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

OASIS Standard, 15 March 2005

Document identifier:
saml-core-2.0-os

Location:
http://docs.oasis-open.org/security/saml/v2.0/

Editors:
Scott Cantor, Internet2
John Kemp, Nokia
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:
Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz Allen Hamilton
Thomas Wisniewski, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Nick Ragouzis, Individual
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Peter C Davis, Neustar
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Paul Madsen, NTT
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems

Eve Maler, Sun Microsystems
Ron Monzillo, Sun Microsystems
Greg Whitehead, Trustgenix

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 an **OASIS Standard** document produced by the Security Services Technical Committee. It was approved by the OASIS membership on 1 March 2005.

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 on its web page (http://www.oasis-open.org/committees/security) a catalog of any changes made to this document as a result of comments.

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

The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This specification defines both the structure of SAML assertions, and an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML [XML] and use XML namespaces [XMLNS]. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. The SAML bindings specification [SAMLBind] provides frameworks for the embedding and transport of SAML protocol messages. The SAML profiles specification [SAMLProf] provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

For additional explanation of SAML terms and concepts, refer to the SAML technical overview [SAMLTechOvw] and the SAML glossary [SAMLGloss] . Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMLP-XSD] are also available. The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

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

1.1 Notation

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].

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

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

Note: Notes like this are sometimes used to highlight non-normative commentary.

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. In cases of disagreement between the SAML schema documents 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].

Prefix	XML Namespace	Comments
`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].
`xs:`	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. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.
`xsi:`	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

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. See Section 4.2 for information on SAML namespace versioning.

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#
```

Finally, the schema for XML Encryption [XMLEnc] is imported into the assertion schema and is associated with the following XML namespace:

```
http://www.w3.org/2001/04/xmlenc#
```

1.3 Common Data Types

The following sections define how to use and interpret common data types that appear throughout the SAML schemas.

1.3.1 String Values

All SAML string values have the type **xs:string**, which is built in to the W3C XML Schema Datatypes specification [Schema2]. Unless otherwise noted in this specification or particular profiles, all strings in SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the XML Recommendation [XML] Section 2.3).

Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that have the XML Schema **xs: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] Section 2.11. XML attribute values defined as strings (or types derived from strings) are normalized as described in [XML] Section 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 XML 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.

1.3.2 URI Values

All SAML URI reference values have the type **xs:anyURI**, which is built in to the W3C XML Schema Datatypes specification [Schema2].

Unless otherwise indicated in this specification, all URI reference values used within SAML-defined elements or attributes MUST consist of at least one non-whitespace character, and are REQUIRED to be absolute [RFC 2396].

Note that the SAML specification makes extensive use of URI references as identifiers, such as status codes, format types, attribute and system entity names, etc. In such cases, it is essential that the values be both unique and consistent, such that the same URI is never used at different times to represent different underlying information.

1.3.3 Time Values

All SAML time values have the type **xs: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 time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

1.3.4 ID and ID Reference Values

The **xs:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type **xs:ID** in this specification MUST satisfy the following properties in addition to those imposed by the definition of the **xs: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 random or 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 **xs:ID** datatype. A pseudorandom generator MUST be seeded with unique material in order to ensure the desired uniqueness properties between different systems.

The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs:IDREF** cannot be used for this purpose. 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. Using **xs:IDREF** would violate the requirement that its value match the value of an ID attribute on some element in the same XML document.

> **Note:** It is anticipated that the World Wide Web Consortium will standardize a global attribute for holding ID-typed values, called `xml:id` [XML-ID]. The Security Services Technical Committee plans to move away from SAML-specific ID attributes to this style of assigning unique identifiers as soon as practicable after the `xml:id` attribute is standardized.

2 SAML Assertions

An assertion is a package of information that supplies zero or more statements made by a **SAML authority**; SAML authorities are sometimes referred to as **asserting parties** in discussions of assertion generation and exchange, and system entities that use received assertions are known as **relying parties**. (Note that these terms are different from **requester** and **responder**, which are reserved for discussions of SAML protocol message exchange.)

SAML assertions are usually made about a **subject**, represented by the `<Subject>` element. However, the `<Subject>` element is optional, and other specifications and profiles may utilize the SAML assertion structure to make similar statements without specifying a subject, or possibly specifying the subject in an alternate way. Typically there are a number of **service providers** that can make use of assertions about a subject in order to control access and provide customized service, and accordingly they become the relying parties of an asserting party called an **identity provider**.

This SAML specification defines three different kinds of assertion statements that can be created by a SAML authority. All SAML-defined statements are associated with a subject. The three kinds of statement defined in this specification are:

- **Authentication:** The assertion subject was authenticated by a particular means at a particular time.
- **Attribute:** The assertion subject is associated with the supplied attributes.
- **Authorization Decision:** A request to allow the assertion 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.

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 assertions and statements.

The SAML technical overview [SAMLTechOvw] and glossary [SAMLGloss] provide more detailed explanation of SAML terms and concepts.

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/2002/REC-xmldsig-core-
20020212/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: saml-schema-assertion-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard Schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New assertion schema for SAML V2.0 namespace.
        </documentation>
    </annotation>
…
</schema>
```

