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

Last-Call Working Draft 17, 13 July 2004

Document identifier:

sstc-saml-core-2.0-draft-17

Location:

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

Editors:

Scott Cantor, Internet2 (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
John Hughes, Entegrity
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Bob Blakley, IBM
Marlena Erdos, IBM
RL “Bob” Morgan, Internet2
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Prateek Mishra, Netegrity (co-chair)
Charles Knouse, Oblix
Simon Godik, Overxeer
John Linn, RSA Security
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 convey this information.

Status:

This is a last-call working draft produced by the Security Services Technical Committee. **See the Revision History for details of changes made in this revision.**

Comments on this last-call draft are solicited by **2 August 2004** so that the TC can subsequently prepare an OASIS Committee Draft. Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them by filling out the web form located at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. 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 [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 [SAMLBind] provides frameworks for this embedding and transport. Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMLP-XSD] are available. For general explanations of SAML terms and concepts, refer to the SAML technical overview [SAML-TechOvw] and the SAML glossary [SAMLGloss].

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

1.1 Notation

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

The 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 schema listings in this specification, the schema documents take precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.

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

Prefix	XML Namespace	Comments
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace, defined in a schema [SAML-XSD]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
`samlp:`	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace, defined in a schema [SAMLP-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
`ds:`	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].
`xenc:`	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].

Prefix	XML Namespace	Comments
xsd:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. The prefix is generally shown in mentions of XML Schema-related constructs in text, however.

This specification uses the following typographical conventions in text: <SAMLElement>, <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

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 [XMLSig], which is associated with the following XML namespace:

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

See Section 4.2 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 [Schema2]. All strings in SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the XML Recommendation [XML] §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 [Schema2], and MUST be expressed in UTC form, with no time zone component.

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. Such a pseudorandom generator MUST be seeded with unique material in order to insure the desired uniqueness properties between different systems.

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 identifier reference 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 in this specification or particular profiles, 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 whitespace, or conversion of locale-specific formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft 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 [XML] §2.11. Attribute values defined as strings (or types derived from strings) are normalized as described in [XML] §3.3.3. All whitespace characters are replaced with blanks (ASCII code 32_{Decimal}).

The SAML specification does not define collation or sorting order for attribute values or element content. 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 described in Section 7, 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
	…
	</documentation>
	</annotation>
…
</schema>
```

2.2 Name Identifiers

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

2.2.1 Element <BaseID>

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

NameQualifier [Optional]

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

SPNameQualifier [Optional]

> Further qualifies an identifier with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate identifiers on the basis of the relying party or parties.

The following schema fragment defines the <BaseID> element and its **BaseIDType** complex type:

```
<element name="BaseID" type="saml:BaseIDAbstractType"/>
<complexType name="BaseIDAbstractType" abstract="true" mixed="true">
    <complexContent>
        <extension base="anyType">
            <attribute name="NameQualifier" type="string" use="optional"/>
            <attribute name="SPNameQualifier" type="string" use="optional"/>
       </extension>
    </complexContent>
</complexType>
```

2.2.2 Element <NameID>

The <NameID> element is of type **NameIDType**, which restricts **BaseIDAbstractType** to simple string content that contains the name identifier itself, that which provides additional attributes as follows:

Format [Optional]

> A URI reference representing the classification of string-based identifier information. See Section 8.3 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 8.3.1) is in effect.
>
> When a Format value other than those specified in Section 8.3 is used, the content of the <NameID> element is to be interpreted according to the definition of that format as provided outside of this specification. If not otherwise indicated by the definition of the format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties are implementation-specific.

SPProvidedID [Optional]

> A 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 <NameID> element. This attribute provides a means of integrating the use of SAML with existing identifiers already in use by a service provider.

The following schema fragment defines the <NameID> element and its **NameIDType** complex type:

```
<element name="NameID" type="saml:NameIDType"/>
<complexType name="NameIDType" mixed="false">
```

```
    <simpleContent>
        <restriction base="saml:BaseIDAbstractType">
            <simpleType>
                <restriction base="string"/>
            </simpleType>
            <attribute name="Format" type="anyURI" use="optional"/>
            <attribute name="SPProvidedID" type="string" use="optional"/>
        </restriction>
    </simpleContent>
</complexType>
```

2.2.3 Element <EncryptedID>

The <EncryptedID> element extends **BaseIDAbstractType** to carry the content of the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. Its <EncryptedID> element contains the following elements:

<xenc:EncryptedData> [Required]

> The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The Type attribute SHOULD be present and, if present, MUST contain a value of http://www.w3.org/2001/04/xmlenc#Element. The encrypted content MUST contain an element that has a type that is derived from **BaseIDAbstractType** or from **AssertionType**.

<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. The value of the Recipient attribute SHOULD be the URI identifier of a SAML system entity, as defined by Section 8.3.6.

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 [XMLEnc] §6.3.

The following schema fragment defines the <EncryptedID> element and its **EncryptedIDType** complex type:

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

2.2.4 Element <Issuer>

The <Issuer> element, with complex type **NameIDType**, provides 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 pieces of descriptive data. If no Format value is provided, the identifier urn:oasis:names:tc:SAML:2.0:nameid-format:entity is in effect.

The following schema fragment defines the <Issuer> element:

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

2.3 Assertions

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

2.3.1 Element <AssertionIDRef>

The <AssertionIDRef> element makes a reference to a SAML assertion by its unique identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is not specified as part of the reference.

The following schema fragment defines the <AssertionIDRef> element:

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

2.3.2 Element <AssertionURIRef>

The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. Resolving the URI reference (in a fashion dictated by the URI reference itself) is intended to produce the assertion. See the Bindings specification [SAMLBind] for information on how this element is used in a protocol binding.

The following schema fragment defines the <AssertionURIRef> element:

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

2.3.3 Element <Assertion>

The <Assertion> element is of the **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 4.

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 4.

ID [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 1.2.2.

<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.
>
> 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 by specific profiles are application-specific.

<ds:Signature> [Optional]

An XML Signature that authenticates the assertion, as described in Section 5.

<Subject> [Optional]

The subject of the statement(s) in the assertion.

<Conditions> [Optional]

Conditions that MUST be taken into account in assessing the validity of and/or using 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.

Zero or more of the following statement elements:

<Statement>

A statement defined in an extension schema.

<AuthnStatement>

An authentication statement.

<AuthzDecisionStatement>

An authorization decision statement.

<AttributeStatement>

An attribute statement.

An assertion with no statements MUST contain a <Subject> element. Such an assertion identifies a principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further information associated with that principal.

Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If omitted, then the statements in the assertion are assumed to identify (implicitly or explicitly) the subject or subjects to which they apply in an application- or profile-specific manner.

If a <ds:Signature> element is present, a relying party SHOULD verify that the signature is valid. If it is invalid, the relying party SHOULD NOT rely on the contents of the assertion.

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:Subject" minOccurs="0"/>
                <element ref="saml:Conditions" minOccurs="0"/>
                <element ref="saml:Advice" minOccurs="0"/>
                <choice minOccurs="0" maxOccurs="unbounded">
                        <element ref="saml:Statement"/>
                        <element ref="saml:AuthnStatement"/>
                        <element ref="saml:AuthzDecisionStatement"/>
                        <element ref="saml:AttributeStatement"/>
                </choice>
        </sequence>
        <attribute name="MajorVersion" type="integer" use="required"/>
        <attribute name="MinorVersion" type="integer" use="required"/>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

2.3.3.1 Element <Subject>

The optional <Subject> element specifies the principal that is the subject of all of the (zero or more) statements in the assertion. It contains a name identifier, a series of one or more subject confirmations, or both:

<BaseID>, <NameID>, or <EncryptedID> [Optional]

Identifies the subject.

<SubjectConfirmation> [Zero or More]

Information that allows the subject to be confirmed. If more than one subject confirmation is provided, then usage of any one of them is sufficient to confirm the subject for the purpose of applying the assertion.

If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it can treat the entity presenting the assertion to the relying party as the entity that the SAML authority associates with the name identifier for the purposes of processing the assertion. 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:BaseID"/>
                                <element ref="saml:NameID"/>
                                <element ref="saml:EncryptedID"/>
                        </choice>
                        <element ref="saml:SubjectConfirmation" minOccurs="0"
maxOccurs="unbounded"/>
                </sequence>
                <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
        </choice>
</complexType>
```

2.3.3.2 Element <SubjectConfirmation>

The <SubjectConfirmation> element provides the means for a relying party to verify the correspondence of the subject of the assertion with the party with whom the relying party is communicating. It contains the following attributes and elements:

Method [Required]

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

<SubjectConfirmationData> [Optional]

Additional confirmation information to be used by a specific confirmation method. For example, typical content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax and Processing specification [XMLSig], which identifies a cryptographic key. Particular confirmation methods MAY define a schema type to describe the elements, attributes, or content that may appear in the <SubjectConfirmationData> element.

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

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
        <sequence>

                <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
        </sequence>
        <attribute name="Method" type="anyURI" use="required"/>
</complexType>

```

2.3.3.3 Element <SubjectConfirmationData>

The `<SubjectConfirmationData>` element has the **SubjectConfirmationDataType** complex type. It specifies additional data that allows the subject to be confirmed or constrains the circumstances under which the confirmation can take place. It contains the following optional attributes that can apply to any method:

`NotBefore` [Optional]

A time instant before which the subject cannot be confirmed.

`NotOnOrAfter` [Optional]

A time instant at which the subject can no longer be confirmed.

`Recipient` [Optional]

Specifies the entity or location to which an entity can present the assertion while confirming itself.

`InResponseTo` [Optional]

Specifies the `RequestID` of a SAML protocol message in response to which an entity can present the assertion while confirming itself.

`Address` [Optional]

Specifies the network address from which an entity can present the assertion while authenticating itself.

Particular confirmation methods MAY require the use of one or more of these attributes. Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if any, SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore` MUST be less than (earlier than) the value for `NotOnOrAfter`.

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

```
<element name="SubjectConfirmationData"
type="saml:SubjectConfirmationDataType"/>
<complexType name="SubjectConfirmationDataType" mixed="true">
        <complexContent>
                <extension base="anyType">
                        <attribute name="NotBefore" type="dateTime"
use="optional"/>
                        <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
                        <attribute name="Recipient" type="anyURI" use="optional"/>
                        <attribute name="InResponseTo" type="NCName"
use="optional"/>
                        <attribute name="Address" type="string" use="optional"/>
                </extension>
        </complexContent>
</complexType>
```

2.3.3.4 Complex Type KeyInfoConfirmationDataType

The **KeyInfoConfirmationDataType** complex type constrains a <SubjectConfirmationData> element to contain one or more <ds:KeyInfo> elements that identify cryptographic keys that are used in some way to authenticate the subject. The particular confirmation method MUST define the exact mechanism by which the confirmation data can be used.

This complex type SHOULD be used by any confirmation method that defines its confirmation data in terms of the <ds:KeyInfo> element.

Note that in accordance with [XMLSig], each <ds:KeyInfo> element MUST identify a single cryptographic key. Multiple keys MAY be identified with separate <ds:KeyInfo> elements, such as when a principal uses different keys to confirm itself to different relying parties

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

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">
      <complexContent>
            <restriction base="saml:SubjectConfirmationDataType">
                  <sequence>
                        <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
                  </sequence>
            </restriction>
      </complexContent>
</complexType>
```

2.3.3.5 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 1.2.2.

