Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

Working Draft 05, 17 February 2004

Document identifier:
sstc-saml-core-2.0-draft-05

Location:
http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:
Scott Cantor, individual (cantor.2@osu.edu)
John Kemp, Nokia (john.kemp@nokia.com)
Eve Maler, Sun Microsystems (eve.maler@sun.com)

Contributors:
Stephen Farrell, Baltimore Technologies
Irving Reid, Baltimore Technologies
Hal Lockhart, BEA Systems
David Orchard, BEA Systems
Krishna Sankar, Cisco Systems
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Bob Blakley, IBM
Marlena Erdos, IBM
RL “Bob” Morgan, individual
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Prateek Mishra, Netegrity (co-chair)
Charles Knouse, Oblix
Simon Godik, Overxeer
Rob Philpott, RSA Security (co-chair)
Darren Platt, formerly of RSA Security
Jahan Moreh, Sigaba
Jeff Hodges, Sun Microsystems
Phillip Hallam-Baker, VeriSign (former editor)

Abstract:

This specification defines the syntax and semantics for XML-encoded assertions about authentication, attributes and authorization, and for the protocols that conveys this information.

Status:

This is a working draft produced by the Security Services Technical Committee. Publication of this draft does not imply TC endorsement. This is an active working draft that may be updated, replaced, or obsoleted at any time. **See the Revision History for details of changes made in this revision.**

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them to the security-services-comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to security-services-comment-request@lists.oasis-open.org with "subscribe" in the body) or use other OASIS-supported means of submitting comments. The committee will publish vetted errata on the Security Services TC web page (http://www.oasis-open.org/committees/security/).

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the Security Services TC (http://www.oasis-open.org/committees/security/ipr.php).

Table of Contents

1 Introduction

This specification defines the syntax and semantics for Security Assertion Markup Language (SAML) assertions and the protocols for requesting and returning them. SAML assertions, requests, and responses are encoded in XML and use XML namespaces [XMLNS]. They 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 and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages.

The keywords "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 [RFC 2119]:

> …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.
```

In cases of disagreement between the SAML schema documents [SAML-XSD] [SAMLP-XSD] and this specification, the schema documents take precedence.

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

- The prefix `saml:` stands for the SAML assertion namespace, `urn:oasis:names:tc:SAML:2.0:assertion`.
- The prefix `samlp:` stands for the SAML request-response protocol namespace, `urn:oasis:names:tc:SAML:2.0:protocol`.
- The prefix `ds:` stands for the W3C XML Signature namespace, `http://www.w3.org/2000/09/xmldsig#` [XMLSig-XSD].
- The prefix `xenc:` stands for the W3C XML Encryption namespace, `http://www.w3.org/2001/04/xmlenc#` .
- The prefix `xsd:` stands for the W3C XML Schema namespace, `http://www.w3.org/2001/XMLSchema` [Schema1], 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:

```
urn:oasis:names:tc:SAML:2.0:assertion
```

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

```
urn:oasis:names:tc:SAML:2.0:protocol
```

The assertion schema is imported into the protocol schema. Also imported into both schemas is the schema for XML Signature , which is associated with the following XML namespace:

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

See Section SAML Namespace Version for information on SAML namespace versioning.

1.2.1 String and URI Values

All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively, which are built in to the W3C XML Schema Datatypes specification . All strings in SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the XML Recommendation §2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise indicated in this specification, all URI reference values MUST consist of at least one non-whitespace character, and are REQUIRED to be absolute [RFC 2396].

1.2.2 Time Values

All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes specification , and MUST be expressed in UTC form.

SAML system entities SHOULD NOT rely on other applications supporting time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

1.2.3 ID and ID Reference Values

The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties in addition to those imposed by the definition of the **xsd:ID** type itself:

- Any party that assigns an identifier MUST ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object.
- Where a data object declares that it has a particular identifier, there MUST be exactly one such declaration.

The mechanism by which a SAML system entity 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 or equal to 2^{-128} and SHOULD be less than or equal to 2^{-160}. This requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in length. The encoding must conform to the rules defining the **xsd:ID** datatype.

The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that **xsd:IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML reference identifier might actually be defined in a document separate from that in which the identifier

reference is used, which violates the **xsd:IDREF** requirement that its value match the value of an ID attribute on some element in the same XML document.

1.2.4 Comparing SAML Values

Unless otherwise noted, all elements in SAML documents that have the XML Schema **xsd:string** type, or a type derived from that, MUST be compared using an exact binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-insensitive string comparisons, normalization or trimming of white space, or conversion of locale-specific formats such as numbers or currency. This requirement is intended to conform to the W3C Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

If an implementation is comparing values that are represented using different character encodings, the implementation MUST use a comparison method that returns the same result as converting both values to the Unicode character encoding, Normalization Form C [UNICODE-C], and then performing an exact binary comparison. This requirement is intended to conform to the W3C Character Model for the World Wide Web [W3C-CharMod], and in particular the rules for Unicode-normalized Text.

Applications that compare data received in SAML documents to data from external sources MUST take into account the normalization rules specified for XML. Text contained within elements is normalized so that line endings are represented using linefeed characters (ASCII code $10_{Decimal}$), as described in the XML Recommendation §2.11. Attribute values defined as strings (or types derived from strings) are normalized as described in §3.3.3. All white space characters are replaced with blanks (ASCII code $32_{Decimal}$).

The SAML specification does not define collation or sorting order for attribute or element values. SAML implementations MUST NOT depend on specific sorting orders for values, because these can differ depending on the locale settings of the hosts involved.

2 SAML Assertions

An assertion is a package of information that supplies one or more statements made by a SAML authority. This SAML specification defines three different kinds of assertion statement that can be created by a SAML authority. As mentioned above and described in Section SAML Extensions, extensions are permitted by the SAML assertion schema, allowing user-defined extensions to assertions and statements, as well as allowing the definition of new kinds of assertion and statement. The three kinds of statement defined in this specification are:

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

The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, attribute, authorization decision, or user-defined statements containing the specifics.

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="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
       elementFormDefault="unqualified"
       attributeFormDefault="unqualified"
       blockDefault="substitution"
       version="2.0">
       <import namespace="http://www.w3.org/2000/09/xmldsig#"
       schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
       <import namespace="http://www.w3.org/2001/04/xmlenc#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
schema.xsd"/>
       <annotation>
              <documentation>
                 Document identifier: sstc-saml-schema-assertion-2.0
                 Location: http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security
                 Revision history:
                 V2.0:
                   Updated the schema and namespace to V2.0.
                   Removed <AuthorityBinding> and corresponding type.
                    Added new NameIdentifier and Issuer types and elements.
       </documentation>
       </annotation>
…
</schema>
```

2.2 Simple Types

The following section defines the SAML assertion-related simple types.

2.2.1 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`

The SAML authority cannot determine whether the specified action is permitted or denied.

The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to provide an affirmative statement that it is not able to issue a decision. Additional information as to the reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>` elements.

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

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

2.3 Name Identifiers

The following sections define the SAML constructs that contain descriptive identifiers of subjects and assertion and message issuers.

2.3.1 Element <BaseNameIdentifier>

The `<BaseNameIdentifier>` element is an extension point that allows applications to add new kinds of name identifiers. Its **BaseNameIdentifierAbstractType** complex type is abstract and is thus usable only as the base of a derived type. It defines the following common attributes for all name identifier representations:

`NameQualifier` [Optional]

The security or administrative domain that qualifies the name identifier of the subject. This attribute provides a means to federate names from disparate user stores without collision.

`SPNameQualifier` [Optional]

Further qualifies a federated name identifier with the name of the service provider or affiliation of providers which has federated the principal's identity.

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

```
<element name="BaseNameIdentifier" type="saml:BaseNameIdentifierAbstractType"/>
<complexType name="BaseNameIdentifierAbstractType" abstract="true">
```

```
    <complexContent>
        <extension base="anyType">
            <attribute name="NameQualifier" type="string" use="optional"/>
            <attribute name="SPNameQualifier" type="string" use="optional"/>
        </extension>
    </complexContent>
</complexType>
```

2.3.2 Element <NameIdentifier>

The <NameIdentifier> element is of type **NameIdentifierType**, which restricts **BaseNameIdentifierAbstractType** to simple string content and provides additional attributes as follows:

Format [Optional]

A URI reference representing the classification of string-based identifier information. See Section NameIdentifier Format Identifiers for some URI references that MAY be used as the value of the Format attribute and their associated descriptions and processing rules. If no Format value is provided, the identifier urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see Section Unspecified) is in effect.

When a Format value other than those specified in Section NameIdentifier Format Identifiers is used, the content of the <NameIdentifier> element is to be interpreted according to the specification of that format as defined outside of this specification. If not otherwise indicated by the specification of the format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties are implementation-specific.

SPProvidedIdentifier [Optional]

The name identifier established by the service provider or affiliation of providers for the principal, if different from the primary name identifier given in the content of the <NameIdentifier> element.

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

```
<element name="NameIdentifier" type="saml:NameIdentifierType"/>
<complexType name="NameIdentifierType" mixed="false">
    <simpleContent>
        <restriction base="saml:BaseNameIdentifierAbstractType">
            <simpleType>
                <restriction base="string"/>
            </simpleType>
            <attribute name="Format" type="anyURI" use="optional"/>
            <attribute name="SPProvidedIdentifier" type="string"
use="optional"/>
        </restriction>
    </simpleContent>
</complexType>
```

2.3.3 Element <EncryptedNameIdentifier>

The <EncryptedNameIdentifier> element extends **BaseNameIdentifierAbstractType** to carry the content of the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification . The <EncryptedNameIdentifier> element contains the following additional elements and attributes:

<xenc:EncryptedData> [Required]

The encrypted content and associated encryption details, as defined by . The encrypted content MUST contain an element that has a type that is derived from **BaseNameIdentifierAbstractType**.