2.2 Name Identifiers

The following sections define the SAML constructs that contain descriptive identifiers for subjects and the issuers of assertions and protocol messages.

There are a number of circumstances in SAML in which it is useful for two system entities to communicate regarding a third party; for example, the SAML authentication request protocol enables third-party authentication of a subject. Thus, it is useful to establish a means by which parties may be associated with identifiers that are meaningful to each of the parties. In some cases, it will be necessary to limit the scope within which an identifier is used to a small set of system entities (to preserve the privacy of a subject, for example). Similar identifiers may also be used to refer to the issuer of a SAML protocol message or assertion.

It is possible that two or more system entities may use the same name identifier value when referring to different identities. Thus, each entity may have a different understanding of that same name. SAML provides **name qualifiers** to disambiguate a name identifier by effectively placing it in a federated **namespace** related to the name qualifiers. SAML V2.0 allows an identifier to be qualified in terms of both an asserting party and a particular relying party or affiliation, allowing identifiers to exhibit pair-wise semantics, when required.

Name identifiers may also be encrypted to further improve their privacy-preserving characteristics, particularly in cases where the identifier may be transmitted via an intermediary.

> **Note:** To avoid use of relatively advanced XML schema constructs (among other reasons), the various types of identifier elements do not share a common type hierarchy.

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 includes the following attributes for use by extended identifier representations:

`NameQualifier` [Optional]

> The security or administrative domain that qualifies the identifier. 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 `NameQualifier` and `SPNameQualifier` attributes SHOULD be omitted unless the identifier's type definition explicitly defines their use and semantics.

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

```
<attributeGroup name="IDNameQualifiers">
   <attribute name="NameQualifier" type="string" use="optional"/>
   <attribute name="SPNameQualifier" type="string" use="optional"/>
</attributeGroup>
<element name="BaseID" type="saml:BaseIDAbstractType"/>
<complexType name="BaseIDAbstractType" abstract="true">
   <attributeGroup ref="saml:IDNameQualifiers"/>
</complexType>
```

2.2.2 Complex Type NameIDType

The **NameIDType** complex type is used when an element serves to represent an entity by a string-valued name. It is a more restricted form of identifier than the `<BaseID>` element and is the type underlying both the `<NameID>` and `<Issuer>` elements. In addition to the string content containing the actual identifier, it provides the following optional attributes:

`NameQualifier` [Optional]

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

`SPNameQualifier` [Optional]

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

`Format` [Optional]

> A URI reference representing the classification of string-based identifier information. See Section 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute and their associated descriptions and processing rules. Unless otherwise specified by an element based on this type, if no `Format` value is provided, then the value `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in effect.
>
> When a `Format` value other than one specified in Section 8.3 is used, the content of an element of this type 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 a service provider or affiliation of providers for the entity, if different from the primary name identifier given in the content of the element. This attribute provides a means of integrating the use of SAML with existing identifiers already in use by a service provider. For example, an existing identifier can be "attached" to the entity using the Name Identifier Management protocol defined in Section 3.6.

Additional rules for the content of (or the omission of) these attributes can be defined by elements that make use of this type, and by specific `Format` definitions. The `NameQualifier` and `SPNameQualifier` attributes SHOULD be omitted unless the element or format explicitly defines their use and semantics.

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

```
<complexType name="NameIDType">
   <simpleContent>
```

```
            <extension base="string">
                <attributeGroup ref="saml:IDNameQualifiers"/>
                <attribute name="Format" type="anyURI" use="optional"/>
                <attribute name="SPProvidedID" type="string" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
```

2.2.3 Element <NameID>

The <NameID> element is of type **NameIDType** (see Section 2.2.2), and is used in various SAML assertion constructs such as the <Subject> and <SubjectConfirmation> elements, and in various protocol messages (see Section 3).

The following schema fragment defines the <NameID> element:

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

2.2.4 Element <EncryptedID>

The <EncryptedID> element is of type **EncryptedElementType**, and carries the content of an unencrypted identifier element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The <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 of **NameIDType** or **AssertionType**, or a type that is derived from **BaseIDAbstractType**, **NameIDType**, or **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 mechanism 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] Section 6.3.

