="draft-sstc-schema-assertion-21.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>draft-sstc-schema-protocol-21.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>
```

8. References

[TBD][elm9]

[Kerberos]	R. Needham et al., *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.
[PKCS1]	B. Kaliski, *PKCS #1: RSA Encryption Version 2*.0, RSA Laboratories, also IETF RFC 2437, October 1998.
[RFC2104]	H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*, http://www.ietf.org/rfc/rfc2104.txt, IETF RFC 2104, February 1997.
[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997
[SAML-XSD]	P. Hallam-Baker et al., *SAML assertion schema*, http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd, OASIS, December 2001.
[SAMLBind]	P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf, OASIS, December 2001.
[SAMLGloss]	J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf, OASIS, December 2001.
[SAMLP-XSD]	P. Hallam-Baker et al., *SAML protocol schema*, http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd, OASIS, December 2001.
[Schema1]	H. S. Thompson et al., *XML Schema Part 1: Structures*, http://www.w3.org/TR/xmlschema-1/, World Wide Web Consortium Recommendation, May 2001.
[Schema2]	P. V. Biron et al., *XML Schema Part 2: Datatypes*, http://www.w3.org/TR/xmlschema-2, World Wide Web Consortium Recommendation, May 2001.
[XMLSig]	D. Eastlake et al., *XML-Signature Syntax and Processing*, http://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.
[XMLSig-XSD]	XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd.
[XMLEnc]	*XML Encryption Specification*, In development.
[XTAML]	P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*, http://www.xmltrustcenter.org/, VeriSign Inc. September 2001.

Appendix A. Notices

Page: 1
[elm1]Check this list. Should we just list editors? List whole TC? List whoever wants to be listed?

Page: 6
[elm2]Check: This is old!

Page: 6
[elm3]Need conceptual material here. Explain concepts/terms such as the domain model, URIs for identifiers, extension points, etc.

Page: 9
[PHB4] Need some better text here n’est pas?

Page: 17
[elm5]I don’t know what’s really supposed to go here.

Page: 20
[PHB6]Need to have text for these, how exactly does failure differ from error?

Page: 30
[elm7]Need to get these filled in.

Page: 31
[elm8]The references to the XML Signature schema are outdated. In the SAML schemas, and throughout the spec, this needs to be fixed.

Page: 37
[elm9]: Need to check and sort all references.