NotOnOrAfter [Optional]

Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as described in Section 1.2.2.

<Condition> [Any Number]

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

<AudienceRestriction> [Any Number]

Specifies that the assertion is addressed to a particular audience.

<OneTimeUse**>** [Optional]

Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future use. Although the schema permits multiple occurrences, there MUST be at most one instance of this element.

<ProxyRestriction> [Optional]

Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent assertions of their own on the basis of the information contained in the original assertion. Although the schema permits multiple occurrences, there MUST be at most one instance of this element.

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

```
<element name="Conditions" type="saml:ConditionsType"/>
```

```
<complexType name="ConditionsType">
        <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:Condition"/>
                <element ref="saml:AudienceRestriction"/>
                <element ref="saml:OneTimeUse">
                <element ref="saml:ProxyRestriction"/>
        </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, using the following rules in the order shown:

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 considered to be ***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 considered 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 considered to be ***Indeterminate*** in accordance with rule 3 above.

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

Also note that some conditions may not directly impact the validity of the containing assertion (they always evaluate to ***Valid***), but may restrict the behavior of relying parties with respect to the use of the assertion.

2.3.3.5.1 Attributes NotBefore and NotOnOrAfter

The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be valid throughout the validity period.

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.

If both attributes are present, the value for `NotBefore` MUST be less than (earlier than) the value for `NotOnOrAfter`.

The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is built in to the W3C XML Schema Datatypes specification [Schema2]. All time instants are specified in Universal Coordinated Time (UTC) as described in Section 1.2.2.

Implementations MUST NOT generate time instants that specify leap seconds.

2.3.3.5.2 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"/>
```

2.3.3.5.3 Elements <AudienceRestriction> and <Audience>

The <AudienceRestriction> 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 element:

<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. It MAY also contain the unique identifier of a SAML system entity, as described by the name identifier Format URI of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

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 <AudienceRestriction> 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.

Note that multiple <AudienceRestriction> elements MAY be included in a single assertion, and each MUST be evaluated independently.

The following schema fragment defines the <AudienceRestriction> element and its **AudienceRestrictionType** complex type:

```
<element name="AudienceRestriction"
type="saml:AudienceRestrictionType"/>
<complexType name="AudienceRestrictionType">
      <complexContent>
            <extension base="saml:ConditionAbstractType">
                  <sequence>
                        <element ref="saml:Audience" maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

2.3.3.5.4 Element <OneTimeUse>

In general, relying parties may choose to retain assertions (or the information they contain in some other form), and apply them repeatedly in making decisions. The <OneTimeUse> condition element allows an authority to indicate that the information in the assertion is likely to change very soon and fresh information should be obtained for each use. An example would be an assertion containing an <AuthzDecisionStatement> which was the result of a policy which specified access control which was a function of the time of day.

If system clocks in a distributed environment could be precisely synchronized, then this requirement could be met by careful use of the validity interval. However, since some clock skew between systems will always be present, combined with unknown and possibly variable transmission delays, there is no convenient way fo the issuer to appropriately limit the lifetime of an assertion without running a substantial risk that it will already have expired before it arrives.

The <OneTimeUse> element indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be retained for future use. Relying parties are always free to request a fresh assertion for every use. However, implementations that choose to retain assertions for future use MUST observe the <OneTimeUse> element. This condition is independent from the NotBefore and NotOnOrAfter condition information.

A SAML authority MUST NOT include more than one <OneTimeUse> element within a <Conditions> element of an assertion.

For the purposes of determining the validity of the <Conditions> element, the <OneTimeUse> is considered to always be valid.

The following schema fragment defines the <OneTimeUse> element and its **OneTimeUseType** complex type:

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

2.3.3.5.5 Element <ProxyRestriction>

Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent assertions of their own on the basis of the information contained in the original assertion. A relying party MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

The <ProxyRestriction> element contains the following elements and attributes:

Count [Optional]

Specifies the number of indirections that MAY exist between this assertion and an assertion which has ultimately been issued on the basis of it.

<Audience> [Zero or More]

Specifies the set of audiences to whom new assertions MAY be issued on the basis of this assertion.

A Count value of zero indicates that a relying party MUST NOT issue an assertion to another relying party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves contain a <ProxyRestriction> element with a Count value of at most one less than this value.

If no <Audience> elements are specified, then no audience restrictions are imposed on the relying parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves contain an <AudienceRestriction> element with at least one of the <Audience> elements present in the previous <ProxyRestriction> element, and no <Audience> elements present that were not in the previous <ProxyRestriction> element.

A SAML authority MUST NOT include more than one `<ProxyRestriction>` element within a `<Conditions>` element of an assertion.

For the purposes of determining the validity of the `<Conditions>` element, the `<ProxyRestriction>` condition is considered to always be valid.

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

```
<element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
<complexType name="ProxyRestrictionType">
       <complexContent>
              <extension base="saml:ConditionAbstractType">
                     <sequence>
                            <element ref="saml:Audience" minOccurs="0"
maxOccurs="unbounded"/>
                     </sequence>
                     <attribute name="Count" type="nonNegativeInteger"
use="optional"/>
              </extension>
       </complexContent>
</complexType>
```

2.3.3.6 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>`, `<EncryptedAssertion>`, `<AssertionIDRef>`, and `<AssertionURIRef>` 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:AssertionIDRef"/>
              <element ref="saml:AssertionURIRef"/>
              <element ref="saml:Assertion"/>
              <element ref="saml:EncryptedAssertion"/>
              <any namespace="##other" processContents="lax"/>
       </choice>
</complexType>
```

2.3.4 Element <EncryptedAssertion>

The `<EncryptedAssertion>` element represents an assertion in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAssertion>` element contains the following elements:

<xenc:EncryptedData> [Required]

> The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The Type attribute SHOULD be present and, if present, MUST contain a value of http://www.w3.org/2001/04/xmlenc#Element. The encrypted content MUST contain an element that has a type derived from **AssertionType**.

<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. The value of the Recipient attribute SHOULD be the URI identifier of a SAML system entity as defined by Section 8.3.6.

Encrypted assertions are intended as a confidentiality protection when the plain-text value passes through an intermediary.

The following schema fragment defines the <EncryptedAssertion> element and its **EncryptedAssertionType** complex type:

```
<element name="EncryptedAssertion" type="saml:EncryptedAssertionType"/>
<complexType name="EncryptedAssertionType">
        <sequence>
            <element ref="xenc:EncryptedData"/>
            <element ref="xenc:EncryptedKey" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
</complexType>
```

2.4 Statements

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

2.4.1 Element <Statement>

The <Statement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. Its **StatementAbstractType** complex type is abstract 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"/>
```

2.4.2 Element <AuthnStatement>

The <AuthnStatement> 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 **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

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

AuthnInstant [Required]

> Specifies the time at which the authentication took place. The time value is encoded in UTC, as

described in Section 1.2.2.

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.

SessionNotOnOrAfter [Optional]

Specifies a time instant at which the session between the subject and the authority issuing this statement MUST be considered ended. The time value is encoded in UTC, as described in Section 1.2.2.

<SubjectLocality> [Optional]

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

<AuthnContext> [Required]

The context used by the identity provider in the authentication event that yielded this statement. Contains a reference to an authentication context class, an authentication context declaration, declaration reference, or both. See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication context information.

Assertions containing <AuthnStatement> elements MUST contain a <Subject> element.

For privacy reasons, when including a SessionIndex attribute, the value MUST NOT be a unique value identifying a principal's session at the authority. That is, it MUST NOT be reused in subsequent assertions about the same principal. It MAY be a globally unique value such that it differs in each assertion (much as the assertion's ID attribute does), or it MAY be a small integer value that is used in assertions issued on behalf of many different subjects at the same time.

The following schema fragment defines the <AuthnStatement> element and its **AuthnStatementType** complex type:

```
<element name="AuthnStatement" type="saml:AuthnStatementType"/>
<complexType name="AuthnStatementType">
      <complexContent>
            <extension base="saml:StatementAbstractType">
                  <sequence>
                        <element ref="saml:SubjectLocality" minOccurs="0"/>
                        <element ref="saml:AuthnContext"/>
                  </sequence>
                  <attribute name="AuthnInstant" type="dateTime"
use="required"/>
                  <attribute name="SessionIndex" type="string"
use="optional"/>
                  <attribute name="SessionNotOnOrAfter" type="dateTime"
use="optional"/>
            </extension>
      </complexContent>
</complexType>
```

2.4.2.1 Element <SubjectLocality>

The <SubjectLocality> element specifies the DNS domain name and IP address for the system from which the subject was authenticated. It has the following attributes:

Address [Optional]

The network address of the system from which the subject was authenticated.

`DNSName` [Optional]

The DNS name of the system from which the subject was authenticated.

This element is entirely advisory, since both of these fields are quite easily “spoofed,” but may be useful information in some applications.

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

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

2.4.2.2 Element <AuthnContext>

The `<AuthnContext>` element specifies the context of an authentication event with an authentication context class reference, an authentication context declaration or declaration reference, or both. Its complex **AuthnContextType** has the following elements:

`<AuthnContextClassRef>` [Optional]

A URI reference identifying an authentication context class that describes the authentication context declaration that follows.

`<AuthnContextDecl>` or `<AuthnContextDeclRef>` [Optional]

Either an authentication context declaration provided by value, or a URI reference that identifies such a declaration. The URI reference MAY directly resolve into an XML document containing the referenced declaration.

`<AuthenticatingAuthority>` [Zero or More]

Zero or more unique identifiers of authentication authorities that were involved in the authentication of the principal in addition to the assertion issuer.

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

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
      <sequence>
            <element ref="saml:AuthnContextClassRef" minOccurs="0"/>
            <choice minOccurs="0">
                  <element ref="saml:AuthnContextDecl"/>
                  <element ref="saml:AuthnContextDeclRef"/>
            </choice>
            <element ref="saml:AuthenticatingAuthority" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
</complexType>
<element name="AuthnContextClassRef" type="anyURI"/>
<element name="AuthnContextDeclRef" type="anyURI"/>
<element name="AuthnContextDecl" type="anyType"/>
<element name="AuthenticatingAuthority" type="anyURI"/>
```

2.4.3 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 **StatementAbstractType** with the addition of the following elements:

`<Attribute>` or `<EncryptedAttribute>` [One or More]

> The `<Attribute>` element specifies an attribute of the subject. An encrypted SAML attribute may be included with the `<EncryptedAttribute>` element.