Note that an entire assertion can be encrypted into this element and used as an identifier. In such a case, the <Subject> element of the encrypted assertion supplies the "identifier" of the subject of the enclosing assertion. Note also that if the identifying assertion is invalid, then so is the enclosing assertion.

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

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

2.2.5 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 (see Section 2.2.2).

Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see Section 8.3.6).

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 either contain assertion information or provide a means to refer to an existing assertion.

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. See Section 3.3.1 for a protocol element that uses such a reference to ask for the corresponding assertion.

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. The URI reference MAY be used to retrieve the corresponding assertion in a manner specific to the URI reference. See Section 3.7 of the Bindings specification [SAMLBind] for information on how this element is used in a protocol binding to accomplish this.

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:

`Version` [Required]

The version of this assertion. The identifier for the version of SAML defined in this specification is "2.0". SAML versioning is discussed in Section 4.

`ID` [Required]

The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness.

`IssueInstant` [Required]

The time instant of issue in UTC, as described in Section 1.3.3.

`<Issuer>` [Required]

The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be unambiguous to the intended relying parties.

This specification defines no particular 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 protects the integrity of and authenticates the issuer of the assertion, as described below and in Section 5.

`<Subject>` [Optional]

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

`<Conditions>` [Optional]

Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion. See Section 2.5 for additional information on how to evaluate conditions.

`<Advice>` [Optional]

Additional information related to the assertion that assists processing in certain situations but which MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

Zero or more of the following statement elements:

`<Statement>`

A statement of a type defined in an extension schema. An `xsi:type` attribute MUST be used to indicate the actual statement type.

`<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 `<Subject>` is omitted, then the statements in the assertion apply to a subject or subjects identified in an application- or profile-specific manner. SAML itself defines no such statements, and an assertion without a subject has no defined meaning in this specification.

Depending on the requirements of particular protocols or profiles, the issuer of a SAML assertion may often need to be authenticated, and integrity protection may often be required. Authentication and message integrity MAY be provided by mechanisms provided by a protocol binding in use during the delivery of an assertion (see [SAMLBind]). The SAML assertion MAY be signed, which provides both authentication of the issuer and integrity protection.

If such a signature is used, then the `<ds:Signature>` element MUST be present, and a relying party MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity and appropriateness of the issuer and may continue to process the assertion in accordance with this specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-specific rules, and so on).

Note that whether signed or unsigned, the inclusion of multiple statements within a single assertion is semantically equivalent to a set of assertions containing those statements individually (provided the subject, conditions, etc. are also the same).

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="Version" type="string" use="required"/>
   <attribute name="ID" type="ID" use="required"/>
   <attribute name="IssueInstant" type="dateTime" use="required"/>
</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 of or 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 mechanism when the plain-text value passes through an intermediary.

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

```
<element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
```

2.4 Subjects

This section defines the SAML constructs used to describe the subject of an assertion.

2.4.1 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 an 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 satisfying any one of them is sufficient to confirm the subject for the purpose of applying the assertion.

A `<Subject>` element can contain both an identifier and zero or more subject confirmations which a relying party can verify when processing an assertion. If any one of the included subject confirmations are verified, the relying party MAY treat the entity presenting the assertion as one that the asserting party has associated with the principal identified in the name identifier and associated with the statements in the assertion. This attesting entity and the actual subject may or may not be the same entity.

If there are no subject confirmations included, then any relationship between the presenter of the assertion and the actual subject is unspecified.

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>
```

2.4.1.1 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 or mechanism to be used to confirm the subject. URI references identifying SAML-defined confirmation methods are currently defined in the SAML profiles specification [SAMLProf]. Additional methods MAY be added by defining new URIs and profiles or by private agreement.

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

    Identifies the entity expected to satisfy the enclosing subject confirmation requirements.

`<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 (See also Section 2.4.1.3). 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>
      <choice minOccurs="0">
         <element ref="saml:BaseID"/>
         <element ref="saml:NameID"/>
         <element ref="saml:EncryptedID"/>
      </choice>
      <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
   </sequence>
   <attribute name="Method" type="anyURI" use="required"/>
</complexType>
```

2.4.1.2 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 act of subject confirmation can take place. Subject confirmation takes place when a relying party seeks to verify the relationship between an entity presenting the assertion (that is, the attesting entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply to any method:

`NotBefore` [Optional]

A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as described in Section 1.3.3.

`NotOnOrAfter` [Optional]

A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as described in Section 1.3.3.