<xenc:EncryptedKey> [Zero or more]

Wrapped decryption keys, as defined by **[XMLEnc]**. Each wrapped key SHOULD include a Recipient attribute that specifies the entity for whom the key has been encrypted.

Encrypted identifiers are intended as a privacy protection when the plain-text value passes through an intermediary; as such, the ciphertext MUST be unique to any given encryption operation. For more on such issues, see **§6.3**.

The following schema fragment defines the <EncryptedNameIdentifier> element and its **EncryptedNameIdentifierType** complex type:

```
<element name="EncryptedNameIdentifier"
type="saml:EncryptedNameIdentifierType"/>
    <complexType name="EncryptedNameIdentifierType" mixed="false">
        <complexContent>
            <restriction base="saml:BaseNameIdentifierType">
                <sequence>
                    <element ref="xenc:EncryptedData"/>
                    <element ref="xenc:EncryptedKey" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
            </restriction>
    </complexContent>
</complexType>
```

2.3.4 Element <Issuer>

The <Issuer> element, with complex type **NameIdentifierType**, provides structured information about the issuer of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name, but permits various attributes of descriptive metadata.

The following schema fragment defines the <Issuer> element:

```
<element name="Issuer" type="saml:NameIdentifierType"/>
```

2.4 Assertions

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

2.4.1 Element <AssertionIDReference>

The <AssertionIDReference> element makes a reference to a SAML assertion.

The following schema fragment defines the <AssertionIDReference> element:

```
<element name="AssertionIDReference" type="NCName"/>
```

2.4.2 Element <Assertion>

The <Assertion> element is of **AssertionType** complex type. This type specifies the basic information that is common to all assertions, including the following elements and attributes:

`MajorVersion` [Required]

The major version of this assertion. The identifier for the version of SAML defined in this specification is 2. SAML versioning is discussed in Section SAML Versioning.

`MinorVersion` [Required]

The minor version of this assertion. The identifier for the version of SAML defined in this specification is 0. SAML versioning is discussed in Section SAML Versioning.

`AssertionID` [Required]

The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in Section 1.2.3 for identifier uniqueness.

`IssueInstant` [Required]

The time instant of issue in UTC, as described in Section Time Values.

`<Issuer>` [Required]

The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be unambiguous to the intended relying parties. If the Format attribute is omitted, the identifier `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see section 7.3.1) is assumed.

This specification defines no relationship between the entity represented by this element and the signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the assertion or to specific profiles are application-specific.

`<ds:Signature>` [Optional]

An XML Signature that authenticates the assertion, as described in Section SAML and XML Signature Syntax and Processing.

`<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.

One or more of the following statement elements:

`<Statement>`

A statement defined in an extension schema.

`<SubjectStatement>`

A subject statement defined in an extension schema.

`<AuthenticationStatement>`

An authentication statement.

`<AuthorizationDecisionStatement>`

An authorization decision statement.

`<AttributeStatement>`

An attribute statement.

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

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
      <sequence>
            <element ref="saml:Issuer"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="saml:Conditions" minOccurs="0"/>
            <element ref="saml:Advice" minOccurs="0"/>
            <choice maxOccurs="unbounded">
                  <element ref="saml:Statement"/>
                  <element ref="saml:SubjectStatement"/>
                  <element ref="saml:AuthenticationStatement"/>
                  <element ref="saml:AuthorizationDecisionStatement"/>
                  <element ref="saml:AttributeStatement"/>
            </choice>
      </sequence>
      <attribute name="MajorVersion" type="integer" use="required"/>
      <attribute name="MinorVersion" type="integer" use="required"/>
      <attribute name="AssertionID" type="ID" use="required"/>
      <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>Element <Conditions>
```

The `<Conditions>` element MAY contain the following elements and attributes:

`NotBefore` [Optional]

Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC as described in Section Time Values.

`NotOnOrAfter` [Optional]

Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as described in Section Time Values.

`<Condition>` [Any Number]

Provides an extension point allowing extension schemas to define new conditions.

`<AudienceRestrictionCondition>` [Any Number]

Specifies that the assertion is addressed to a particular audience.

<`DoNotCacheCondition`**>** [Any Number]

Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future use.

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:AudienceRestrictionCondition"/>
      <element ref=”saml:DoNotCacheCondition”>
            <element ref="saml:Condition"/>
      </choice>
      <attribute name="NotBefore" type="dateTime" use="optional"/>
      <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
```

If an assertion contains a `<Conditions>` element, the validity of the assertion is dependent on the sub-elements and attributes provided. When processing the sub-elements and attributes of a `<Conditions>` element, the following rules MUST be used in the order shown to determine the overall validity of the assertion:

1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is considered to be ***Valid***.

2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the assertion is ***Invalid***.

3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity of the assertion cannot be determined and is deemed to be ***Indeterminate***.

4. If all sub-elements and attributes of the <Conditions> element are determined to be ***Valid***, then the assertion is considered to be ***Valid***.

The <Conditions> element MAY be extended to contain additional conditions. If an element contained within a <Conditions> element is encountered that is not understood, the status of the condition cannot be evaluated and the validity status of the assertion MUST be deemed to be ***Indeterminate*** in accordance with rule 3 above.

Note that an assertion that has validity status ***Valid*** may not be trustworthy for reasons such as not being issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

2.4.2.0.1 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 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 . All time instants are specified in Universal Coordinated Time (UTC) as described in Section Time Values.

Implementations MUST NOT generate time instants that specify leap seconds.

2.4.2.0.2 Element <Condition>

The <Condition> element serves as an extension point for new conditions. Its **ConditionAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

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

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

2.4.2.0.3 Elements <AudienceRestrictionCondition> and <Audience>

The <AudienceRestrictionCondition> element specifies that the assertion is addressed to one or more specific audiences identified by <Audience> elements. Although a SAML relying party that is outside the audiences specified is capable of drawing conclusions from an assertion, the SAML authority

explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the following elements:

<Audience>

A URI reference that identifies an intended audience. The URI reference MAY identify a document that describes the terms and conditions of audience membership.

The audience restriction condition evaluates to ***Valid*** if and only if the SAML relying party is a member of one or more of the audiences specified.

The SAML authority cannot prevent a party to whom the assertion is disclosed from taking action on the basis of the information provided. However, the <AudienceRestrictionCondition> element allows the SAML authority 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 uphold 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" maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

2.4.2.0.4 Element <DoNotCacheCondition>

Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be retained for future use. A SAML authority SHOULD NOT include more than one <DoNotCacheCondition> element within a <Conditions> element of an assertion. Note that no Relying Party implementation is required to perform caching. However, any that do so MUST observe this condition. If multiple <DoNotCacheCondition> elements appear within a <Conditions> element, a Relying Party MUST treat the multiple elements as though a single <DoNotCacheCondition> element was specified. For the purposes of determining the validity of the <Conditions> element, the <DoNotCacheCondition> (see Section Element <Conditions>) is considered to always be valid.

```
<element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType" />
<complexType name="DoNotCacheConditionType">
   <complexContent>
     <extension base="saml:ConditionAbstractType"/>
   </complexContent>
</complexType>
```

2.4.2.1 Element <Advice>

The <Advice> element contains any additional information that the SAML authority wishes to provide. This information MAY be ignored by applications without affecting either the semantics or the validity of the assertion.

The <Advice> element contains a mixture of zero or more <Assertion> elements, <AssertionIDReference> elements, and elements in other namespaces, with lax schema validation in effect for these other elements.

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">
      <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:AssertionIDReference"/>
            <element ref="saml:Assertion"/>
            <any namespace="##other" processContents="lax"/>
      </choice>
</complexType>
```

2.5 Statements

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

2.5.1 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 and is thus usable only as the base of a 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"/>
```

2.5.2 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 a SAML authority to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends **StatementAbstractType**, is abstract and is thus usable only as the base of a derived type.

The **SubjectStatementAbstractType** contains an optional SessionIndex attribute.

SessionIndex [Optional]

Indexes a particular session between the subject, and the authority issuing this statement. The value of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST NOT be a globally unique value for a principal's session at the authority.

The following schema fragment defines the <SubjectStatement> element and its **SubjectStatementAbstractType** abstract type:

```
<element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
```

```
<complexType name="SubjectStatementAbstractType" abstract="true">
      <complexContent>
            <extension base="saml:StatementAbstractType">
                  <sequence>
                        <element ref="saml:Subject"/>
                  </sequence>
                  <attribute name="SessionIndex" type="string"
use="optional"/>
            </extension>
      </complexContent>
</complexType>
```

2.5.2.1 Element <Subject>

The <Subject> element specifies the principal that is the subject of the statement. It contains either or both of the following elements:

<NameIdentifier>, <EncryptedNameIdentifier>, or <BaseNameIdentifier>

An identification of a subject by its name, security domain(s), and other associated information

<SubjectConfirmation>

Information that allows the subject to be authenticated.