Assertions containing `<AttributeStatement>` elements MUST contain a `<Subject>` element.

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:StatementAbstractType">
                  <choice maxOccurs="unbounded">
                        <element ref="saml:Attribute"/>
                        <element ref="saml:EncryptedAttribute"/>
                  </choice>
            </extension>
      </complexContent>
</complexType>
```

2.4.3.1 Element <AttributeDesignator>

The `<AttributeDesignator>` element identifies an attribute by name. It has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.4 for more information). The `<AttributeDesignator>` element contains the following XML attributes:

`Name` [Required]

> The name of the attribute.

`NameFormat` [Optional]

> A URI reference representing the classification of the attribute name for purposes of interpreting the name. See Section 8.3 for some URI references that MAY be used as the value of the `NameFormat` attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:attname-format:unspecified` (see Section 8.3.1) is in effect.

`FriendlyName` [Optional]

> A string that provides a more human-readable form of the attribute's name, which may be useful in cases in which the actual `Name` is complex, such as an OID or a UUID. This value MUST NOT be used as a basis for formally identifying SAML attributes.

Arbitrary attributes

> This complex type uses an `<xsd:anyAttribute>` extension point to allow arbitrary XML attributes to be added to `<AttributeDesignator>` constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply additional parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

The following schema fragment defines the <AttributeDesignator> element and its **AttributeDesignatorType** complex type:

```
<element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
      <attribute name="Name" type="string" use="required"/>
      <attribute name="NameFormat" type="anyURI" use="optional"/>
      <attribute name="FriendlyName" type="string" use="optional"/>
      <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

2.4.3.2 Element <Attribute>

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 and attributes:

<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. If more than one <AttributeValue> element is supplied for an attribute, and any of the elements have a datatype assigned through xsi:type, then all of the <AttributeValue> elements must have the identical datatype assigned.

Arbitrary attributes

This complex type inherits from **AttributeDesignatorType** the ability to add arbitrary XML attributes to <Attribute> constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply the context in which the attribute should be understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

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>
```

2.4.3.2.1 Element <AttributeValue>

The <AttributeValue> element supplies the value of a specified attribute. It is of the **xsd: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 datatype 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.

> **Note:** Specifying a datatype on <AttributeValue> using xsi:type will require the presence of the extension schema that defines the datatype in order for schema processing to proceed.

The following schema fragment defines the <AttributeValue> element:

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

2.4.3.3 Element <EncryptedAttribute>

The <EncryptedAttribute> element represents a SAML attribute in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAttribute> element contains the following elements:

<xenc:EncryptedData> [Required]

> The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The Type attribute SHOULD be present and, if present, MUST contain a value of http://www.w3.org/2001/04/xmlenc#Element. The encrypted content MUST contain an element that has a type that is derived from **AttributeType**.

<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. The value of the Recipient attribute SHOULD be the URI identifier of a SAML system entity as defined by Section 8.3.6.

Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through an intermediary.

The following schema fragment defines the <EncryptedAttribute> element and its **EncryptedAttributeType** complex type:

```
<element name="EncryptedAttribute" type="saml:EncryptedAttributeType"/>
<complexType name="EncryptedAttributeType">
    <sequence>
        <element ref="xenc:EncryptedData"/>
        <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.4.4 Element <AuthzDecisionStatement>

> **Note:** The <AuthzDecisionStatement> feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

The <AuthzDecisionStatement> 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 `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

`Resource` [Required]

A URI reference identifying the resource to which access authorization is sought. This attribute MAY 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.

Assertions containing `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

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

```
<element name="AuthzDecisionStatement"
type="saml:AuthzDecisionStatementType"/>
<complexType name="AuthzDecisionStatementType">
      <complexContent>
            <extension base="saml:StatementAbstractType">
                  <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>
```

2.4.4.1 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 in the enclosing `<Response>`.

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>
```

2.4.4.2 Element <Action>

The `<Action>` element specifies an action on the specified resource for which permission is sought. Its string-data content provides the label for an action sought to be performed on the specified resource, and it has the following attribute:

`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 8.1.2 is in effect.

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" use="required"/>
              </extension>
       </simpleContent>
</complexType>
```

2.4.4.3 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:

<AssertionIDRef> [Any number]

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

<AssertionURIRef> [Any number]

Specifies an assertion by means of a URI reference.

<Assertion> [Any number]

Specifies an assertion by value.

<EncryptedAssertion> [Any number]

Specifies an encrypted 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:AssertionIDRef"/>
             <element ref="saml:AssertionURIRef"/>
             <element ref="saml:Assertion"/>
             <element ref="saml:EncryptedAssertion"/>
       </choice>
</complexType>
```

3 SAML Protocols

SAML assertions and messages about them MAY be generated and exchanged using a variety of protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting queries, assertions, and messages using existing widely deployed transport protocols.

Specific SAML request and response messages derive from common types. The requester sends an element derived from **RequestAbstractType** to a SAML responder, and the responder generates an element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

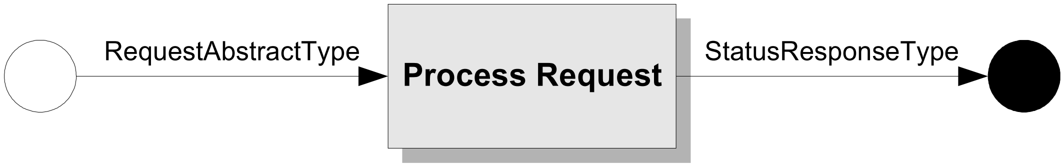

Figure 1: SAML Request-Response Protocol

The protocols defined by SAML achieve the following actions:

- Returning one or more requested assertions (includes a direct request of the desired assertions, as well as querying for assertions that meet particular criteria)
- Performing authentication on request and returning the corresponding assertion
- Registering a name identifier or terminating a name registration on request
- Retrieving a protocol message that has been requested by means of an artifact
- Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on request
- Providing a name identifier mapping on request

Throughout this section, text mentions of elements and types in the SAML protocol namespace are not shown with the conventional namespace prefix `samlp:`. For clarity, text mentions of elements and types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:`.

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
                    …
                    </documentation>
          </annotation>
…
</schema>
```

3.2 Requests and Responses

The following sections define the SAML constructs that underlie all of the request and response messages used in SAML protocols.

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:

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

`ID` [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 `ID` 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 4.

`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 4.

`IssueInstant` [Required]

The time instant of issue of the request. The time value is encoded in UTC, as described in Section 1.2.2.

`Consent` [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a user in the sending this request. See Section 8.4 for some URI references that MAY be used as the value of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in effect.

`<saml:Issuer>` [Optional]

Identifies the entity that generated the request message.

`<ds:Signature>` [Optional]

An XML Signature that authenticates the request, as described in Section 5.

<Extensions> [Optional]

This extension point contains optional protocol message extension elements that are agreed on between the communicating parties. No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extensions MUST NOT include local (non-namespace-qualified) elements or elements qualified by a SAML-defined namespace within this element.

If a <ds:Signature> element is present, a responder SHOULD verify that the signature is valid. If it is invalid, the responder SHOULD NOT rely on the contents of the request and SHOULD respond with an error.

If a Consent attribute is included and the value indicates that some form of user consent has been obtained, then the request SHOULD be signed.

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:Extensions" minOccurs="0"/>
       </sequence>
       <attribute name="ID" 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="##other" processContents="lax"
maxOccurs="unbounded"/>
       </sequence>
</complexType>
```

3.2.1.1 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:

ID [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 ID 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 request's ID 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 4.

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 4.

IssueInstant [Required]

The time instant of issue of the response. The time value is encoded in UTC, as described in Section 1.2.2.

Recipient [Optional]

A URI reference indicating 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 profiles of use. If it is 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.

<saml:Issuer> [Optional]

Identifies the entity that generated the response message.

<ds:Signature> [Optional]

An XML Signature that authenticates the response, as described in Section 5.

<Extensions> [Optional]

This contains optional protocol message extension elements that are agreed on between the communicating parties. . No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extensions MUST NOT include local (non-namespace-qualified) elements or elements qualified by a SAML-defined namespace within this element.

<Status> [Required]

A code representing the status of the corresponding request.

If a <ds:Signature> element is present, a requester SHOULD verify that the signature is valid. If it is invalid, the requester SHOULD NOT rely on the contents of the response.

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:Extensions" minOccurs="0"/>
              <element ref="samlp:Status"/>
       </sequence>
       <attribute name="ID" 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>
```