`Recipient` [Optional]

A URI specifying the entity or location to which an attesting entity can present the assertion. For example, this attribute might indicate that the assertion must be delivered to a particular network endpoint in order to prevent an intermediary from redirecting it someplace else.

`InResponseTo` [Optional]

The `ID` of a SAML protocol message in response to which an attesting entity can present the assertion. For example, this attribute might be used to correlate the assertion to a SAML request that resulted in its presentation.

`Address` [Optional]

The network address/location from which an attesting entity can present the assertion. For example, this attribute might be used to bind the assertion to particular client addresses to prevent an attacker from easily stealing and presenting the assertion from another location. IPv4 addresses SHOULD be represented in the usual dotted-decimal format (e.g., "`1.2.3.4`"). IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

Arbitrary attributes

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

Arbitrary elements

> This complex type uses an `<xs:any>` extension point to allow arbitrary XML elements to be added to `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This allows additional elements to be added as needed to supply additional confirmation-related information.

Particular confirmation methods and profiles that make use of those methods MAY require the use of one or more of the attributes defined within this complex type. For examples of how these attributes (and subject confirmation in general) can be used, see the Profiles specification [SAMLProf].

Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present, 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>
      <restriction base="anyType">
         <sequence>
            <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
         <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"/>
         <anyAttribute namespace="##other" processContents="lax"/>
      </restriction>
   </complexContent>
</complexType>
```

2.4.1.3 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 an attesting entity. The particular confirmation method MUST define the exact mechanism by which the confirmation data can be used. The optional attributes defined by the **SubjectConfirmationDataType** complex type MAY also appear.

This complex type, or a type derived from it, 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>
```

2.4.1.4 Example of a Key-Confirmed <Subject>

To illustrate the way in which the various elements and types fit together, below is an example of a `<Subject>` element containing a name identifier and a subject confirmation based on proof of possession of a key. Note the use of the **KeyInfoConfirmationDataType** to identify the confirmation data syntax as being a `<ds:KeyInfo>` element:

```
<Subject>
   <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
   scott@example.org
   </NameID>
   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
         <ds:KeyInfo>
            <ds:KeyName>Scott's Key</ds:KeyName>
         </ds:KeyInfo>
      </SubjectConfirmationData>
   </SubjectConfirmation>
</Subject>
```

2.5 Conditions

This section defines the SAML constructs that place constraints on the acceptable use of SAML assertions.

2.5.1 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 1.3.3.

`NotOnOrAfter` [Optional]

   Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as described in Section 1.3.3.

`<Condition>` [Any Number]

   A condition of a type defined in an extension schema. An `xsi:type` attribute MUST be used to indicate the actual condition type.

`<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 subsequently act as asserting parties themselves and issue 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.

Because the use of the `xsi:type` attribute would permit an assertion to contain more than one instance of a SAML-defined subtype of **ConditionsType** (such as **OneTimeUseType**), the schema does not explicitly limit the number of times particular conditions may be included. A particular type of condition MAY define limits on such use, as shown above.

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>
```

2.5.1.1 General Processing Rules

If an assertion contains a `<Conditions>` element, then the validity of the assertion is dependent on the sub-elements and attributes provided, using the following rules in the order shown below.

Note that an assertion that has condition validity status ***Valid*** may nonetheless be untrustworthy or invalid for reasons such as not being well-formed or schema-valid, 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.

1. If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is considered to be ***Valid*** with respect to condition processing.
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, or if an element is encountered that is not understood, 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*** with respect to condition processing.

The first rule that applies terminates condition processing; thus a determination that an assertion is ***Invalid*** takes precedence over that of ***Indeterminate***.

An assertion that is determined to be ***Invalid*** or ***Indeterminate*** MUST be rejected by a relying party (within whatever context or profile it was being processed), just as if the assertion were malformed or otherwise unusable.

2.5.1.2 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 correct or accurate 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, then it is considered unspecified. If the `NotBefore` attribute is unspecified (and if all other conditions that are supplied evaluate to ***Valid***), then the assertion is ***Valid*** with respect to conditions at any time before the time instant specified by the `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if all other conditions that are supplied evaluate to ***Valid***), the assertion is ***Valid*** with respect to conditions 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*** with respect to conditions at any time.

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

2.5.1.3 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"/>
```

2.5.1.4 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 asserting party 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 URI from a SAML name identifier that describes a system entity (see Section 8.3.6).

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 asserting party 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 asserting party 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 effect of this requirement and the preceding definition is that within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a conjunction (an "AND").