If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the SAML authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying party is the entity that the SAML authority associates with the <NameIdentifier>. 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>
            <sequence>
                  <choice>
                        <element ref="saml:BaseNameIdentifier"/>
                        <element ref="saml:NameIdentifier"/>
                        <element ref="saml:EncryptedNameIdentifier"/>
                  </choice>
                  <element ref="saml:SubjectConfirmation" minOccurs="0"/>
            </sequence>
            <element ref="saml:SubjectConfirmation"/>
      </choice>
</complexType>
```

2.5.2.2

2.5.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>

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 reference that identifies a protocol to be used to authenticate the subject. URI references identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the SAML bindings and profiles specification [SAMLBind]. Additional methods may be added by defining new profiles or by private agreement.

<SubjectConfirmationData> [Optional]

Additional authentication information to be used by a specific authentication protocol.

<ds:KeyInfo> [Optional]

An XML Signature element that provides access to a cryptographic key held by the subject.

The following schema fragment defines the <SubjectConfirmation> element and its **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData> element and the <ConfirmationMethod> element:

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
      <sequence>
            <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
            <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
            <element ref="ds:KeyInfo" minOccurs="0"/>
      </sequence>
</complexType>
<element name="SubjectConfirmationData" type="anyType"/>
<element name="ConfirmationMethod" type="anyURI"/>
```

2.5.3 Element <AuthenticationStatement>

The <AuthenticationStatement> element describes a statement by the SAML authority asserting that the statement's 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 elements and attributes:

AuthenticationMethod [Required]

A URI reference that specifies the type of authentication that took place. URI references identifying common authentication protocols are listed in Section Authentication Method Identifiers.

AuthenticationInstant [Required]

Specifies the time at which the authentication took place. The time value is encoded in UTC as described in Section Time Values.

<SubjectLocality> [Optional]

Specifies the DNS domain name and IP address for the system entity from which the subject was apparently authenticated.

Note: The <AuthorityBinding> element and its corresponding type were removed from <AuthenticationStatement> for V2.0 of SAML.

<AuthenticationStatement> elements MUST contain a SessionIndex value, conforming to the rules specified in section 2.5.2.

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:SubjectLocality" minOccurs="0"/>
                  </sequence>
                  <attribute name="AuthenticationMethod" type="anyURI"
use=”required”/>
                  <attribute name="AuthenticationInstant" type="dateTime"
use=”required”/>
            </extension>
      </complexContent>
</complexType>
```

2.5.3.1 Element <SubjectLocality>

The `<SubjectLocality>` 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 was authenticated.

`DNSAddress` [Optional]

The DNS address of the system entity 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 `<SubjectLocality>` element and its **SubjectLocalityType** complex type:

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

2.5.4 Element <AttributeStatement>

The `<AttributeStatement>` element describes a statement by the SAML authority asserting that the statement’s 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>
```

2.5.4.1 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 query to request that attribute values within a specific namespace be returned (see Section Element <AttributeQuery> for more information). The <AttributeDesignator> element contains the following XML attributes:

AttributeNamespace [Required]

The namespace in which the AttributeName elements are interpreted.

AttributeName [Required]

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" use="required"/>
      <attribute name="AttributeNamespace" type="anyURI" use="required"/>
</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> [Any Number]

The value of the attribute. If an attribute contains more than one discrete value, it is RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute exists but has no value, then the <AttributeValue> element MUST be omitted.

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" minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

2.5.4.1.1 Element <AttributeValue>

The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType** type, which allows any well-formed XML to appear as the content of the element.

If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer** or **xsd:string**), the data type MAY be declared explicitly by means of an xsi:type declaration in the <AttributeValue> element. If the attribute value contains structured data, the necessary data elements MAY be defined in an extension schema.

The following schema fragment defines the <AttributeValue> element:

```
<element name="AttributeValue" type="anyType"/>
```

2.5.5 Element <AuthorizationDecisionStatement>

The <AuthorizationDecisionStatement> element describes a statement by the SAML authority asserting that a request for access by the statement's subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence.

The resource is identified by means of a URI reference. In order for the assertion to be interpreted correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

> In general, the rules for equivalence and definition of a normal form, if any, are scheme dependent. When a scheme uses elements of the common syntax, it will also use the common syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL with an explicit ":port", where the port is the default for the scheme, is equivalent to one where the port is elided.

To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the URI normalized form wherever possible as follows:

- SAML authorities SHOULD encode all resource URI references in normalized form.
- Relying parties SHOULD convert resource URI references to normalized form prior to processing.

Inconsistent URI reference interpretation can also result from differences between the URI reference syntax and the semantics of an underlying file system. Particular care is required if URI references are employed to specify an access control policy language. The following security conditions should be satisfied by the system which employs SAML assertions:

- Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive, a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a part of the resource URI reference.
- Many file systems support mechanisms such as logical paths and symbolic links, which allow users to establish logical equivalences between file system entries. A requester SHOULD NOT be able to gain access to a denied resource by creating such an equivalence.

The <AuthorizationDecisionStatement> element is of type **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the addition of the following elements (in order) and attributes:

Resource [Required]

A URI reference identifying the resource to which access authorization is sought. It is permitted for this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

Decision [Required]

The decision rendered by the SAML authority with respect to the specified resource. The value is of the **DecisionType** simple type.

<Action> [One or more]

The set of actions authorized to be performed on the specified resource.

<Evidence> [Optional]

A set of assertions that the SAML authority relied on in making the decision.

The following schema fragment defines the <AuthorizationDecisionStatement> element and its **AuthorizationDecisionStatementType** complex type:

```
<element name="AuthorizationDecisionStatement"
type="saml:AuthorizationDecisionStatementType"/>
<complexType name="AuthorizationDecisionStatementType">
      <complexContent>
            <extension base="saml:SubjectStatementAbstractType">
                  <sequence>
                        <element ref="saml:Action" maxOccurs="unbounded"/>
                        <element ref="saml:Evidence" minOccurs="0"/>
                  </sequence>
                  <attribute name="Resource" type="anyURI" use="required"/>
                  <attribute name="Decision" type="saml:DecisionType"
use="required"/>
            </extension>
      </complexContent>
</complexType>
```

2.5.5.1 Element <Action>

The <Action> element specifies an action on the specified resource for which permission is sought. It has the following attribute and string-data content:

Namespace [Optional]

A URI reference representing the namespace in which the name of the specified action is to be interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwedc-negation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

string data [Required]

An action sought to be performed on the specified resource.

The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
<element name="Action" type="saml:ActionType"/>
<complexType name="ActionType">
      <simpleContent>
            <extension base="string">
                  <attribute name="Namespace" type="anyURI"/>
            </extension>
      </simpleContent>
</complexType>
```

2.5.5.2 Element <Evidence>

The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one or more of the following elements:

<AssertionIDReference> [Any number]

Specifies an assertion by reference to the value of the assertion's AssertionID attribute.

<Assertion> [Any number]

Specifies an assertion by value.

Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party and the SAML authority making the authorization decision. For example, in the case that the SAML relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use that assertion as evidence in making its authorization decision without endorsing the `<Evidence>` element's assertion as valid either to the relying party or any other third party.

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

```
<element name="Evidence" type="saml:EvidenceType"/>
<complexType name="EvidenceType">
      <choice maxOccurs="unbounded">
            <element ref="saml:AssertionIDReference"/>
            <element ref="saml:Assertion"/>
      </choice>
</complexType>
```

3 SAML Protocols

SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of protocols. The bindings and profiles specification for SAML [SAMLBind] describes specific means of transporting queries, assertions, and other messages using existing widely deployed protocols.

SAML request and response messages derive from common abstract types, described below, The requester sends an element derived from **RequestAbstractType** to a SAML responder, and the responder generates an element derived from **StatusResponseType**, as shown in Figure 1. Each request element typically corresponds to a specific kind of response element.

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="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       elementFormDefault="unqualified"
       attributeFormDefault="unqualified"
       blockDefault="substitution"
       version="2.0">
       <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
              schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
       <import namespace="http://www.w3.org/2000/09/xmldsig#"
              schemaLocation=" http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd "/>
       <annotation>
              <documentation>
                 Document identifier: sstc-saml-schema-protocol-2.0
                 Location: http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security
                 Revision history:
                   Updated the schema and namespace to V2.0.
                   Removed <RespondWith> and its corresponding type.
                   Added Issuer element, revamped protocol types, added RNI
messages.
              </documentation>
       </annotation>
…
</schema>
```

3.2 Requests and Responses

The following sections define the SAML constructs that underlie request and response messages.

3.2.1 Complex Type RequestAbstractType

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

`RequestID` [Required]

An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in Section 1.2.3 for identifier uniqueness. The values of the `RequestID` attribute in a request and the `InResponseTo` attribute in the corresponding response MUST match.

`MajorVersion` [Required]

The major version of this request. The identifier for the version of SAML defined in this specification is 2. SAML versioning is discussed in Section SAML Versioning.

`MinorVersion` [Required]

The minor version of this request. The identifier for the version of SAML defined in this specification is 0. SAML versioning is discussed in Section SAML Versioning.

`IssueInstant` [Required]

The time instant of issue of the request. The time value is encoded in UTC as described in Section Time Values.

`Consent` [Optional]

Indicates whether or not consent has been obtained from a user in the sending this request.

`<Issuer>` [Optional]

Identifies the entity that generated the request message.

`<ds:Signature>` [Optional]

An XML Signature that authenticates the request, as described in Section SAML and XML Signature Syntax and Processing.

`<RelayState>` [Optional]

This contains state information that MUST be relayed back in the associated response.

`<Extensions>` [Optional]

This contains optional protocol message extension elements that are agreed upon between the communicating parties.

> **Note:** The `<RespondWith>` element has been removed from `<Request>` for V2.0 of SAML.