3.2.1.2 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>
```

3.2.1.3 Element <StatusCode>

The `<StatusCode>` element specifies a code or a set of 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 a URI reference. 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 permissible top-level `<StatusCode>` values are as follows:

`urn:oasis:names:tc:SAML:2.0:status:Success`

The request succeeded. Additional information MAY be returned in the `<StatusMessage>` and/or `<StatusDetail>` elements.

`urn:oasis:names:tc:SAML:2.0:status:Requester`

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

`urn:oasis:names:tc:SAML:2.0:status:Responder`

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

`urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`

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

The following second-level status codes are referenced at various places in this specification. Additional second-level status codes MAY be defined in future versions of the SAML specification. SAML system entities are free to define more specific status codes by defining appropriate URI references.

`urn:oasis:names:tc:SAML:2.0:status:AuthnFailed`

The responding provider was unable to successfully authenticate the principal.

`urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`

The responding provider does not support the specified name identifier format for the requested subject.

`urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext`

The specified authentication context requirements cannot be met by the responder.

`urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP`

Used by an intermediary to indicate that none of the supported identity provider `<Loc>` elements in an `<IDPList>` can be resolved or that none of the supported identity providers are available.

`urn:oasis:names:tc:SAML:2.0:status:NoPassive`

Indicates the identity provider cannot authenticate the principal passively, as has been requested.

`urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP`

Used by an intermediary to indicate that none of the identity providers in an `<IDPList>` are supported by the intermediary.

`urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`

Indicates that an identity provider cannot authenticate the principal directly and is not permitted to proxy the request further.

`urn:oasis:names:tc:SAML:2.0:status: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.

`urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported`

The SAML responder or SAML authority does not support the request.

`urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated`

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

`urn:oasis:names:tc:SAML:2.0:status: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.

`urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow`

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

`urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized`

The resource value provided in the request message is invalid or unrecognized.

`urn:oasis:names:tc:SAML:2.0:status:TooManyResponses`

The response message would contain more elements than the SAML responder is able to return.

`urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`

The responding provider does not recognize the principal specified or implied by the request.

`urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding`

The SAML responder cannot properly fulfill the request using the protocol binding specified in the request.

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="anyURI" use="required"/>
```

```
</complexType>
```

3.2.1.4 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"/>
```

3.2.1.5 Element <StatusDetail>

The <StatusDetail> element MAY be used to specify additional information concerning an error condition. The additional information consists of zero or more elements from any namespace, with no requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

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 or querying for assertions by subject and statement type.

3.3.1 Element <AssertionIDRequest>

If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest> message element can be used to request that they be returned in a <Response> message. The <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 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:AssertionIDRef"
maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

3.3.2 Queries

The following sections define the SAML query request messages.

3.3.2.1 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 `<saml:Subject>` element to **RequestAbstractType**.

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>
                </extension>
        </complexContent>
</complexType>
```

3.3.2.2 Element <AuthnQuery>

The `<AuthnQuery>` 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 `<AuthnQuery>` message MUST NOT be used as a request for a new authentication using credentials provided in the request. `<AuthnQuery>` 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 **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

`SessionIndex` [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 within the context of the supplied session information?”

`<RequestedAuthnContext>` [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 that satisfy the authentication context requirements in this element?"

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

- Rules given in Section 3.3.4for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If the `SessionIndex` attribute is present in the query, at least one `<AuthnStatement>` element 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.
- If the `<RequestedAuthnContext>` element is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions MUST contain an `<AuthnContext>` element that satisfies the element in the query (see Section 3.3.2.3). It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.

The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex type:

```
<element name="AuthnQuery" type="samlp:AuthnQueryType"/>
<complexType name="AuthnQueryType">
       <complexContent>
              <extension base="samlp:SubjectQueryAbstractType">
                     <sequence>
                            <element ref="samlp:RequestedAuthnContext"
minOccurs="0"/>
                     </sequence>
                     <attribute name="SessionIndex" type="string"
use="optional"/>
              </extension>
       </complexContent>
</complexType>
```

3.3.2.3 Element <RequestedAuthnContext>

The <RequestedAuthnContext> element specifies the authentication context requirements of authentication statements returned in response to a request or query. Its **RequestedAuthnContextType** complex type defines the following elements and attributes:

<saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

Specifies one or more URI references identifying authentication context classes or declarations. (For more information about authentication context classes, see [SAMLAuthnCxt].)

Comparison [Optional]

Specifies the comparison method used to evaluate the requested context classes or statements, one of "exact", "minimum", "maximum", or "better". The default is "exact".

Either a set of class references or a set of declaration references can be used. The set of supplied references MUST be evaluated as an ordered set, where the first element is the most preferred authentication context class or declaration. If none of the specified classes or declarations can be satisfied in accordance with the rules below, then the responder MUST return a <Response> message with a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication statement MUST be the exact match of at least one of the authentication contexts specified.

If Comparison is set to "minimum", then the resulting authentication context in the authentication statement MUST be at least as strong (as deemed by the responder) as one of the authentication contexts specified.

If Comparison is set to "better", then the resulting authentication context in the authentication statement MUST be stronger (as deemed by the responder) than any one of the authentication contexts specified.

If Comparison is set to "maximum", then the resulting authentication context in the authentication statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength of at least one of the authentication contexts specified.

The following schema fragment defines the <RequestedAuthnContext> element and its **RequestedAuthnContextType** complex type:

```
<element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
<complexType name="RequestedAuthnContextType">
       <choice>
              <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
              <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
       </choice>
```

```
      <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
use="optional"/>
</complexType>
<simpleType name="AuthnContextComparisonType">
      <restriction base="string">
            <enumeration value="exact"/>
            <enumeration value="minimum"/>
            <enumeration value="maximum"/>
            <enumeration value="better"/>
      </restriction>
</simpleType>
```

3.3.2.4 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, to the extent allowed by policy. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

`<saml:AttributeDesignator>` [Any Number]

Each `<saml: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 3.3.4 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 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>
            </extension>
      </complexContent>
</complexType>
```

3.3.2.5 Element <AuthzDecisionQuery>

The `<AuthzDecisionQuery>` 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.

> **Note:** The `<AuthzDecisionQuery>` feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

This element is of type **AuthzDecisionQueryType**, 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.

`<saml:Action>` [One or More]

The actions for which authorization is requested.

`<saml: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 3.3.4for matching against the `<Subject>` element of the query identify the assertions that may be returned.

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

```
<element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
<complexType name="AuthzDecisionQueryType">
        <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>
```

3.3.3 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** and adds the following elements:

`<saml:Assertion>` or `<saml:EncryptedAssertion>` [Any Number]

Specifies an assertion by value, or optionally an encrypted assertion by value. (See Section 2.3.3 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">
                        <choice minOccurs="0" maxOccurs="unbounded">
                                <element ref="saml:Assertion"/>
                                <element ref="saml:EncryptedAssertion"/>
                        </choice>
                </extension>
        </complexContent>
</complexType>
```

3.3.4 Processing Rules

In response to a query message, every assertion returned by a SAML authority MUST contain a `<saml:Subject>` element that **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 an identifier element (any element whose type is derived from **BaseIDAbstractType**), then S1 MUST include an identical identifier element, but the element MAY be encrypted (or not) in either S1 or S2. In other words, the decrypted form of the identifier MUST be identical in S1 and S2. "Identical" means that the identifier element's content and attribute values MUST be the same. An encrypted identifier will be identical to the original according to this definition, once decrypted.
- If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 MUST include at least one `<saml:SubjectConfirmation>` element such that the assertion's subject can be confirmed in the manner described by at least one element in the requested set.

As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with different `Format` values and element content, even if the two identifiers are considered to refer to the same principal.

If the SAML authority cannot provide an assertion with any statements satisfying the constraints expressed by a query or assertion reference, the `<Response>` element MUST NOT contain an `<Assertion>` element and MUST include a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Success`.

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

3.4 Authentication Request Protocol

When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing authentication statements to establish a security context at one or more relying parties, it can use the authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and request that it return a `<Response>` message containing one or more such assertions.Such assertions MAY contain additional statements of any type, but at least one assertion MUST contain at least one authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

Apart from this requirement, the specific contents of the returned assertions depend on the profile or context of use. Also, the exact means by which the principal or agent authenticates to the identity provider are not specified, though the means of authentication might impact the content of the response. Other issues related to the validation of authentication credentials by the identity provider or any communication between the identity provider and any other entities involved in the authentication process are also out of scope of this protocol.

The descriptions and processing rules in the following sections reference the following actors, many of whom might be the same entity in a particular profile of use:

Request Issuer

> The entity who creates the authentication request and to whom the response is to be returned.

Presenter

> The entity who presents the request to the authority and either authenticates itself during the sending of the message, or relies on an existing security context to establish its identity. If not the

request issuer, the sender acts as an intermediary between the request issuer and the responding identity provider.

Requested Subject

The entity about whom one or more assertions are being requested.

Confirming Subject

The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>` elements of the resulting assertion(s).

Relying Party

The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by the profile or context of use, generally to establish a security context.

3.4.1 Element <AuthnRequest>

To request that an identity provider issue an assertion with an authentication statement, a request issuer or presenter authenticates to it (or relies on an existing security context) and sends it an `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may be information that relates to the content of the assertion and/or information that relates to how the resulting `<Response>` message should be delivered to the request issuer.

The request issuer might not be the same as the presenter of the request, if for example the request issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the requested subject to provide a service.

The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and adds the following elements and attributes, all of which are optional in general, but may be required by specific profiles:

`<saml:Subject>` [Optional]

Specifies the requested subject of the resulting assertion(s). This may include one or more `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions can be confirmed.

If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the requested subject. If no `<SubjectConfirmation>` elements are included, then the presenter is presumed to be the only confirming entity required and the method is implied by the profile of use and/or the policies of the identity provider.

`<NameIDPolicy>` [Optional]

Specifies constraints on the name identifier to be used to represent the requested subject. If omitted, then any type of identifier supported by the identity provider for the requested subject can be used, constrained by any relevant deployment-specific policies, with respect to privacy, for example.

`<saml:Conditions>` [Optional]

Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary. The information in this element is used as input to the process of constructing the assertion, rather than as conditions on the use of the request itself.

<RequestedAuthnContext> [Optional]

Specifies the requirements, if any, that the request issuer places on the authentication context that applies to the responding provider's authentication of the presenter. See Section 3.3.2.3 for processing rules regarding this element.

<Scoping> [Optional]

Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to subsequent identity providers by the responder.