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"/>
```

2.5.1.5 Element <OneTimeUse>

In general, relying parties may choose to retain assertions, or the information they contain in some other form, for reuse. 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 and will be combined with possible transmission delays, there is no convenient way for 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.

To support the single use constraint, a relying party should maintain a cache of the assertions it has processed containing such a condition. Whenever an assertion with this condition is processed, the cache should be checked to ensure that the same assertion has not been previously received and processed by the relying party.

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. That is, this condition does not affect validity but is a condition on use.

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>
```

2.5.1.6 Element <ProxyRestriction>

Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting parties and issue subsequent assertions of their own on the basis of the information contained in the original assertion. A relying party acting as an asserting 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 maximum number of indirections that the asserting party permits to 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 the asserting party permits new assertions to 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. That is, this condition does not affect validity but is a condition on use.

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>
```

2.6 Advice

This section defines the SAML constructs that contain additional information about an assertion that an asserting party wishes to provide to a relying party.

2.6.1 Element <Advice>

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

The `<Advice>` element contains a mixture of zero or more `<Assertion>`, `<EncryptedAssertion>`, `<AssertionIDRef>`, and `<AssertionURIRef>` elements, and namespace-qualified elements in other non-SAML namespaces.

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>
```

2.7 Statements

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

2.7.1 Element <Statement>

The `<Statement>` element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. SAML itself derives its core statements from this extension point. 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"/>
```

2.7.2 Element <AuthnStatement>

The `<AuthnStatement>` element describes a statement by the SAML authority asserting that the assertion subject was authenticated by a particular means at a particular time. Assertions containing `<AuthnStatement>` elements MUST contain a `<Subject>` element.

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 1.3.3.

`SessionIndex` [Optional]

Specifies the index of a particular session between the principal identified by the subject and the authenticating authority.

`SessionNotOnOrAfter` [Optional]

Specifies a time instant at which the session between the principal identified by the subject and the SAML authority issuing this statement MUST be considered ended. The time value is encoded in UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a `NotOnOrAfter` condition attribute that may be present in the assertion.

`<SubjectLocality>` [Optional]

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

`<AuthnContext>` [Required]

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

In general, any string value MAY be used as a `SessionIndex` value. However, when privacy is a consideration, care must be taken to ensure that the `SessionIndex` value does not invalidate other privacy mechanisms. Accordingly, the value SHOULD NOT be usable to correlate activity by a principal across different session participants. Two solutions that achieve this goal are provided below and are RECOMMENDED:

- Use small positive integers (or reoccurring constants in a list) for the `SessionIndex`. The SAML authority SHOULD choose the range of values such that the cardinality of any one integer will be sufficiently high to prevent a particular principal's actions from being correlated across multiple session participants. The SAML authority SHOULD choose values for `SessionIndex` randomly from within this range (except when required to ensure unique values for subsequent statements given to the same session participant but as part of a distinct session).
- Use the enclosing assertion's `ID` value in the `SessionIndex`.

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>
```

2.7.2.1 Element <SubjectLocality>

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

`Address` [Optional]

 The network address of the system from which the principal identified by the subject was authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "`1.2.3.4`"). IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

`DNSName` [Optional]

 The DNS name of the system from which the principal identified by 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>
```

2.7.2.2 Element <AuthnContext>

The `<AuthnContext>` element specifies the context of an authentication event. The element can contain 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 (not including the assertion issuer, who is presumed to have been involved without being explicitly named here).

See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication context information.

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

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
   <sequence>
      <choice>
         <sequence>
            <element ref="saml:AuthnContextClassRef"/>
            <choice minOccurs="0">
```

```
                <element ref="saml:AuthnContextDecl"/>
                <element ref="saml:AuthnContextDeclRef"/>
            </choice>
        </sequence>
        <choice>
            <element ref="saml:AuthnContextDecl"/>
            <element ref="saml:AuthnContextDeclRef"/>
        </choice>
    </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"/>
```

2.7.3 Element <AttributeStatement>

The `<AttributeStatement>` element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing `<AttributeStatement>` elements MUST contain a `<Subject>` element.

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 assertion subject. An encrypted SAML attribute may be included with the `<EncryptedAttribute>` 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>
```

2.7.3.1 Element <Attribute>

The `<Attribute>` element identifies an attribute by name and optionally includes its value(s). It has the **AttributeType** complex type. It is used within an attribute statement to express particular attributes and values associated with an assertion subject, as described in the previous section. It is also used in an attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for more information). The `<Attribute>` 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 8.2 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:attrname-format:unspecified` (see Section 8.2.1) 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 or opaque, such as an OID or a UUID. This attribute's value MUST NOT be used as a basis for formally identifying SAML attributes.

Arbitrary attributes

This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to be added to `<Attribute>` 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, for example, 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.

`<AttributeValue>` [Any Number]

Contains a 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 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.

The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of values indicates that the requester is interested in any or all of the named attribute's values (see also Section 3.3.2.3).

Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the semantics of specifying or omitting `<AttributeValue>` elements.

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

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
   <sequence>
      <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <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>
```

2.7.3.1.1 Element <AttributeValue>

The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the **xs: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 **xs:integer** or **xs: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 other than an XML Schema simple type on `<AttributeValue>` using `xsi:type` will require the presence of the extension schema that defines the datatype in order for schema processing to proceed.