The following schema fragment defines the **RequestAbstractType** complex type:

```
<complexType name="RequestAbstractType" abstract="true">
      <sequence>
            <element ref="saml:Issuer" minOccurs="0"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="samlp:RelayState" minOccurs="0"/>
            <element ref="samlp:Extensions" minOccurs="0"/>
      </sequence>
      <attribute name="RequestID" type="ID" use="required"/>
      <attribute name="MajorVersion" type="integer" use="required"/>
      <attribute name="MinorVersion" type="integer" use="required"/>
      <attribute name="IssueInstant" type="dateTime" use="required"/>
      <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

```
<element name="Extensions" type="samlp:ExtensionsType"/>
<complexType name="ExtensionsType">
      <sequence>
            <any namespace="##any" processContents="lax"
maxOccurs="unbounded"/>
      </sequence>
</complexType>
```

3.2.1.1 Element <RelayState>

SAML requests MAY contain a string-valued element containing state information that the requester wishes the responder to include in the response. This is particularly useful with asynchronous bindings of protocol messages, such as the encoding of messages in browser URLs. This data SHOULD be integrity-protected by the requester and MAY have other protections placed on it by the requester, such as confidentiality. The length of this value SHOULD be kept as short as possible because of limitations of the bindings in which it may be needed.

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

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

3.2.2 Complex Type StatusResponseType

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

`ResponseID` [Required]

An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in Section 1.2.3 for identifier uniqueness.

`InResponseTo` [Optional]

A reference to the identifier of the request to which the response corresponds, if any. If the response is not generated in response to a request, or if the `RequestID` attribute value of a request cannot be determined (because the request is malformed), then this attribute MUST NOT be present. Otherwise, it MUST be present and its value MUST match the value of the corresponding `RequestID` attribute value.

`MajorVersion` [Required]

The major version of this response. The identifier for the version of SAML defined in this specification is 2. SAML versioning is discussed in Section SAML Versioning.

`MinorVersion` [Required]

The minor version of this response. The identifier for the version of SAML defined in this specification is 0. SAML versioning is discussed in Section SAML Versioning.

`IssueInstant` [Required]

The time instant of issue of the response. The time value is encoded in UTC as described in Section Time Values.

`Recipient` [Optional]

The intended recipient of this response. This is useful to prevent malicious forwarding of responses to unintended recipients, a protection that is required by some use profiles. It is set by the generator of the response to a URI reference that identifies the intended recipient. If present, the actual recipient MUST check that the URI reference identifies the recipient or a resource managed by the recipient. If it does not, the response MUST be discarded.

<Issuer> [Optional]

Identifies the entity that generated the response message.

<ds:Signature> [Optional]

An XML Signature that authenticates the response, as described in Section SAML and XML Signature Syntax and Processing.

<RelayState> [Optional]

This contains state information from the associated request being relayed back in the response. It MUST match the <RelayState> value in the associated request, if any.

<Extensions> [Optional]

This contains optional protocol message extension elements that are agreed upon between the communicating parties.

<Status> [Required]

A code representing the status of the corresponding request.

The following schema fragment defines the **StatusResponseType** complex type:

```
<complexType name="StatusResponseType">
      <sequence>
            <element ref="saml:Issuer" minOccurs="0"/>
            <element ref="ds:Signature" minOccurs="0"/>
            <element ref="samlp:RelayState" minOccurs="0"/>
            <element ref="samlp:Extensions" minOccurs="0"/>
            <element ref="samlp:Status"/>
      </sequence>
      <attribute name="ResponseID" type="ID" use="required"/>
      <attribute name="InResponseTo" type="NCName" use="optional"/>
      <attribute name="MajorVersion" type="integer" use="required"/>
      <attribute name="MinorVersion" type="integer" use="required"/>
      <attribute name="IssueInstant" type="dateTime" use="required"/>
      <attribute name="Recipient" type="anyURI" use="optional"/>
</complexType>
```

3.2.2.1 Element <Status>

The <Status> element contains the following elements:

<StatusCode> [Required]

A code representing the status of the corresponding request.

<StatusMessage> [Optional]

A message which MAY be returned to an operator.

<StatusDetail> [Optional]

Additional information concerning an error condition.

The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
      <sequence>
            <element ref="samlp:StatusCode"/>
            <element ref="samlp:StatusMessage" minOccurs="0"/>
            <element ref="samlp:StatusDetail" minOccurs="0"/>
      </sequence>
</complexType>
```

3.2.2.2 Element <StatusCode>

The <StatusCode> element specifies one or more possibly nested, codes representing the status of the corresponding request. The <StatusCode> element has the following element and attribute:

Value [Required]

The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST be provided. The value of the topmost <StatusCode> element MUST be from the top-level list provided in this section.

<StatusCode> [Optional]

A subordinate status code that provides more specific information on an error condition.

The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The local parts of these QNames are as follows:

Success

The request succeeded.

VersionMismatch

The SAML responder could not process the request because the version of the request message was incorrect.

Requester

The request could not be performed due to an error on the part of the requester.

Responder

The request could not be performed due to an error on the part of the SAML responder or SAML authority.

The following second-level status codes are referenced at various places in the specification. Additional second-level status codes MAY be defined in future versions of the SAML specification.

RequestVersionTooHigh

The SAML responder cannot process the request because the protocol version specified in the request message is a major upgrade from the highest protocol version supported by the responder.

RequestVersionTooLow

The SAML responder cannot process the request because the protocol version specified in the request message is too low.

RequestVersionDeprecated

The SAML responder can not process any requests with the protocol version specified in the request.

TooManyResponses

The response message would contain more elements than the SAML responder will return.

RequestDenied

The SAML responder or SAML authority is able to process the request but has chosen not to respond. This status code MAY be used when there is concern about the security context of the request message or the sequence of request messages received from a particular requester.

ResourceNotRecognized

The SAML authority does not wish to support resource-specific attribute queries, or the resource value provided in the request message is invalid or unrecognized.

FederationDoesNotExist

The responding provider does not recognize the federated <NameIdentifier> in the request.

SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT define additional codes in the SAML assertion or protocol namespace.

The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value attribute and have the above semantics only in that context.

The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex type:

```
<element name="StatusCode" type="samlp:StatusCodeType"/>
<complexType name="StatusCodeType">
      <sequence>
            <element ref="samlp:StatusCode" minOccurs="0"/>
      </sequence>
      <attribute name="Value" type="QName" use="required"/>
</complexType>
```

3.2.2.3 Element <StatusMessage>

The <StatusMessage> element specifies a message that MAY be returned to an operator:

The following schema fragment defines the <StatusMessage> element:

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

3.2.2.4 Element <StatusDetail>

The <StatusDetail> element MAY be used to specify additional information concerning an error condition.

The following schema fragment defines the <StatusDetail> element and its **StatusDetailType** complex type:

```
<element name="StatusDetail" type="samlp:StatusDetailType"/>
<complexType name="StatusDetailType">
      <sequence>
            <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
</complexType>
```

3.3 Assertion Query and Request Protocol

This section defines messages and processing rules for requesting existing assertions by reference, artifact, or querying for assertions by subject and statement type.

3.3.1 Element <AssertionIDRequest>

If the requester knows the AssertionID of one or more assertions, the <AssertionIDRequest> message can be used to request that the assertion(s) be returned in a <Response> message. The <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section Element <AssertionIDReference> for more information on this element.

The following schema fragment defines the <AssertionIDRequest> element:

```
<element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
<complexType name="AssertionIDRequestType">
      <complexContent>
            <extension base="samlp:RequestAbstractType">
                  <sequence>
                        <element ref="saml:AssertionIDReference"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

3.3.2 Element <ArtifactRequest>

The `<ArtifactRequest>` message is used to request that one or more assertions be returned in a `<Response>` message by specifying one or more assertion artifacts that represent the assertion(s) being requested. Its use is governed by the specific profile of SAML that is being used; see the SAML specification for bindings and profiles [SAMLBind] for more information on the use of assertion artifacts in profiles.

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

```
<element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>
<complexType name="ArtifactRequestType">
      <complexContent>
            <extension base="samlp:RequestAbstractType">
                  <sequence>
                        <element ref="samlp:AssertionArtifact"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
<element name="AssertionArtifact" type="string"/>
```

3.3.3 Queries

The following sections define the SAML query request messages.

3.3.3.1 Element <SubjectQuery>

The `<SubjectQuery>` message element is an extension point that allows new SAML queries to be defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the `<Subject>` element, and an optional SessionIndex attribute to **RequestAbstractType.**

`SessionIndex` [Optional]

> **If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing subject statements do you have for this subject within the context of the supplied session information?"**

If the `SessionIndex` attribute is present in any defined query,at least one element that extends **SubjectStatementAbstractType** in the set of returned assertions MUST contain an `SessionIndex` attribute that matches the `SessionIndex attribute in the query`. It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.

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:RequestAbstractType">
                  <sequence>
                        <element ref="saml:Subject"/>
                  </sequence>
                  <attribute name="SessionIndex" type="string"
use="optional"/>
            </extension>
      </complexContent>
</complexType>
```

3.3.3.2 Element <AuthenticationQuery>

The `<AuthenticationQuery>` message element is used to make the query "What assertions containing authentication statements are available for this subject?" A successful `<Response>` will contain one or more assertions containing authentication statements.

The `<AuthenticationQuery>` message MUST NOT be used as a request for a new authentication using credentials provided in the request. `<AuthenticationQuery>` is a request for statements about authentication acts that have occurred in a previous interaction between the indicated subject and the Authentication Authority.

This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following attribute:

`AuthenticationMethod` [Optional]

If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject with the supplied authentication method?"