IsPassive [Optional]

A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of the user interface from the request issuer and interact with the presenter in a noticeable fashion. If a value is not provided, the default is "true".

ForceAuthn [Optional]

A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than rely on a previous security context. If a value is not provided, the default is "false". However, if both ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the presenter unless the constraints of IsPassive can be met.

ProtocolBinding [Optional]

A URI reference that identifies a SAML protocol binding to be used when returning the <Response> message. See [SAMLBind] for more information about protocol bindings and URI references defined for them.

AssertionConsumerServiceIndex [Optional]

Indirectly identifies the location to which the <Response> message should be returned to the request issuer. It applies only to profiles in which the request issuer is different from the presenter. The identity provider MUST have a trusted means to map the index value in the attribute to a location associated with the request issuer. [SAMLMetadata] provides one possible mechanism. If omitted, then the identity provider MUST return the <Response> message to the default location associated with the request issuer for the profile of use.

AssertionConsumerServiceURL [Optional]

Specifies by value the location to which the <Response> message MUST be returned to the request issuer. The responder MUST ensure by some means that the value specified is in fact associated with the request issuer. [SAMLMetadata] provides one possible mechanism.

AttributeConsumingServiceIndex [Optional]

Indirectly identifies information associated with the request issuer describing the SAML attributes the request issuer desires or requires be supplied by the identity provider in the <Response> message. The identity provider MUST have a trusted means to map the index value in the attribute to information associated with the request issuer. [SAMLMetadata] provides one possible mechanism. The identity provider MAY use this information to populate one or more <saml:AttributeStatement> elements in the assertion(s) it returns.

ProviderName [Optional]

Specifies the human-readable name of the request issuer for use by the presenter's user agent or the identity provider.

See Section 3.4.1.5 for general processing rules regarding this message.

The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType** complex type:

```
<element name="AuthnRequest" type="samlp:AuthnRequestType"/>
<complexType name="AuthnRequestType">
        <complexContent>
                <extension base="samlp:RequestAbstractType">
                        <sequence>
                                <element ref="saml:Subject" minOccurs="0"/>
                                <element ref="samlp:NameIDPolicy" minOccurs="0"/>
                                <element ref="saml:Conditions" minOccurs="0"/>
                                <element ref="samlp:RequestedAuthnContext"
minOccurs="0"/>
                                <element ref="samlp:Scoping" minOccurs="0"/>
                        </sequence>
                        <attribute name="IsPassive" type="boolean"
use="optional"/>
                        <attribute name="ForceAuthn" type="boolean"
use="optional"/>
                        <attribute name="ProtocolBinding" type="anyURI"
use="optional"/>
                        <attribute name="AssertionConsumerServiceIndex"
type="unsignedShort" use="optional"/>
                        <attribute name="AssertionConsumerServiceURL"
type="anyURI" use="optional"/>
                        <attribute name="AttributeConsumingServiceIndex"
type="unsignedShort" use="optional"/>
                        <attribute name="ProviderName" type="string"
use="optional"/>
                </extension>
        </complexContent>
</complexType>
```

3.4.1.1 Element <NameIDPolicy>

The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

Format [Required]

> Specifies the URI reference corresponding to a name identifier format defined in this or another specification (see Section 8.3for examples).

SPNameQualifier [Optional]

> Used with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:persistent or urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, it optionally specifies that a federated identifier be returned (or created) in the namespace of a service provider other than the request issuer, or in the namespace of an affiliation group of service providers.

AllowCreate [Optional]

> A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the request, to create a new identifier to represent the principal. Defaults to "true". When "false", the request issuer constrains the identity provider to only issue an assertion to it if an acceptable identifier for the principal has already been established between them.

When this element is used, if the content is not understood by or acceptable to the identity provider, then a <Response> message element MUST be returned with an error <Status>, and MAY contain a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates that the resulting assertion(s) MUST contain <EncryptedID> elements instead of plaintext. The underlying name identifier's unencrypted form can be of any type supported by the identity provider for the requested subject.

Any `Format` value (or the omission of this element) MAY result in an `<EncryptedID>` in the resulting assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

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

```
<element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
<complexType name="NameIDPolicyType">
      <sequence/>
      <attribute name="Format" type="anyURI" use="required"/>
      <attribute name="SPNameQualifier" type="string" use="optional"/>
      <attribute name="AllowCreate" type="boolean" use="optional"/>
</complexType>
```

3.4.1.2 Element <Scoping>

The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity providers by the responder. Its **ScopingType** complex type defines the following elements and attribute:

`ProxyCount` [Optional]

> Specifies the number of proxying indirections permissible between the identity provider that receives this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of zero permits no proxying, while omitting this attribute expresses no such restriction.

`<IDPList>` [Optional]

> An advisory list of identity providers and associated information that the request issuer deems acceptable to respond to the request.

`<RequesterID>` [Zero or More]

> Identifies the set of requesting entities on whose behalf the request issuer is acting. Used to communicate the chain of request issuers when proxying occurs, as described in Section 3.4.1.6. See Section 8.3.6 for a description of entity identifiers.

In profiles specifying an active intermediary, the intermediary MAY examine the list and return a `<Response>` message with an error `<Status>` and a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP` or `urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP` if it cannot contact or does not support any of the specified identity providers.

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

```
<element name="Scoping" type="samlp:ScopingType"/>
<complexType name="ScopingType">
      <sequence>
            <element ref="samlp:IDPList" minOccurs="0"/>
            <element ref="samlp:RequesterID" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
</complexType>
<element name="RequesterID" type="anyURI"/>
```

3.4.1.3 Element <IDPList>

The `<IDPList>` element specifies the identity providers trusted by the request issuer to authenticate the presenter. Its **IDPListType** complex type defines the following elements:

<IDPEntry> [One or More]

Information about a single identity provider.

<GetComplete> [Optional]

If the <IDPList> is not complete, using this element specifies a URI reference that resolves to the complete list.

The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
<element name="IDPList" type="samlp:IDPListType"/>
<complexType name="IDPListType">
      <sequence>
            <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
            <element ref="samlp:GetComplete" minOccurs="0"/>
      </sequence>
</complexType>
<element name="GetComplete" type="anyURI"/>
```

3.4.1.4 Element <IDPEntry>

The <IDPEntry> element specifies a single identity provider trusted by the request issuer to authenticate the presenter. Its **IDPEntryType** complex type defines the following attributes:

ProviderID [Required]

The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

Name [Optional]

A human-readable name for the identity provider.

Loc [Optional]

A URI reference representing the location of a profile-specific endpoint supporting the authentication request protocol. The binding to be used must be understood from the profile of use.

The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
<element name="IDPEntry" type="samlp:IDPEntryType"/>
<complexType name="IDPEntryType">
      <sequence/>
      <attribute name="ProviderID" type="anyURI" use="required"/>
      <attribute name="Name" type="string" use="optional"/>
      <attribute name="Loc" type="anyURI" use="optional"/>
</complexType>
```

3.4.1.5 Processing Rules

The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is therefore typically profiled for use in a specific context in which this optionality is constrained and specific kinds of input and output are required or prohibited. The following processing rules apply as invariant behavior across any profile of this protocol exchange. All other processing rules associated with the underlying request and response messages MUST also be observed.

The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing one or more assertions that meet the specifications defined by the request, or with a <Response> message containing a <Status> describing the error that occurred. The responder MAY conduct additional message exchanges with the presenter as needed to initiate or complete the authentication process, subject to the nature of the protocol binding and the authentication mechanism. As described in the next section, this includes proxying the request by directing the presenter to another identity provider by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to authenticate the presenter to the original responder, in effect using SAML as the authentication mechanism.

If the responder is unable to authenticate the presenter or does not recognize the requested subject, it MUST return a `<Response>` with an error `<Status>`, and MAY return a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:AuthnFailed` or `urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

If the `<saml:Subject>` element in the request is present, then the resulting assertions' `<saml:Subject>` MUST **strongly match** the request `<saml:Subject>`, as described in Section 3.3.4, except that the identifier MAY be in a different format if specified by `<NameIDPolicy>`. In such a case, the identifier's physical content MAY be different, but it MUST refer to the same principal.

All of the content defined specifically within `<AuthnRequest>` is optional, although some may be required by certain profiles. In the absence of any specific content at all, the following behavior is assumed:

- The assertion(s) returned MUST contain a `<saml:Subject>` element that represents the presenter. The identifier type and format are determined by the identity provider. At least one statement MUST be a `<saml:AuthnStatement>` that describes the authentication performed by the responder or authentication service associated with it.
- The request presenter should, to the extent possible, be the only entity able to satisfy the `<saml:SubjectConfirmation>` of the assertion(s). In the case of weaker confirmation methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element referencing the request issuer as an acceptable relying party. Other audiences MAY be included as deemed appropriate by the identity provider.

3.4.1.6 Proxying

If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or cannot directly authenticate the presenter, but believes that the presenter has already authenticated to another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying identity provider.

Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`, completing the overall message exchange. Both the proxying and authenticating identity providers MAY include constraints on proxying activity in the messages and assertions they issue, as described in previous sections and below.

The request issuer can influence proxy behavior by including a `<Scoping>` element where the provider sets a desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by including an ordered `<IDPList>` of preferred providers.

An identity provider can control secondary use of its assertions by proxying identity providers using a `<ProxyRestriction>` element in the assertions it issues.

3.4.1.6.1 Proxying Processing Rules

An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. The identity provider MUST return an error <Status> containing a second-level <StatusCode> value of urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded, unless it can directly authenticate the presenter.

If it chooses to proxy to a SAML identity provider, when creating the new <AuthnRequest>, the proxying identity provider MUST include equivalent or stricter forms of all the information included in the original request (such as authentication context policy). Note, however, that the proxying provider is free to specify whatever <NameIDPolicy> it wishes to maximize the chances of a succesful response.

If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for example) will be honored by the authenticating provider.

The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less than the original value. If the original request does not contain a <ProxyCount> attribute, then the new request SHOULD contain a <ProxyCount> attribute.

If an <IDPList> was specified in the original request, the new request MUST also contain an <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the <IDPList>, but MUST NOT remove any from the list.