If a SAML attribute includes an empty value, such as the empty string, the corresponding `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`). This overrides the requirement in Section 1.3.1 that string values in SAML content contain at least one non-whitespace character.

If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "`true`" or "`1`".

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

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

2.7.3.2 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 of or 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 system entity with a SAML name identifier, 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:

```
<element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

2.7.4 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 assertion subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence. Assertions containing `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

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] Section 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] Section 4.2.

`Decision` [Required]

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

`<Action>` [One or more]

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

`<Evidence>` [Optional]

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

The following schema fragment defines the `<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>
```

2.7.4.1 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 but where 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>
```

2.7.4.2 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>
```

2.7.4.3 Element <Evidence>

The `<Evidence>` element contains one or more assertions or assertion references 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 `ID` 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 protocol messages can be generated and exchanged using a variety of protocols. The SAML bindings specification [SAMLBind] describes specific means of transporting protocol messages using existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a number of applications of the protocols defined in this section together with additional processing rules, restrictions, and requirements that facilitate interoperability.

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.

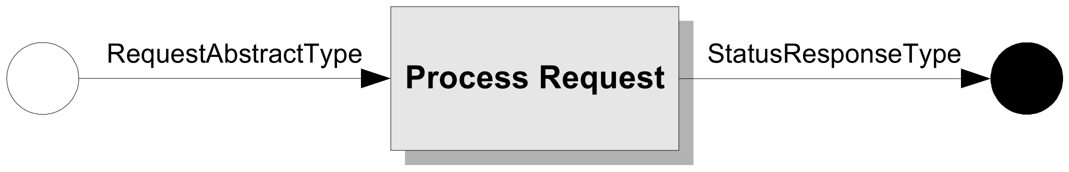

Figure 1: SAML Request-Response Protocol

In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the responder having received a corresponding request.

The protocols defined by SAML achieve the following actions:

- Returning one or more requested assertions. This can occur in response to either a direct request for specific assertions or a query 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 descriptions of elements and types in the SAML protocol namespace are not shown with the conventional namespace prefix `samlp:`. For clarity, text descriptions 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="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml-schema-protocol-2.0
            Location: http://docs.oasis-open.org/security/saml/v2.0/
            Revision history:
            V1.0 (November, 2002):
              Initial Standard Schema.
            V1.1 (September, 2003):
              Updates within the same V1.0 namespace.
            V2.0 (March, 2005):
              New protocol schema based in a SAML V2.0 namespace.
     </documentation>
    </annotation>
…
</schema>
```

3.2 Requests and Responses

The following sections define the SAML constructs and basic requirements 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 **`RequestAbstractType`** for V2.0 of SAML.

`ID` [Required]

An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness. The values of the `ID` attribute in a request and the `InResponseTo` attribute in the corresponding response MUST match.

`Version` [Required]

The version of this request. The identifier for the version of SAML defined in this specification is "2.0". 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 1.3.3.

`Destination` [Optional]

A URI reference indicating the address to which this request has been sent. This is useful to prevent malicious forwarding of requests to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the location at which the message was received. If it does not, the request MUST be discarded. Some protocol bindings may require the use of this attribute (see [SAMLBind]).

`Consent` [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of 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. (For more information on this element, see Section 2.2.5.)

`<ds:Signature>` [Optional]

An XML Signature that authenticates the requester and provides message integrity, as described below and 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 extension elements MUST be namespace-qualified in a non-SAML-defined namespace.

Depending on the requirements of particular protocols or profiles, a SAML requester may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request MAY be signed, which provides both authentication of the requester and message integrity.

If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML responder MUST verify that the signature is valid (that is, that the message has not been tampered with) in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the request or respond with an error (if the request is invalid for some other reason).

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

If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a suspected denial-of-service attack, not responding at all may be warranted.

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="Version" type="string" use="required"/>
   <attribute name="IssueInstant" type="dateTime" use="required"/>
   <attribute name="Destination" type="anyURI" use="optional"/>
   <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>
```

3.2.2 Complex Type StatusResponseType

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

`ID` [Required]

An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 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 (for example, 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.