In response to an authentication query, a SAML authority returns assertions with authentication statements as follows:

- Rules given in Section Responses to for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If the `AuthenticationMethod` attribute is present in the query, at least one `<AuthenticationStatement>` element in the set of returned assertions MUST contain an `AuthenticationMethod` attribute that matches the `AuthenticationMethod attribute in the query`. 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>` element and its **AuthenticationQueryType** complex type:

```
<element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
<complexType name="AuthenticationQueryType">
      <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                  <attribute name="AuthenticationMethod" type="anyURI"/>
            </extension>
      </complexContent>
</complexType>
```

3.3.3.3 Element <AttributeQuery>

The <AttributeQuery> element is used to make the query “Return the requested attributes for this subject.” A successful response will be in the form of assertions containing attribute statements. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

Resource [Optional]

If present, specifies that the attribute query is being made in order to evaluate a specific access request relating to the resource. The SAML authority MAY use the resource attribute to establish the scope of the request. It is permitted for this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396] §4.2.

If the resource attribute is specified and the SAML authority does not wish to support resource-specific attribute queries, or if the resource value provided is invalid or unrecognized, then the Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a second-level <StatusCode> value of ResourceNotRecognized.

<AttributeDesignator> [Any Number] (see Section Time Values)

Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no attributes are specified, it indicates that all attributes allowed by policy are requested.

In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- Rules given in Section Responses to for matching against the <Subject> element of the query identify the assertions that may be returned.
- If any <AttributeDesignator> elements are present in the query, they constrain the attribute values returned, as noted above.
- The SAML authority MAY take the Resource attribute into account in further constraining the values returned, as noted above.
- The attribute values returned MAY be constrained by application-specific policy considerations.

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="Resource" type="anyURI" use="optional"/>
            </extension>
      </complexContent>
</complexType>
```

3.3.3.4 Element <AuthorizationDecisionQuery>

The <AuthorizationDecisionQuery> element is used to make the query “Should these actions on this resource be allowed for this subject, given this evidence?” A successful response will be in the form of assertions containing authorization decision statements. This element is of type

AuthorizationDecisionQueryType, which extends **SubjectQueryAbstractType** with the addition of the following elements and attribute:

`Resource` [Required]

A URI reference indicating the resource for which authorization is requested.

`<Action>` [One or More]

The actions for which authorization is requested.

`<Evidence>` [Optional]

A set of assertions that the SAML authority MAY rely on in making its authorization decision.

In response to an authorization decision query, a SAML authority returns assertions with authorization decision statements as follows:

- Rules given in Section Responses to for matching against the `<Subject>` element of the query identify the assertions that may be returned.

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

```
<element name="AuthorizationDecisionQuery"
type="samlp:AuthorizationDecisionQueryType"/>
<complexType name="AuthorizationDecisionQueryType">
      <complexContent>
            <extension base="samlp:SubjectQueryAbstractType">
                  <sequence>
                        <element ref="saml:Action" maxOccurs="unbounded"/>
                        <element ref="saml:Evidence" minOccurs="0"/>
                  </sequence>
                  <attribute name="Resource" type="anyURI" use="required"/>
            </extension>
      </complexContent>
</complexType>
```

3.3.4 Element <Response>

The `<Response>` message element is used when a response consists of a list of zero or more assertions that answer the request. It has the complex type **ResponseType**, which extends **StatusResponseType** by adding the following element:

`<Assertion>` [Any Number]

Specifies an assertion by value. (See Section Element <Assertion> for more information.)

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:StatusResponseType">
                  <sequence>
                        <element ref="saml:Assertion" minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

3.3.4.1 Responses to Queries

In response to a query message, every assertion returned by a SAML authority MUST contain at least one statement whose <saml:Subject> element **strongly matches** the <saml:Subject> element found in the query.

A <saml:Subject> element S1 strongly matches S2 if and only if the following two conditions both apply:

- If S2 includes a name identifier element (any element whose type is derived from **NameIdentifierAbstractType**), then S1 must include an identical name identifier element.
- If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an identical <saml:SubjectConfirmation> element.

If the SAML authority cannot provide an assertion with any statements satisfying the constraints expressed by a query, the <Response> element MUST NOT contain an <Assertion> element and MUST include a <StatusCode> element with value Success. It MAY return a <StatusMessage> element with additional information.

3.4 Federated Name Registration Protocol

When an identity provider and service provider first federate a principal's identity using a <NameIdentifier> element with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated, the identity provider generates an opaque value that serves as the initial name identifier that both the service provider and the identity provider use in referring to the principal when communicating with each other.

Subsequent to federation, the service provider MAY register a different opaque value with the identity provider. This opaque value is an attribute termed the SPProvidedIdentifier. Until the service provider registers a different name, this attribute is omitted from <NameIdentifier> elements referring to the principal.

Either the service provider or the identity provider MAY register a new name identifier for a principal with each other at any time following federation. The name identifiers specified by providers SHOULD be unique across the identity providers with which the principal's identity is federated and SHOULD be unique within the group of name identifiers that have been registered with the identity provider by this service provider.

Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated) can be replaced and set with this protocol; non-federated, encrypted, or transient identifiers MUST NOT be used.

3.4.1 Element <RegisterNameIdentifierRequest>

To register an SPProvidedIdentifier attribute with an identity provider, the service provider sends a <RegisterNameIdentifierRequest> message. The same message may be sent by an identity provider, seeking to change the <NameIdentifier> value stored by the service provider.

The <RegisterNameIdentifierRequest> message SHOULD be signed or otherwise authenticated and integrity protected.

The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

This message has the complex type **RegisterNameIdentifierRequestType**, which extends **RequestAbstractType** and adds the following elements:

`<NameIdentifier>` [Required]

The federated name identifier and associated attributes that specify the principal as currently recognized by the identity and service providers prior to this request.

`<NewIdentifier>` [Required]

The new federated identifier value to be used when communicating with the requesting provider concerning this principal. If the requester is the service provider, the new identifier will appear in subsequent `<NameIdentifier>` elements in the `SPProvidedIdentifier` attribute. If the requester is the identity provider, the new value will appear in subsequent `<NameIdentifier>` elements as the element's value.

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

```
<element name="NewIdentifier" type="string">
<element name="RegisterNameIdentifierRequest"
type="samlp:RegisterNameIdentifierRequestType"/>
<complexType name="RegisterNameIdentifierRequestType">
       <complexContent>
              <extension base="samlp:RequestAbstractType">
                     <sequence>
                            <element ref="saml:NameIdentifier"/>
                            <element ref="samlp:NewIdentifier"/>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
```

3.4.2 Element <RegisterNameIdentifierResponse>

The recipient of a `<RegisterNameIdentifierRequest>` message MUST respond with a `<RegisterNameIdentifierResponse>` message, which is of type **StatusResponseType** with no additional content.

The `<RegisterNameIdentifierResponse>` message SHOULD be signed or otherwise authenticated and integrity protected.

The `<Issuer>` of the response MUST contain the unique identifier of the responding provider, with a `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

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

```
<element name="RegisterNameIdentifierResponse"
type="samlp:StatusResponseType"/>
```

3.4.3 Processing Rules

The recipient MUST validate any signature present on the request or response message. To be considered valid, the signature provided MUST be the signature of the `<Issuer>` contained in the message.

If the request includes a `<NameIdentifier>` for which no federation exists between the service provider and the identity provider, the responding provider MUST respond with a `<samlp:Status>` containing a second-level `<samlp:StatusCode>` of `samlp:FederationDoesNotExist`.

If the service provider requests that its identifier be changed, the identity provider MUST include the <NewIdentifier> element's value as the SPProvidedIdentifier when subsequently communicating to the service provider regarding this principal.

If the identity provider requests that its identifier be changed, the service provider MUST use the <NewIdentifier> element's value as the <NameIdentifier> element value when subsequently communicating with the identity provider regarding this principal.

In either case, the <NameIdentifier> value in the request and its associated SPProvidedIdentifier attribute MUST contain the most recent name identifier information established between the providers for the principal. The NameQualifier attribute MUST contain the unique identifier of the identity provider. If the principal's identity federation is between the identity provider and an affiliation group of which the service provider is a member, then the SPNameQualifier attribute MUST contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the service provider.

Changes to these identifiers may take a potentially significant amount of time to propagate through the systems at both the requester and the responder. Implementations might wish to allow each party to accept either identifier for some period of time following the successful completion of a name identifier change. Not doing so could result in the inability of the principal to access resources.

All other processing rules associated with the underlying request and response messages MUST be observed.

3.5 Federation Termination Protocol

When a principal (or an appropriate agent acting on his or her behalf) terminates an identity federation between a service provider and an identity provider through an interaction with the service provider, the service provider MUST send a <FederationTerminationNotification> message to the identity provider. The service provider is stating that it will no longer accept authentication assertions from the identity provider for the specified principal.

Likewise, when a principal terminates an identity federation through an interaction with the identity provider, the identity provider MUST send a <FederationTerminationNotification> message to the service provider. In this case, the identity provider is stating that it will no longer provide authentication assertions to the service provider for the specified principal.

Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated) can be replaced and set with this protocol; non-federated, encrypted, or transient identifiers MUST NOT be used.

3.5.1 Element <FederationTerminationNotification>

A provider sends a <FederationTerminationNotification> to the provider with which it is terminating a federation. The <FederationTerminationNotification> message SHOULD be signed or otherwise authenticated and integrity protected.

The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

This message has the complex type **FederationTerminationNotificationType**, which extends **RequestAbstractType** and adds the following elements:

<NameIdentifier> [Required]

The federated name identifier and associated attributes that specify the principal as currently recognized by the identity and service providers prior to this request. Format MUST be

urn:oasis:names:tc:SAML:2.0:nameid-format:federated.

The following schema fragment defines the <RegisterNameIdentifierRequest> element and its **RegisterNameIdentifierRequestType** complex type:

```
<element name="FederationTerminationNotification"
type="samlp:FederationTerminationNotificationType"/>
<complexType name="FederationTerminationNotificationType">
      <complexContent>
            <extension base="samlp:RequestAbstractType">
                  <sequence>
                        <element ref="saml:NameIdentifier"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

3.5.2 Element <FederationTerminationResponse>

The recipient of a <FederationTerminationNotification> message MUST respond with a <FederationTerminationResponse> message, which is of type **StatusResponseType** with no additional content.