The authentication request and response are processed in normal fashion, in accordance with the rules given in this section and the profile of use. Once the presenter has authenticated to the proxying identity provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- The proxying identity provider prepares a new assertion on its own behalf by copying in the relevant information from the original assertion or non-SAML equivalent.
- The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original request issuer's preferences, as defined by its <NameIDPolicy> element.
- The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext> element containing a <saml:AuthenticatingAuthority> element referencing the identity provider to which the proxying identity provider referred the presenter. If the original assertion contains <saml:AuthnContext> information that includes one or more <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the new assertion, with the new element placed after them.
- If the authenticating identity provider is not a SAML provider, then the proxying identity provider MUST generate a unique identifier value for the authenticating provider. This value SHOULD be consistent over time across different requests. The value MUST not conflict with values used or generated by other SAML providers.
- Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in accordance with the policies of the proxying identity provider, provided that the original requirements dictated by the request issuer are met.

If, in the future, the identity provider is asked to authenticate the same presenter for a second request issuer, and this request is equally or less strict than the original request (as determined by the proxying identity provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating identity provider and immediately issue another assertion (assuming the original assertion or non-SAML equivalent it received is still valid).

3.5 Artifact Resolution Protocol

The artifact resolution protocol provides a mechanism by which SAML protocol messages can be transported in a SAML binding by reference instead of by value. Both requests and responses can be obtained by reference using this specialized protocol. A message sender, instead of binding a message to a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding protocol to resolve the artifact into the original protocol message. The most common use for this mechanism is with bindings that cannot easily carry a message because of size constraints.

Depending on the characteristics of the underlying message being passed by reference, the artifact resolution protocol MAY require protections such as mutual authentication, integrity protection, confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used by any party.

Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly as if the message so obtained had been sent originally in place of the artifact.

3.5.1 Element <ArtifactResolve>

The `<ArtifactResolve>` message is used to request that a SAML protocol message be returned in an `<ArtifactResponse>` message by specifying an artifact that represents the SAML protocol message. The original transmission of the artifact is governed by the specific protocol binding that is being used; see [SAMLBind] for more information on the use of artifacts in bindings.

The `<ArtifactResolve>` message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and adds the following element:

`<Artifact>` [Required]

> The artifact value that the requester received and now wishes to translate into the protocol message it represents. See [SAMLBind] for specific artifact format information.

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

```
<element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
<complexType name="ArtifactResolveType">
        <complexContent>
                <extension base="samlp:RequestAbstractType">
                        <sequence>
                                <element ref="samlp:Artifact"/>
                        </sequence>
                </extension>
        </complexContent>
</complexType>
<element name="Artifact" type="string"/>
```

3.5.2 Element <ArtifactResponse>

The recipient of an `<ArtifactResolve>` message MUST respond with an `<ArtifactResponse>` message element. This element is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol message being returned. This wrapped message element can be a request or a response.

The `<ArtifactResponse>` message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ArtifactResponse> element and its **ArtifactResponseType** complex type:

```
<element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
<complexType name="ArtifactResponseType">
      <complexContent>
            <extension base="samlp:StatusResponseType">
                  <sequence>
                        <any namespace="##any" processContents="lax"
minOccurs="0"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

3.5.3 Processing Rules

If the responder recognizes the artifact as valid, then it responds with the associated protocol message in an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse> element with no embedded message. In both cases, the <Status> element MUST include a <StatusCode> element with the code value Success. A response message with no embedded message inside it is termed an empty response in the remainder of this section.

The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent request with the same artifact by any requester results in an empty response as described above.

Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY be intended for consumption by any party that receives it and can respond appropriately. In most other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST be associated with the intended recipient of the message that the artifact represents. If the artifact issuer receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the original intended recipient, then the artifact issuer MUST return an empty response.

The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and return it in an <ArtifactResolve> message to the issuer.

Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message will contain its own message identifier, and in the case of an embedded response, may contain a different InResponseTo value that corresponds to the original request message to which the embedded message is responding.

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

3.6 Name Identifier Management Protocol

After establishing a persistent name identifier for a principal, an identity provider wishing to change the value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will no longer be used to refer to the principal, informs service providers of the change by sending them a <ManageNameIDRequest> message.

A service provider also uses this message to register or change the SPProvidedID value to be included when the underlying name identifier is used to communicate with it, or to terminate the use of a name identifier between itself and the identity provider.

3.6.1 Element <ManageNameIDRequest>

A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name identifier or to indicate the termination of the use of a name identifier.

The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

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

<saml:NameID> or <saml:EncryptedID> [Required]

> The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request.

<NewID> or <NewEncryptedID> or <Terminate> [Required]

> The new identifier value (in plaintext or encrypted form) to be used when communicating with the requesting provider concerning this principal, or an indication that the use of the old identifier has been terminated. In the former case, if the requester is the service provider, the new identifier MUST appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the identity provider, the new value will appear in subsequent <NameID> elements as the element's content.

The following schema fragment defines the <ManageNameIDRequest> element and its **ManageNameIDRequestType** complex type:

```
<element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
<complexType name="ManageNameIDRequestType">
    <complexContent>
        <extension base="samlp:RequestAbstractType">
            <sequence>
                <choice>
                    <element ref="saml:NameID"/>
                    <element ref="saml:EncryptedID"/>
                </choice>
                <choice>
                    <element ref="samlp:NewID"/>
                    <element ref="samlp:NewEncryptedID"/>
                    <element ref="samlp:Terminate"/>
                </choice>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="NewID" type="string"/>
<element name="NewEncryptedID" type="saml:EncryptedIDType"/>
<element name="Terminate" type="samlp:TerminateType"/>
<complexType name="TerminateType">
    <sequence/>
</complexType>
```

3.6.2 Element <ManageNameIDResponse>

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

The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ManageNameIDResponse> element:

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

3.6.3 Processing Rules

If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize, the responding provider MUST respond with an error <Status> and MAY respond with a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

If the <Terminate> element is included in the request, the requesting provider is indicating that (in the case of a service provider) it will no longer accept assertions from the identity provider or (in the case of an identity provider) it will no longer issue assertions to the service provider about the principal. The receiving provider can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for whom a relationship has been terminated.

If the service provider requests that its identifier be changed by including a <NewID> (or <NewEncryptedID>) element, the identity provider MUST include the element's content as the SPProvidedID when subsequently communicating to the service provider regarding this principal.

If the identity provider requests that its identifier be changed by including a <NewID> (or <NewEncryptedID>) element, the service provider MUST use the element's content as the <saml:NameID> element content when subsequently communicating with the identity provider regarding this principal.

In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute MUST contain the most recent name identifier information established between the providers for the principal.

In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:persistent or urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, the NameQualifier attribute MUST contain the unique identifier of the identity provider or be omitted. If the identifier was established 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 or be omitted.

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.

3.7 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 log out a principal due to a timeout. The reason for the logout event can be indicated through the Reason attribute.

The principal may have established authenticated sessions with both the session authority and individual session participants, based on assertions containing authentication statements 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 assertion containing the authentication statement 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 assertions containing authentication statements under its current session with the principal, with the exception of the session participant that sent the <LogoutRequest> message to the session authority.

3.7.1 Element <LogoutRequest>

A session participant or session authority sends a <LogoutRequest> message to indicate that a session has been terminated.

The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

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

NotOnOrAfter [Optional]

The time at which the request expires. The time value is encoded in UTC, as described in Section 1.2.2.

Reason [Optional]

An indication of the reason for the logout, in the form of a URI reference.

<saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

The identifier and associated attributes (in plaintext or encrypted form) 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.

The following schema fragment defines the <LogoutRequest> element and associated **LogoutRequestType** complex type:

```
<element name="LogoutRequest" type="samlp:LogoutRequestType"/>
<complexType name="LogoutRequestType">
        <complexContent>
                <extension base="samlp:RequestAbstractType">
                        <sequence>
                                <choice>
                                        <element ref="saml:BaseID"/>
                                        <element ref="saml:NameID"/>
                                        <element ref="saml:EncryptedID"/>
                                </choice>
                                <element ref="samlp:SessionIndex" minOccurs="0"
maxOccurs="unbounded"/>
                        </sequence>
                        <attribute name="Reason" type="anyURI" minOccurs="0"/>
                        <attribute name="NotOnOrAfter" type="dateTime"
minOccurs="0"/>
                </extension>
        </complexContent>
</complexType>
<element name="SessionIndex" type="string"/>
```

3.7.2 Element <LogoutResponse>

The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of type **StatusResponseType**, with no additional content specified.

The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <LogoutResponse> element:

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

3.7.3 Processing Rules

The message sender MAY use the Reason attribute to indicate the reason for sending the <LogoutRequest>. The following values are defined by this specification for use by all message senders; other values MAY be agreed on between participants:

urn:oasis:names:tc:SAML:2.0:logout:user

Specifies that the message is being sent because the principal wishes to terminate the indicated session.

urn:oasis:names:tc:SAML:2.0:logout:admin

Specifies that the message is being sent because an administrator wishes to terminate the indicated session for that principal.

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

Additional processing rules are provided in the following sections.

3.7.3.1 Session Participant Rules

When a session participant receives a <LogoutRequest> message, the session participant MUST authenticate the message. If the sender is the authority that provided an assertion containing an authentication statement linked to the principal's current session, the session participant MUST invalidate the principal's session(s) referred to by the <saml:BaseID>, <saml:NameID>, or <saml:EncryptedID> element, and any <SessionIndex> elements supplied in the message. If no <SessionIndex> elements are supplied, then all sessions associated with the principal MUST be invalidated.

The session participant MUST apply the logout request message to any assertion that meets the following conditions, even if the assertion arrives after the logout request:

- The <SessionIndex> of one of the assertion's authentication statements matches one specified in the logout request, or the logout request contains no <SessionIndex> elements.
- The assertion would otherwise be valid.
- The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on the message).

3.7.3.2 Session Authority Rules

When a session authority receives a <LogoutRequest> message, the session authority MUST authenticate the sender. If the sender is a session participant to which the session authority provided an containing an authentication statement for the current session, then the session authority SHOULD do the following in the specified order:

- Send a <LogoutRequest> message to any session authority on behalf of whom the session authority proxied the user's authentication, unless the second authority is the originator of the <LogoutRequest>.
- Send a <LogoutRequest> message to each session participant for which the session authority provided assertions in the current session, *other than* the originator of a current <LogoutRequest>.
- Terminate the principal's current session as specified by the <saml:BaseID>, <saml:NameID>, or <saml:EncryptedID> element, and any <SessionIndex> elements present in the logout request message.

If an error occurs during this further processing of the logout (for example, other session participants may not all implement the particular single logout protocol binding used by the requesting session participant), then the session authority MUST respond to the original requester with a <LogoutResponse> message, indicating the status of the logout request. The value urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding is provided for a second-level <StatusCode>, indicating that a session participant should retry the <LogoutRequest> using a different protocol binding.

Note that a session authority MAY initiate a logout for reasons other than having received a <LogoutRequest> from a session participant – these include, but are not limited to:

- If some timeout period was agreed out-of-band with an individual session participant, the session authority MAY send a <LogoutRequest> to that individual participant alone.
- An agreed global timeout period has been exceeded.
- The principal or some other trusted entity has requested logout of the principal directly at the session authority.
- The session authority has determined that the principal's credentials may have been compromised.

When constructing a logout request message, the session authority MUST set the value of the NotOnOrAfter attribute of the message to a time value, indicating an expiration time for the message.

In addition to the values specified in Section 3.6.3for the Reason attribute, the following values are also available for use by the session authority only:

urn:oasis:names:tc:SAML:2.0:logout:global-timeout

Specifies that the message is being sent because of the global session timeout interval period being exceeded.

urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

Specifies that the message is being sent because a timeout interval period agreed between a participant and the authority has been exceeded.

3.8 Name Identifier Mapping Protocol

When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name identifier for the same principal in a particular format or federation namespace, it can send a request to the identity provider using this protocol.

For example, a service provider that wishes to communicate with another service provider with whom it does not share an identifier for the principal can use an identity provider that shares an identifier for the principal with both service providers to map from its own identifier to a new identifier, generally encrypted, with which it can communicate with the second service provider.

Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a `<saml:EncryptedID>` element unless a specific deployment dictates such protection is unncessary.

3.8.1 Element <NameIDMappingRequest>

To request an alternate name identifier for a principal from an identity provider, a requester sends an `<NameIDMappingRequest>` message. This message has the complex type **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

`<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

> The identifier and associated descriptive data that specify the principal as currently recognized by the requester and the responder.

`<NameIDPolicy>` [Required]

> The requirements regarding the format and optional name qualifier for the identifier to be returned.

The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

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