`Version` [Required]

The version of this response. The identifier for the version of SAML defined in this specification is "2.0". 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 1.3.3.

`Destination` [Optional]

A URI reference indicating the address to which this response has been sent. This is useful to prevent malicious forwarding of responses to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the location at which the message was received. If it does not, the response MUST be discarded. Some protocol bindings may require the use of this attribute (see [SAMLBind]).

`Consent` [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this response. 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 response message. (For more information on this element, see Section 2.2.5.)

`<ds:Signature>` [Optional]

An XML Signature that authenticates the responder and provides message integrity, as described below and 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 extension elements MUST be namespace-qualified in a non-SAML-defined namespace.

`<Status>` [Required]

A code representing the status of the corresponding request.

Depending on the requirements of particular protocols or profiles, a SAML responder may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML response MAY be signed, which provides both authentication of the responder and message integrity.

If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML requester receiving the response MUST verify that the signature is valid (that is, that the message has not been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the response as it deems appropriate.

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

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="Version" type="string" use="required"/>
   <attribute name="IssueInstant" type="dateTime" use="required"/>
   <attribute name="Destination" type="anyURI" use="optional"/>
   <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

3.2.2.1 Element <Status>

The `<Status>` element contains the following elements:

`<StatusCode>` [Required]

A code representing the status of the activity carried out in response to the corresponding request.

`<StatusMessage>` [Optional]

A message which MAY be returned to an operator.

`<StatusDetail>` [Optional]

Additional information concerning the status of the request.

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

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

3.2.2.2 Element <StatusCode>

The `<StatusCode>` element specifies 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. Note that responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for additional information by intentionally presenting erroneous requests.

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. 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:InvalidAttrNameOrValue`

> Unexpected or invalid content was encountered within a `<saml:Attribute>` or `<saml:AttributeValue>` element.

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

> The responding provider cannot or will not support the requested name identifier policy.

`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 responding 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:PartialLogout`

Used by a session authority to indicate to a session participant that it was not able to propagate logout to all other session participants.

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

Indicates that a responding 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:UnknownAttrProfile`

An entity that has no knowledge of a particular attribute profile has been presented with an attribute drawn from that profile.

`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>
```

3.2.2.3 Element <StatusMessage>

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

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

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

3.2.2.4 Element <StatusDetail>

The `<StatusDetail>` element MAY be used to specify additional information concerning the status of the request. 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 (defined in Section 2.4) 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 3.3.4 for 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 a `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 3.3.2.2.1). 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>
```