The <FederationTerminationResponse> message SHOULD be signed or otherwise authenticated and integrity protected.

The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

The following schema fragment defines the <FederationTerminationResponse> element:

```
<element name="FederationTerminationResponse"
type="samlp:StatusResponseType"/>
```

3.5.3 Processing Rules

The recipient MUST validate any signature present on the request or response message. To be considered valid, the signature provided MUST be the signature of the <Issuer> contained in the message.

If the request includes a <NameIdentifier> for which no federation exists between the service provider and the identity provider, the responding provider MUST respond with a <samlp:Status> containing a second-level <samlp:StatusCode> of samlp:FederationDoesNotExist.

Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been terminated. A provider MAY choose to invalidate the session of a user for whom federation has been terminated.

3.6 Single Logout Protocol

The single logout protocol provides a message exchange protocol by which all sessions provided by a particular session authority are near-simultaneously terminated. The single logout protocol is used either when a principal logs out at a session participant or when the principal logs out directly at the session authority. This protocol may also be used to logout a principal due to a timeout. The reason for the logout event may be indicated through the `reason` attribute.

The principal may have established authenticated sessions both with the session authority, and individual session participants, based on authentication assertions supplied by the session authority.

When the Principal invokes the single logout process at a Session participant, the session participant MUST send a `<LogoutRequest>` message to the session authority that provided the authentication service related to that session at the session participant.

When either the principal invokes a logout at the session authority, or a session participant sends a logout request to the session authority specifying that principal, the session authority MUST send a `<LogoutRequest>` message to each session participant to which it provided authentication assertions under its current session with the principal, with the exception of the session participant that sent the `<LogoutRequest>` message to the session authority.

3.6.1 Element <LogoutRequest>

`<NameIdentifier>` [Required]

The federated name identifier and associated attributes that specify the principal as currently recognized by the identity and service providers prior to this request.

`<SessionIndex>` [Optional]

The identifier that indexes this session at the message recipient.

`NotOnOrAfter` [Optional]

The time at which the request expires.

`reason` [Optional]

An indication of the reason for the logout.

```
   <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
   <complexType name="LogoutRequestType">
       <complexContent>
           <extension base="samlp:RequestAbstractType">
               <sequence>
                   <element ref="saml:NameIdentifier"/>
                   <element name="SessionIndex" type="string" minOccurs="0"
maxOccurs="unbounded"/>
               </sequence>
               <attribute name="reason" type=”string” minOccurs=”0”/>
               <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
           </extension>
       </complexContent>
   </complexType>
```

3.6.2 Element <LogoutResponse> (UNFINISHED)

```
<element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

3.6.3 Processing Rules (UNFINISHED)

The `<Issuer>` of either message in this protocol MUST contain the unique identifier of the requesting provider, with a `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

Message recipients MUST validate any signature present on the messages specified in this protocol. To be considered valid, the signature provided must be the signature of the <Issuer> contained in the message.

4 SAML Versioning

The SAML specification set is versioned in two independent ways. Each is discussed in the following sections, along with processing rules for detecting and handling version differences, when applicable. Also included are guidelines on when and why specific version information is expected to change in future revisions of the specification.

When version information is expressed as both a Major and Minor version, it may be expressed discretely, or in the form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

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

4.1 SAML Specification Set Version

Each release of the SAML specification set will contain a major and minor version designation describing its relationship to earlier and later versions of the specification set. The version will be expressed in the content and filenames of published materials, including the specification set document(s), and XML schema instance(s). There are no normative processing rules surrounding specification set versioning, since it merely encompasses the collective release of normative specification documents which themselves contain processing rules.

The overall size and scope of changes to the specification set document(s) will informally dictate whether a set of changes constitutes a major or minor revision. In general, if the specification set is backwards compatible with an earlier specification set (that is, valid older messages, protocols, and semantics remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0, described in Section Interoperability with SAML V1.0.

4.1.1 Schema Version

As a non-normative documentation mechanism, any XML schema instances published as part of the specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the specification set version in which it has been published. Validating implementations MAY use the attribute as a means of distinguishing which version of a schema is being used to validate messages, or to support a multiplicity of versions of the same logical schema.

4.1.2 SAML Assertion Version

The SAML `<Assertion>` element contains attributes for expressing the major and minor version of the assertion using a pair of integers. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the assertions of the same version. That is, specification set version 1.0 describes assertion version 1.0, and so on.

There is explicitly NO relationship between the assertion version and the SAML assertion XML namespace that contains the schema definitions for that assertion version.

The following processing rules apply:

- A SAML authority MUST NOT issue any assertion with an assertion version number not supported by the authority.
- A SAML relying party MUST NOT process any assertion with a major assertion version number not supported by the relying party.

- A SAML relying party MAY process or MAY reject an assertion whose minor assertion version number is higher than the minor assertion version number supported by the relying party. However, all assertions that share a major assertion version number MUST share the same general processing rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0 assertion without ill effect.

4.1.3 SAML Protocol Version

The SAML protocol `<Request>` and `<Response>` elements contain attributes for expressing the major and minor version of the request or response message using a pair of integers. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the protocol messages of the same version. That is, specification set version 1.0 describes request and response version V1.0, and so on.

There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace that contains the schema definitions for protocol messages for that protocol version.

The version numbers used in SAML protocol `<Request>` and `<Response>` elements will be the same for any particular revision of the SAML specification set.

4.1.3.1 Request Version

The following processing rules apply to requests:

- A SAML requester SHOULD issue requests with the highest request version supported by both the SAML requester and the SAML responder.
- If the SAML requester does not know the capabilities of the SAML responder, then it should assume that it supports requests with the highest request version supported by the requester.
- A SAML requester MUST NOT issue a request message with a request version number matching a response version number that the requester does not support.
- A SAML responder MUST reject any request with a major request version number not supported by the responder.
- A SAML responder MAY process or MAY reject any request whose minor request version number is higher than the highest supported request version that it supports. However, all requests that share a major request version number MUST share the same general processing rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

4.1.4 Response Version

The following processing rules apply to responses:

- A SAML responder MUST NOT issue a response message with a response version number higher than the request version number of the corresponding request message.
- A SAML responder MUST NOT issue a response message with a major response version number lower than the major request version number of the corresponding request message except to report the error `RequestVersionTooHigh`.

An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-level `<StatusCode>` value of `VersionMismatch`, and MAY result in reporting one of the following

second-level values: `RequestVersionTooHigh`, `RequestVersionTooLow`, or `RequestVersionDeprecated`.

4.1.5 Permissible Version Combinations

In general, assertions of a particular major version may appear in response messages of the same major version, as permitted by the importation of the SAML assertion namespace into the SAML protocol schema. Future versions of this specification are expected to explicitly describe the permitted combinations across major versions.

Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to appear in a V1.1 response message.

4.2 SAML Namespace Version

XML schema instances and "qualified names" (QNames) published as part of the specification set contain one or more target namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and syntactical definitions that make up that part of the specification.

The namespace URIs defined by the specification set will generally contain version information of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to the major and minor version of the specification set in which the namespace is first introduced and defined. This information is not typically consumed by an XML processor, which treats the namespace opaquely, but is intended to communicate the relationship between the specification set and the namespaces it defines.

As a general rule, implementers can expect the namespaces (and the associated schema definitions) defined by a major revision of the specification set to remain valid and stable across minor revisions of the specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but this is expected to be rare. In such cases, the older namespaces and their associated definitions should be expected to remain valid until a major specification set revision.

4.2.1 Schema Evolution

In general, maintaining namespace stability while adding or changing the content of a schema are competing goals. While certain design strategies can facilitate such changes, it is complex to predict how older implementations will react to any given change, making forward compatibility difficult to achieve. Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace stability. Except in special circumstances (for example to correct major deficiencies or fix errors), implementations should expect forward compatible schema changes in minor revisions, allowing new messages to validate against older schemas.

Implementations SHOULD expect and be prepared to deal with new extensions and message types in accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types that leverage the extension facilities described in Section SAML Extensions. Older implementations SHOULD reject such extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples include new query, statement, or condition types.

5 SAML and XML Signature Syntax and Processing

SAML assertions and SAML protocol request and response messages may be signed, with the following benefits:

- An assertion signed by the SAML authority supports:
  - Assertion integrity.
  - Authentication of the SAML authority to a SAML relying party.
  - If the signature is based on the SAML authority's public-private key pair, then it also provides for non-repudiation of origin.
- A SAML protocol request or response message signed by the message originator supports:
  - Message integrity.
  - Authentication of message origin to a destination.
  - If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.

A digital signature is not always required in SAML. For example, it may not be required in the following situations:

- In some circumstances signatures may be "inherited," such as when an unsigned assertion gains protection from a signature on the containing protocol response message. "Inherited" signatures should be used with care when the contained object (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context must be retained to allow validation, exposing the XML content and adding potentially unnecessary overhead.
- The SAML relying party or SAML requester may have obtained an assertion or protocol message from the SAML authority or SAML responder directly (with no intermediaries) through a secure channel, with the SAML authority or SAML responder having authenticated to the relying party or SAML responder by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition, the applicable security requirements depend on the communicating applications and the nature of the assertion or message transported.

It is recommended that, in all other contexts, digital signatures be used for assertions and request and response messages. Specifically:

- A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority SHOULD be signed by the SAML authority.
- A SAML protocol message arriving at a destination from an entity other than the originating site SHOULD be signed by the origin site.

Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are intended to be the primary SAML signature mechanism, but the specification attempts to ensure compatibility with profiles that may require other mechanisms.

Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST be used if signing.

5.1 Signing Assertions

All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema as described in Section Assertions.

5.2 Request/Response Signing

All SAML protocol request and response messages MAY be signed using the XML Signature. This is reflected in the schema as described in Sections Requests and Responses and .

5.3 Signature Inheritance

A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>` or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children, then the assertion can be considered to inherit the signature from the enclosing element. The resulting interpretation should be equivalent to the case where the assertion itself was signed with the same key and signature options.

Many SAML use cases involve SAML XML data enclosed within other protected data structures such as signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may define additional rules for interpreting SAML elements as inheriting signatures or other authentication information from the surrounding context, but no such inheritance should be inferred unless specifically identified by the profile.

5.4 XML Signature Profile

The XML Signature specification calls out a general XML syntax for signing data with flexibility and many choices. This section details the constraints on these facilities so that SAML processors do not have to deal with the full generality of XML Signature processing. This usage makes specific use of the **xsd:ID**-typed attributes optionally present on the root elements to which signatures can apply: the `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section as the identifier attributes.

5.4.1 Signing Formats and Algorithms

XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and detached.

SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol messages. SAML processors SHOULD support the use of RSA signing and verification for public key operations in accordance with the algorithm identified by http://www.w3.org/2000/09/xmldsig#rsa-sha1.

5.4.2 References

Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element may or may not be the root element of the actual XML document containing the signed assertion or message.

Signatures MUST contain a single <ds:Reference> containing a URI reference to the identifier attribute value of the root element of the message being signed. For example, if the attribute value is "foo", then the URI attribute in the <ds:Reference> element MUST be "#foo".

5.4.3 Canonicalization Method

SAML implementations SHOULD use Exclusive Canonicalization , with or without comments, both in the <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform> algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages embedded in an XML context can be verified independent of that context.

5.4.4 Transforms

Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature transform (with the identifier http://www.w3.org/2000/09/xmldsig#enveloped-signature) or the exclusive canonicalization transforms (with the identifier http://www.w3.org/2001/10/xml-exc-c14n# or http://www.w3.org/2001/10/xml-exc-c14n#WithComments).

Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by applying the transforms manually to the content and reverifying the result as consisting of the same SAML message.

5.4.5 KeyInfo

XML Signature defines usage of the <ds:KeyInfo> element. SAML does not require the use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY be absent.

5.4.6 Binding Between Statements in a Multi-Statement Assertion

Use of signing does not affect semantics of statements within assertions in any way, as stated in Section SAML Assertions.

5.4.7 Interoperability with SAML V1.0

The use of XML Signature described above is incompatible with the usage described in the SAML V1.0 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to ensure interoperability. It was constrained by the inability to use URI references to identify the SAML content to be signed. With this limitation removed by the addition of SAML identifier attributes, a decision has been made to forgo backwards compatibility with the older specification in this respect.

5.4.8 Example

Following is an example of a signed response containing a signed assertion. Line breaks have been added for readability; the signatures are not valid and cannot be successfully verified.