```
<element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
<complexType name="NameIDMappingRequestType">
        <complexContent>
                <extension base="samlp:RequestAbstractType">
                        <sequence>
                                <choice>
                                        <element ref="saml:BaseID"/>
                                        <element ref="saml:NameID"/>
                                        <element ref="saml:EncryptedID"/>
                                </choice>
                                <element ref="samlp:NameIDPolicy"/>
                        </sequence>
                </extension>
        </complexContent>
</complexType>
```

3.8.2 Element <NameIDMappingResponse>

The recipient of a `<NameIDMappingRequest>` message MUST respond with a `<NameIDMappingResponse>` message. This message has the complex type **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following element:

`<saml:NameID>` or `<saml:EncryptedID>` [Required]

> The identifier and associated attributes that specify the principal in the manner requested, usually in encrypted form.

The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

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

```
<element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
<complexType name="NameIDMappingResponseType">
```

```
        <complexContent>
                <extension base="samlp:StatusResponseType">
                        <choice>
                                <element ref="saml:NameID"/>
                                <element ref="saml:EncryptedID"/>
                        </choice>
                </extension>
        </complexContent>
</complexType>
```

3.8.3 Processing Rules

If the responder does not recognize the principal identified in the request, it MAY respond with an error `<Status>` containing a second-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal`.

At the responder's discretion, the `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy` status code MAY be returned to indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

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

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. 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 documents and XML schema documents. 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 documents 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 syntax, protocols, and semantics remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major revision.

4.1.1 Schema Version

As a non-normative documentation mechanism, any XML schema documents published as part of the specification set will contain a `version` attribute on the `<xsd:schema>` element whose value is 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 multiple 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 target XML namespace specified for the schema definitions for that assertion version.

The following processing rules apply:

- A SAML authority MUST NOT issue any assertion with an overall *Major.Minor* 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. For example, 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. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

4.1.3 SAML Protocol Version

The various SAML protocols' 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 target XML namespace specified for the schema definitions for that protocol version.

The version numbers used in SAML protocol request and response elements will match 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 the responder supports requests with the highest request version supported by the requester.
- A SAML requester MUST NOT issue a request message with an overall *Major.Minor* 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. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

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 `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-level `<StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting

one of the following second-level values:
`urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`,
`urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow`, or
`urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated`.

4.1.5 Permissible Version Combinations

Assertions of a particular major version appear only in response messages of the same major version, as permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1 response message, if the appropriate assertion schema is referenced during namespace importation. But a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major versions.

4.2 SAML Namespace Version

XML schema documents 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 syntactic definitions that make up that part of the specification.

The namespace URI references 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. (This pattern is also followed by the SAML-defined URI-based identifiers that are listed in Section 8.)

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 to 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 7. 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, and, if the signature is based on the SAML authority's public-private key pair, 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, and, if the signature is based on the originator's public-private key pair, non-repudiation of origin.

A digital signature is not always required in SAML. For example, 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. As another example, 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, and so on. 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 sender SHOULD be signed by the sender.

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 this specification attempts to ensure compatibility with profiles that may require other mechanisms.

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

5.1 Signing Assertions

All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as described in Section 2.

5.2 Request/Response Signing

All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected in the schema as described in Section 3.

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 [XMLSig] calls out a general XML syntax for signing data with flexibility and many choices. This section details 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, specifically the `ID` attribute on `<Assertion>` and the various request and response elements. These 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 enclosing root element. The assertion's or protocol message's root element may or may not be the root element of the actual XML document containing the signed assertion or protocol 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 [Excl-C14N], 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 the semantics of statements within assertions in any way, as stated in Section 2.

5.4.7 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.

TODO: Update example.

```
<Response
  IssueInstant="2003-04-17T00:46:02Z"
  MajorVersion="2"
  MinorVersion="0"
  Recipient="www.opensaml.org"
  ID="_c7055387-af61-4fce-8b98-e2927324b306"
  xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0: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="urn:oasis:names:tc:SAML:2.0:status:Success"/></Status>
<Assertion
  ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
  IssueInstant="2003-04-17T00:46:02Z"
  Issuer="www.opensaml.org"
  MajorVersion="2"
  MinorVersion="0"
  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">
<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"/>
<Conditions
  NotBefore="2003-04-17T00:46:02Z"
  NotOnOrAfter="2003-04-17T00:51:02Z">
<AudienceRestriction><Audience>http://www.opensaml.org</Audience>
</AudienceRestriction></Conditions>
<AuthnStatement
  AuthnInstant="2003-04-17T00:46:00Z"
  AuthnMethod="urn:oasis:names:tc:SAML:1.0:am:password">
</AuthnStatement>
<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 and XML Encryption Syntax and Processing

Encryption is used as the means to implement confidentiality. The most common motives for confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for competitive advantage or similar reasons. Confidentiality may also be required to insure the effectiveness of some other security mechanism. For example, a secret password or key may be encrypted.

Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- Communications confidentiality may be provided by mechanisms associated with a particular binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS or SOAP Message Security mechanisms for confidentiality.
- A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>` element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.
- An entire assertion may be encrypted, as described in Section 2.3.4.
- The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.3.
- An `<Attribute>` element may be encrypted, as described in Section 2.4.3.3.

6.1 General Considerations

Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's `Type` attribute SHOULD be used and, if it is present, MUST have the value `http://www.w3.org/2001/04/xmlenc#Element`.

Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

6.2 Combining Signatures and Encyption

Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in the reverse order that signing and encryption were performed.

- When the entire assertion is encrypted, the signature MUST first be calculated and in place, and then the element encrypted.
- When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be performed first and then the signature calculated over the assertion or message containing the encrypted element.

6.3 Examples

The following example shows an encrypted assertion in a response message:

TBD

The following example shows an encrypted name identifier.

TBD

7 SAML Extensibility

SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas. An example of an application that extends SAML assertions is the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions and protocols.

See the SAML Profiles specification [SAMLProf] for information on how to define new profiles of use, which can be combined with extensions to put the SAML framework to new uses.

7.1 Schema Extension

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.

7.1.1 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:

- `<BaseID>` and **BaseIDAbstractType**
- `<Condition>` and **ConditionAbstractType**
- `<Statement>` and **StatementAbstractType**

The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:

- `<AuthnStatement>` and **AuthnStatementType**
- `<AttributeStatement>` and **AttributeStatementType**
- `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- `<AudienceRestriction>` and **AudienceRestrictionType**
- `<ProxyRestriction>` and **ProxyRestrictionType**
- `<OneTimeUse>` and **OneTimeUseType**

7.1.2 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:

- `<Request>` and **RequestAbstractType**
- `<SubjectQuery>` and **SubjectQueryAbstractType**

The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:

- <AuthnQuery> and **AuthnQueryType**
- <AuthzDecisionQuery> and **AuthzDecisionQueryType**
- <AttributeQuery> and **AttributeQueryType**

7.2 Schema Wildcard Extension Points

The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes from arbitrary namespaces, which serves as a built-in extension point without requiring an extension schema.

7.2.1 Assertion Extension Points

The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- <SubjectConfirmationData>: Uses **xsd:anyType**, which allows any sub-elements and attributes.
- <AuthnContextDecl>: Uses **xsd:anyType**, which allows any sub-elements and attributes.
- <AttributeValue>: Uses **xsd:anyType**, which allows any sub-elements and attributes.
- <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other namespaces with lax schema validation processing.

The following constructs in the assertion schema allow arbitrary global attributes:

- <AttributeDesignator> and **AttributeDesignatorType**
- <Attribute> and **AttributeType** (based on **AttributeDesignatorType**)

7.2.2 Protocol Extension Points

The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema validation processing.
- <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax schema validation processing.
- <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with lax schema validation processing. (It is specifically intended to carry a SAML request or response message element, however.)