```
<Response
  IssueInstant="2003-04-17T00:46:02Z"
  MajorVersion="1"
  MinorVersion="1"
  Recipient="www.opensaml.org"
```

```
  ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"
  xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ds:Signature
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod
  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod
  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference
  URI="#_c7055387-af61-4fce-8b98-e2927324b306">
<ds:Transforms>
<ds:Transform
  Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:Transform
  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<InclusiveNamespaces
  PrefixList="#default saml samlp ds xsd xsi"
  xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod
  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg
LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+
c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
<Status><StatusCode Value="samlp:Success"/></Status>
<Assertion
  AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
  IssueInstant="2003-04-17T00:46:02Z"
  Issuer="www.opensaml.org"
  MajorVersion="1"
  MinorVersion="1"
  xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Conditions
  NotBefore="2003-04-17T00:46:02Z"
```

```
  NotOnOrAfter="2003-04-17T00:51:02Z">
<AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
</AudienceRestrictionCondition></Conditions>
<AuthenticationStatement
  AuthenticationInstant="2003-04-17T00:46:00Z"
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
<Subject>
<NameIdentifier
  Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
scott@example.org</NameIdentifier>
<SubjectConfirmation>
<ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
</SubjectConfirmation></Subject>
<SubjectLocality
  IPAddress="127.0.0.1"/>
</AuthenticationStatement>
<ds:Signature
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod
  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod
  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference
  URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
<ds:Transforms>
<ds:Transform
  Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:Transform
  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<InclusiveNamespaces
  PrefixList="#default saml samlp ds xsd xsi"
  xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod
  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
MWuL/cBUj2OtBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
MRIwEAYDVQQIEwlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
F1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
bmZvcm1hdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgQ0Eg
LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
CzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
Ym9yMQ4wDAYDVQQKEwVVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
dTEnMCUGCSqGSIb3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldDIuZWR1MIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIaOAPSZBl13R6+KYiE7x4XAWIrCP+
c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
8I3bsbmRAUg4UP9hH6ABVq4KQKMknxu1xQxLhpR1ylGPdiowMNTrEG8cCx3w/w==
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</ds:Signature></Assertion></Response>
```

6 SAML Extensions

The SAML schemas support extensibility. An example of an application that extends SAML assertions is the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use the extensibility features in SAML to create extension schemas.

Note that elements in the SAML schemas are blocked from substitution, which means that no SAML elements can serve as the head element of a substitution group. However, SAML types are not defined as `final`, so that all SAML types MAY be extended and restricted. The following sections discuss only elements and types that have been specifically designed to support extensibility.

6.1 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`, and are thus usable only as the base of a derived type:

- `<Condition>`
- `<Statement>`
- `<SubjectStatement>`

The following elements that are directly usable as part of SAML MAY be extended:

- `<AuthenticationStatement>`
- `<AuthorizationDecisionStatement>`
- `<AttributeStatement>`
- `<AudienceRestrictionCondition>`

The following elements are defined to allow elements from arbitrary namespaces within them, which serves as a built-in extension point without requiring an extension schema:

- `<AttributeValue>`
- `<Advice>`

6.2 Protocol Schema Extension

The following SAML protocol elements are intended specifically for use as extension points in an extension schema; their types are set to `abstract`, and are thus usable only as the base of a derived type:

- `<Query>`
- `<SubjectQuery>`

The following elements that are directly usable as part of SAML MAY be extended:

- `<Request>`

- `<AuthenticationQuery>`
- `<AuthorizationDecisionQuery>`
- `<AttributeQuery>`
- `<Response>`

7 SAML-Defined Identifiers

The following sections define URI-based identifiers for common authentication methods, resource access actions, and subject name identifier formats.

Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of the most current RFC that specifies the protocol is used. URI references created specifically for SAML have one of the following stems:

```
urn:oasis:names:tc:SAML:1.0:
urn:oasis:names:tc:SAML:1.1:
```

7.1 Authentication Method Identifiers

The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although both can refer to the same underlying mechanisms. An authentication statement with an `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The `AuthenticationMethod` attribute indicates how that authentication was done. Note that the authentication statement does not provide the means to perform that authentication, such as a password, key, or certificate.

In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element, which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML relying party to confirm that the request or message came from a system entity that corresponds to the subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method that the relying party can use to do this in the future. This may or may not have any relationship to an authentication that was performed previously. Unlike the authentication method, the subject confirmation method may be accompanied by some piece of information, such as a certificate or key, that will allow the relying party to perform the necessary check.

Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML bindings and profiles specification [SAMLBind] for more information. Additional methods may be added by defining new profiles or by private agreement.

The following identifiers refer to SAML-specified authentication methods.

7.1.1 Password

URI: urn:oasis:names:tc:SAML:1.0:am:password

The authentication was performed by means of a password.

7.1.2 Kerberos

URI: urn:ietf:rfc:1510

The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the Needham-Schroeder symmetric key authentication mechanism [Needham78].

7.1.3 Secure Remote Password (SRP)

URI: urn:ietf:rfc:2945

The authentication was performed by means of Secure Remote Password protocol as specified in [RFC 2945].

7.1.4 Hardware Token

URI: urn:oasis:names:tc:SAML:1.0:am:HardwareToken

The authentication was performed using some (unspecified) hardware token.

7.1.5 SSL/TLS Certificate Based Client Authentication:

URI: urn:ietf:rfc:2246

The authentication was performed using either the SSL or TLS protocol with certificate-based client authentication. TLS is described in [RFC 2246].

7.1.6 X.509 Public Key

URI: urn:oasis:names:tc:SAML:1.0:am:X509-PKI

The authentication was performed by some (unspecified) mechanism on a key authenticated by means of an X.509 PKI [X.500][PKIX]. It may have been one of the mechanisms for which a more specific identifier has been defined below.

7.1.7 PGP Public Key

URI: urn:oasis:names:tc:SAML:1.0:am:PGP

The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific identifier has been defined below.

7.1.8 SPKI Public Key

URI: urn:oasis:names:tc:SAML:1.0:am:SPKI

The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has been defined below.

7.1.9 XKMS Public Key

URI: urn:oasis:names:tc:SAML:1.0:am:XKMS

The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific identifier has been defined below.

7.1.10 XML Digital Signature

URI: urn:ietf:rfc:3075

The authentication was performed by means of an XML digital signature **[RFC 3075]**.

7.1.11 Unspecified

URI: urn:oasis:names:tc:SAML:1.0:am:unspecified

The authentication was performed by an unspecified means.

7.2 Action Namespace Identifiers

The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see Section Element <Action>) to refer to common sets of actions to perform on resources.

7.2.1 Read/Write/Execute/Delete/Control

URI: urn:oasis:names:tc:SAML:1.0:action:rwedc

Defined actions:

```
Read Write Execute Delete Control
```

These actions are interpreted as follows:

`Read`

The subject may read the resource.

`Write`

The subject may modify the resource.

`Execute`

The subject may execute the resource.

`Delete`

The subject may delete the resource.

`Control`

The subject may specify the access control policy for the resource.

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

URI: urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

Defined actions:

```
Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control
```

The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action `~Read` is affirmatively denied read permission.

A SAML authority MUST NOT authorize both an action and its negated form.

7.2.3 Get/Head/Put/Post

URI: urn:oasis:names:tc:SAML:1.0:action:ghpp

Defined actions:

```
GET HEAD PUT POST
```

These actions bind to the corresponding HTTP operations. For example a subject authorized to perform the `GET` action on a resource is authorized to retrieve it.

The `GET` and `HEAD` actions loosely correspond to the conventional read permission and the `PUT` and `POST` actions to the write permission. The correspondence is not exact however since an HTTP GET operation may cause data to be modified and a POST operation may cause modification to a resource other than the one specified in the request. For this reason a separate Action URI reference specifier is provided.

7.2.4 UNIX File Permissions

URI: urn:oasis:names:tc:SAML:1.0:action:unix

The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

The action string is a four-digit numeric code:

extended user group world

Where the *extended* access permission has the value

+2 if sgid is set

+4 if suid is set

The *user group* and *world* access permissions have the value

+1 if execute permission is granted

+2 if write permission is granted

+4 if read permission is granted

For example, `0754` denotes the UNIX file access permission: user read, write and execute; group read and execute; and world read.

7.3 NameIdentifier Format Identifiers

The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see Section) to refer to common formats for the content of the `<NameIdentifier>` element and the associated processing rules, if any.

Note: Several identifiers that were deprecated in V1.1 have been removed for V2.0 of SAML.

7.3.1 Unspecified

URI: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

The interpretation of the content of the element is left to individual implementations.

7.3.2 Email Address

URI: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

7.3.3 X.509 Subject Name

URI: urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

Indicates that the content of the element is in the form specified for the contents of the `<ds:X509SubjectName>` element in the XML Signature Recommendation . Implementors should note that the XML Signature specification specifies encoding rules for X.509 subject names that differ from the rules given in IETF RFC 2253 [RFC 2253].

7.3.4 Windows Domain Qualified Name

URI: urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

Indicates that the content of the element is a Windows domain qualified name. A Windows domain qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator MAY be omitted.

7.3.5 Provider Identifier

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:provider

Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser profiles). Such an identifier can be used to make assertions about system entities that can issue SAML requests, responses, and assertions.

7.3.6 Federated Identifier

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:federated

Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity federation between an identity provider and a service provider (or affiliation of service providers). Federated name identifiers generated by identity providers MUST be constructed using pseudo-random values that have no discernible correspondence with the subject's actual identifier (for example, username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

The element's content MUST contain the most recent identifier of the subject set by the identity provider.

The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider participating in the identity federation. It MAY be omitted if the value can be derived from the context of the message containing the element, such as the issuer of an assertion.

The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider or affiliation of providers participating in the identity federation. It MAY be omitted if the element is

contained in a message intended only for consumption directly by the service provider, and the value would be the name of that service provider.

The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject most recently set by the service provider or affiliation, if any. If no such identifier has been established, than the attribute MUST be omitted.

Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear text with providers other than the providers that have established the identity federation. Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and protections. Deployments without such requirements are free to use other kinds of identifiers in their SAML exchanges.

7.3.7 Transient Identifier

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:transient

Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of 256 characters.

The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier represents a transient and temporary identity federation, as described in Section Federated Identifier. In such a case, they MAY be omitted in accordance with the rules specified in that section.

8 References

The following works are cited in the body of this specification.

8.1 Normative References

[ExcI-C14N] J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web Consortium, July 2002. http://www.w3.org/TR/xml-exc-c14n/.

[Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures.* World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/. Note that this specification normatively references [Schema2], listed below.

[Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-2/.

[XML] T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. http://www.w3.org/TR/REC-xml.

[XMLEnc] D. Eastlake et al., XML Encryption Syntax and Processing, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, World Wide Web Consortium. Note that this specification normatively references , listed below.

[XMLEnc-XSD] XML Encryption Schema. World Wide Web Consortium. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd.

[XMLNS] T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January 1999. http://www.w3.org/TR/REC-xml-names.

[XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium, February 2002. http://www.w3.org/TR/xmldsig-core/. Note that this specification normatively references [XMLSig-XSD], listed below.

[XMLSig-XSD] XML Signature Schema. World Wide Web Consortium. http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd.

8.2 Non-Normative References

[LibertyProt] J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty Alliance Project, January 2003, http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.

[Needham78] 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.

[PGP] Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*. IETF RFC 1991, August 1996. http://www.ietf.org/rfc/rfc1991.txt.

[PKIX] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF RFC 2459, January 1999. http://www.ietf.org/rfc/rfc2459.txt.

[RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5).* IETF RFC 1510, September 1993. http://www.ietf.org/rfc/rfc1510.txt.

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt.

[RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. http://www.ietf.org/rfc/rfc2253.txt.

[RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax.* IETF RFC 2396, August, 1998. http://www.ietf.org/rfc/rfc2396.txt.

[RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999. http://www.ietf.org/rfc/rfc2630.txt.

[RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. http://www.ietf.org/rfc/rfc2822.txt.

[RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System.* IETF RFC 2945, September 2000. http://www.ietf.org/rfc/rfc2945.txt.

[RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF 3075, March 2001. http://www.ietf.org/rfc/rfc3075.txt.

[SAMLBind] E. Maler et al. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-1.1. http://www.oasis-open.org/committees/security/.

[SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML).* OASIS, September 2003. Document ID oasis-sstc-saml-conform-1.1. HYPERLINK "http://www.oasis-open.org/committees/security/"http://www.oasis-open.org/committees/security/.

[SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. OASIS, November 2002. http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf.

[SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1. HYPERLINK "http://www.oasis-open.org/committees/security/"http://www.oasis-open.org/committees/security/.

[SAMLP-XSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID oasis-sstc-saml-schema-protocol-1.1. HYPERLINK "http://www.oasis-open.org/committees/security/"http://www.oasis-open.org/committees/security/.

[SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-sec-consider-1.1. HYPERLINK "http://www.oasis-open.org/committees/security/"http://www.oasis-open.org/committees/security/.

[SAML-XSD] E. Maler et al. *SAML assertion schema.* OASIS, September 2003. Document ID oasis-sstc-saml-schema-assertion-1.1. HYPERLINK "http://www.oasis-open.org/committees/security/"http://www.oasis-open.org/committees/security/.

[SPKI] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. *SPKI Certificate Theory*. IETF RFC 2693, September 1999. http://www.ietf.org/rfc/rfc2693.txt.

[UNICODE-C] M. Davis, M. J. Dürst. *Unicode Normalization Forms.* UNICODE Consortium, March 2001. http://www.unicode.org/unicode/reports/tr15/tr15-21.html.

[W3C-CHAR] M. J. Dürst. *Requirements for String Identity Matching and String Indexing.* World Wide Web Consortium, July 1998. http://www.w3.org/TR/WD-charreq.

[W3C-CharMod] M. J. Dürst. *Character Model for the World Wide Web 1.0.* World Wide Web Consortium, April, 2002. http://www.w3.org/TR/charmod/.

2295 **[X.500]** ITU-T Recommendation X.501: Information Technology - Open Systems
2296 Interconnection - The Directory: Models. 1993.

2297 **[XKMS]** W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J.
2298 Lapp. XML Key Management Specification (XKMS). W3C Note 30 March 2001.
2299 http://www.w3.org/TR/xkms/.

Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- @@

2304 # Appendix B. Revision History

Rev	Date	By Whom	What
01	20 Oct 2003	Eve Maler	Initial draft. Converted to OpenOffice. **CORE-1** through **CORE-4**. Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed.
02	4 Jan 2004	Eve Maler	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal (http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587) for work item **W-2**, Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet). Fixed **CORE-10** (the description of subelement occurrence in the `<Evidence>` element).
03	24 Jan 2004	Scott Cantor	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars.
04	1 Feb 2004	Eve Maler	Made minor edits to new and existing material; changed new <AssertionRequest> element name to <AssertionIDRequest>; changed new <AssertionArtifact> and <NewIdentifier> element declarations from local to global; made distinction between normative and non-normative references; implemented the blocking of element substitution. The bulk of work item W-2, Identity Federation, is now reflected here. What remains is the federation termination protocol, plus a few other pieces that are covered under other work items.
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol, removed NameID date attributes, clarified Name Reg processing rules, added Extensions facility and Consent attribute. Also moved Signature on assertions to a location consistent with Request and Response. Added session protocol material; still unfinished.

2305

Appendix C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.