7.3 Identifier Extension

SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8. However, it is always possible to define additional URI-based identifiers for these purposes. It is RECOMMENDED that these additional identifiers be defined in a formal profile of use.

8 SAML-Defined Identifiers

The following sections define URI-based identifiers for common resource access actions, subject name identifier formats, and attribute name 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, according to the specification set version in which they were first introduced:

```
urn:oasis:names:tc:SAML:1.0:
urn:oasis:names:tc:SAML:1.1:
urn:oasis:names:tc:SAML:2.0:
```

8.1 Action Namespace Identifiers

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

8.1.1 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.

8.1.2 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 8.1.1are 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.

8.1.3 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.

8.1.4 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.

8.2 Attribute Name Format Identifiers

The following identifiers MAY be used in the `NameFormat` attribute defined on the **AttributeDesignatorType** complex type to refer to the classification of the attribute name for purposes of interpreting the name.

8.2.1 Unspecified

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified`

The interpretation of the attribute name is left to individual implementations.

8.2.2 URI Reference

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`

The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-specific. See [SAMLProf] for attribute profiles that make use of this identifier.

8.2.3 Basic

URI: `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`

The class of strings acceptable as the attribute name MUST be drawn from the set of values belonging to the primitive type **xsd:Name** as defined in [Schema2] §3.3.6 . See [SAMLProf] for attribute profiles that make use of this identifier.

8.3 Name Identifier Format Identifiers

The following identifiers MAY be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the associated processing rules, if any.

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

8.3.1 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.

8.3.2 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 ">".

8.3.3 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 [XMLSig]. 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].

8.3.4 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.

8.3.5 Kerberos Principal Name

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`

Indicates that the content of the element is in the form of a Kerberos principal name using the format `name[/instance]@REALM`. The syntax, format and characters allowed for the name, instance, and realm are described in [RFC 1510].

8.3.6 Entity Identifier

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`

Indicates that the content of the element is the identifier of an entity that provides SAML-based services (such as a SAML authority) or is a participant in SAML profiles (such as a service provider supporting the browser SSO profile). Such an identifier can be used in the `<Issuer>` element to identify the issuer of a SAML request, response, or assertion, or within the `<NameID>` element to make assertions about system entities that can issue SAML requests, responses, and assertions. It can also be used in other elements and attributes whose purpose is to identify a system entity in various protocol exchanges.

The syntax of such an identifier is a URI of not more than 1024 characters in length. It is RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

8.3.7 Persistent Identifier

URI: `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`

Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to an identity provider and a service provider or affiliation of service providers. Persistent 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, pair-wise pseudonym to prevent the discovery of the subject's identity or activities. Persistent name identifier values MUST NOT exceed a length of 256 characters.

The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity provider that generated the identifier (see Section 8.3.6). 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 unique identifier of the service provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). 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 `SPProvidedID` attribute MUST contain the alternative identifier of the principal most recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been established, than the attribute MUST be omitted.

Persistent 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 shared identifier. 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, but MUST NOT overload this format with persistent but non-opaque values

Note also that while persistent identifiers are typically used to reflect an account linking relationship between a pair of providers, a service provider is not obligated to recognize or make use of the long term nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly different and does not affect the behavior of the identity provider or any processing rules specific to persistent identifiers in the protocols defined in this specification.

8.3.8 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 pair-wise identifier. In such a case, they MAY be omitted in accordance with the rules specified in Section 8.3.7.

8.4 Consent Identifiers

The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** complex type to communicate whether a user gave consent, and under what conditions, for the request.

8.4.1 Unspecified

URI: `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

No claim as to user consent is being made.

8.4.2 Obtained

URI: `urn:oasis:names:tc:SAML:2.0:consent:obtained`

Indicates that a user’s consent has been obtained by the issuer of the request.

8.4.3 Prior

URI: `urn:oasis:names:tc:SAML:2.0:consent:prior`

Indicates that a user’s consent has been obtained by the issuer of the request at some point prior to the action that initiated the request.

8.4.4 Implicit

URI: `urn:oasis:names:tc:SAML:2.0:consent:current-implicit`

Indicates that a user’s consent has been implicitly obtained by the issuer of the request during the action that initiated the request, as part of a broader indication of consent. Implicit consent is typically more proximal to the action in time and presentation than prior consent, such as part of a session of activities.

8.4.5 Explicit

URI: `urn:oasis:names:tc:SAML:2.0:consent:current-explicit`

Indicates that a user’s consent has been explicitly obtained by the issuer of the request during the action that initiated the request.

8.4.6 Unavailable

URI: `urn:oasis:names:tc:SAML:2.0:consent:unavailable`

Indicates that the issuer of the request did not obtain consent.

8.4.7 Inapplicable

URI: `urn:oasis:names:tc:SAML:2.0:consent:inapplicable`

Indicates that the issuer of the request does not believe that they need to obtain or report consent.

9 References

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

9.1 Normative References

[Excl-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 [XMLEnc-XSD], 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.

9.2 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.

[SAMLAuthnCxt] J. Kemp. Authentication Context for the *OASIS Security Assertion Markup Language (SAML)*. OASIS, February 2004. Document ID sstc-saml-authn-context-2.0. http://www.oasis-open.org/committees/security/.

[SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0. http://www.oasis-open.org/committees/security/.

[SAMLProf] E. Maler et al. Profiles *for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0. http://www.oasis-open.org/committees/security/.

[SAMLMetadata] E. Maler et al. Metadata *for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-metadata-2.0. 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-2.0. 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-2.0. 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-2.0. 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-2.0. 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-2.0. http://www.oasis-open.org/committees/security/.

[SAML-TechOvw] J. Hughes et al. SAML Technical Overview. OASIS, July 2004. Document ID oasis-sstc-saml-tech-overview-2.0. http://www.oasis-open.org/committees/security/.

[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: Fundamentals.* World Wide Web Consortium, February 2004. http://www.w3.org/TR/charmod/.

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

[XACML] eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

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:

- @@

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.
http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf			
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).
http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf			
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.
http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf			
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.
http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf			
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol (**W-2**), 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 (**W-1**); still unfinished.
http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf			
06	20 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added AssertionURIReference (**W-19**), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response (**W-1**). Implemented the freezing of authZ decision statement functionality (**W-28b**).
http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf			

Rev	Date	By Whom	What
07	7 Mar 2004	Scott Cantor, Eve Maler	Implemented new arrangement for subject information and decision on KeyInfo description, as agreed at 2 Mar 2004 telecon. Adjusted normative language around subject "matching" rules based on subject changes. Revised AuthnRequest proposal based on those changes and feedback from list and focus calls. Incorporated additional schema and processing rules related to ECP and proxying use cases from ID-FF. Added AuthnContext to AuthenticationStatement. Added NameIdentifierMapping protocol (**W-2**).
http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf			
08	15 Mar 2004	Scott Cantor, Eve Maler	Added ArtifactRequest/Response pair as a new protocol. Implemented proposed W-28a attribute changes (rev 03 of the proposal, reflecting focus group input).
http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf			
09	8 Apr 2004	Eve Maler	Minor cleanup, plus decisions from March-April 2004 F2F meeting: Moved Signature element up in Assertion contents. Clarified that DoNotCacheCondition has one-time-use semantics. Made NameFormat on the Attribute element clearly optional. Changed the default ValueType identifier name. Added the ability to put arbitrary attributes on the AttributeDesignator element. Removed Source on the Attribute element. Changed the content of Extensions in the Request element to ##other. Removed the restriction saying only federated identifiers could be replaced and set with the termination protocol. Changed Reason on the LogoutRequest element to be a URI reference. Made SessionIndex in the LogoutRequest element globally declared. Added bibliographic references to the new SAML specs.
http://www.oasis-open.org/committees/download.php/6323/sstc-saml-core-2.0-draft-09-diff.pdf			
10	12 Apr 2004	Scott Cantor, Eve Maler	Allowed assertions to be subjectless. Allowed Audience to reference a specific provider URI. Changed AuthnMethodand AuthnContext handling. Removed RelayState. Added AllowCreate on NameIDPolicy. Consolidated two protocols into the name identifier management protocol. Added a name identifier URI for Kerberos principals.
http://www.oasis-open.org/committees/download.php/6347/sstc-saml-core-2.0-draft-10-diff.pdf			
11	11 May 2004	Eve Maler	Updated the wording describing the permissible combinations of assertion vs. protocol versions (issue TECH-2). Removed the proposed ValueType field on AttributeDesignator; how to do this will be described in the Baseline Attributes spec instead.
http://www.oasis-open.org/committees/download.php/6709/sstc-saml-core-2.0-draft-11-diff.pdf			
12	17 May 2004	Scott Cantor, Eve Maler	Added ReauthenticateOnOrAfter, shortened various elements and attributes per TC decision, revised text around entity/provider and persistent identifiers, added additional schema and discussion on encryption, turned subject confirmation method into an attribute.

Rev	Date	By Whom	What
http://www.oasis-open.org/committees/download.php/6791/sstc-saml-core-2.0-draft-12-diff.pdf			
13	20 May 2004	Eve Maler, Scott Cantor	Truncated condition subtype names.
http://www.oasis-open.org/committees/download.php/6857/sstc-saml-core-2.0-draft-13-diff.pdf			
14	30 May 2004	Scott Cantor	Various schema enhancements for encryption of NameID, removed AuthnMethod, moved Session attributes, enhanced SubjectConfirmationData with generally useful attributes, added KeyInfoConfirmationDataType, some rewording of persistent NameID format
http://www.oasis-open.org/committees/download.php/6998/sstc-saml-core-2.0-draft-14-diff.pdf			
15	30 June 2004	Scott Cantor	Near final schema cleanup, and feedback from F2F.
http://www.oasis-open.org/committees/download.php/7473/sstc-saml-core-2.0-draft-15-diff.pdf			
16	12 July 2004	Eve Maler, Scott Cantor	Copyediting, clarifications on usage of various elements, addition of Consent values derived from ID-FF 1.2 submission.
http://www.oasis-open.org/committees/download.php/7711/sstc-saml-core-2.0-draft-16-diff.pdf			
17	13 Jul 2004	Eve Maler	Final title-page cleanup for last-call working draft publication.

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.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.