3.3.2.2.1 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. These elements are defined in Section 2.7.2.2. 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>
```

3.3.2.3 Element <AttributeQuery>

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

`<saml:Attribute>` [Any Number]

Each `<saml:Attribute>` element specifies an attribute whose value(s) are to be returned. If no attributes are specified, it indicates that all attributes allowed by policy are requested. If a given `<saml:Attribute>` element contains one or more `<saml:AttributeValue>` elements, then if that attribute is returned in the response, it MUST NOT contain any values that are not equal to the values specified in the query. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on `<saml:Attribute>`, see Section 2.7.3.1.

A single query MUST NOT contain two `<saml:Attribute>` elements with the same `Name` and `NameFormat` values (that is, a given attribute MUST be named only once in a query).

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 `<Attribute>` elements are present in the query, they constrain/filter the attributes and optionally the values returned, as noted above.
- The attributes and values returned MAY also be constrained by application-specific policy considerations.

The second-level status codes `urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile` and `urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue` MAY be used to indicate problems with the interpretation of attribute or value information in a query.

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

3.3.2.4 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. For more information on this element, see Section 2.7.4.2.

`<saml:Evidence>` [Optional]

A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more information on this element, see Section 2.7.4.3.

In response to an authorization decision query, a SAML authority returns assertions with authorization decision 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.

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 satisfy 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 on these elements.

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 SAML-defined 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 (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), 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 S1 can be confirmed in the manner described by at least one `<saml:SubjectConfirmation>` element in S2.

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 is 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:

Requester

> 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 identity provider and either authenticates itself during the transmission of the message, or relies on an existing security context to establish its identity. If not the requester, the presenter acts as an intermediary between the requester and the responding identity provider.

Requested Subject

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

Attesting Entity

> 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.

Identity Provider

> The entity to whom the presenter gives the request and from whom the presenter receives the response.

3.4.1 Element <AuthnRequest>

To request that an identity provider issue an assertion with an authentication statement, a presenter authenticates to that identity provider (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 requester. The process of authentication of the presenter may take place before, during, or after the initial delivery of the `<AuthnRequest>` message.

The requester might not be the same as the presenter of the request if, for example, the requester is a relying party that intends to use the resulting assertion to authenticate or authorize the requested subject so that the relying party can decide whether 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. For more information on this element, see Section 2.4.

If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the presenter is presumed to be the only attesting 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 requester expects to limit 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. (For more information on this element, see Section 2.5.)

`<RequestedAuthnContext>` [Optional]

Specifies the requirements, if any, that the requester places on the authentication context that applies to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing rules regarding this element.

`<Scoping>` [Optional]

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

`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.

`IsPassive` [Optional]

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

`AssertionConsumerServiceIndex` [Optional]

Indirectly identifies the location to which the `<Response>` message should be returned to the requester. It applies only to profiles in which the requester is different from the presenter, such as the Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message to the default location associated with the requester for the profile of use. If the index specified is invalid, then the identity provider MAY return an error `<Response>` or it MAY use the default location. This attribute is mutually exclusive with the `AssertionConsumerServiceURL` and `ProtocolBinding` attributes.

`AssertionConsumerServiceURL` [Optional]

Specifies by value the location to which the `<Response>` message MUST be returned to the requester. The responder MUST ensure by some means that the value specified is in fact associated with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing `<AuthnRequest>` message is another. This attribute is mutually exclusive with the `AssertionConsumerServiceIndex` attribute and is typically accompanied by the `ProtocolBinding` attribute.

`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. This attribute is mutually exclusive with the `AssertionConsumerServiceIndex` attribute and is typically accompanied by the `AssertionConsumerServiceURL` attribute.

`AttributeConsumingServiceIndex` [Optional]

Indirectly identifies information associated with the requester describing the SAML attributes the requester desires or requires to 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 requester. [SAMLMeta] 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 requester for use by the presenter's user agent or the identity provider.

See Section 3.4.1.4 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="ForceAuthn" type="boolean" use="optional"/>
         <attribute name="IsPassive" 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` [Optional]

Specifies the URI reference corresponding to a name identifier format defined in this or another specification (see Section 8.3 for examples). The additional value of `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` is defined specifically for use within this attribute to indicate a request that the resulting identifier be encrypted.

`SPNameQualifier` [Optional]

Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of a service provider other than the requester, or in the namespace of an affiliation group of service providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` in Section 8.3.7.

`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 "false". When "false", the requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for the principal has already been established. Note that this does not prevent the identity provider from creating such identifiers outside the context of this specific request (for example, in advance for a large number of principals).

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`.

If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any additional constraints due to the content of this element or the policies of the identity provider or principal.

The special `Format` value `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.

Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly specific to the service provider) require that an encrypted identifier be used.

Note that if the requester wishes to permit the identity provider to establish a new identifier for the principal if none exists, it MUST include this element with the `AllowCreate` attribute set to "`true`". Otherwise, only a principal for whom the identity provider has previously established an identifier usable by the requester can be authenticated successfully. This is primarily useful in conjunction with the `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value (see Section 8.3.7).

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

```
<element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
<complexType name="NameIDPolicyType">
   <attribute name="Format" type="anyURI" use="optional"/>
   <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 requester 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 requester deems acceptable to respond to the request.

`<RequesterID>` [Zero or More]

Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate the chain of requesters when proxying occurs, as described in Section 3.4.1.5. 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 requester 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 can be used to retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML instance whose root element is an `<IDPList>` that does not itself contain a `<GetComplete>` element.

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"/>
```

3.4.1.3.1 Element <IDPEntry>

The `<IDPEntry>` element specifies a single identity provider trusted by the requester 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">
   <attribute name="ProviderID" type="anyURI" use="required"/>
   <attribute name="Name" type="string" use="optional"/>
   <attribute name="Loc" type="anyURI" use="optional"/>
</complexType>
```

3.4.1.4 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, or if prevented from providing an assertion by policies in effect at the identity provider (for example the intended subject has prohibited the identity provider from providing assertions to the relying party), then 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 implied:

- 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 in at least one assertion 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 attesting 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 requester as an acceptable relying party. Other audiences MAY be included as deemed appropriate by the identity provider.

3.4.1.5 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 requester 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.

3.4.1.5.1 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 successful 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 requester '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 requester are met.

If, in the future, the identity provider is asked to authenticate the same presenter for a second requester, 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, or to enable a message to be communicated via a secure channel between the SAML requester and responder, avoiding the need for a signature.

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

Note that this protocol is typically not used with "transient" name identifiers, since their value is not intended to be managed on a long term basis.

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 **ManageNameIDRequestType**, 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. (For more information on these elements, see Section 2.2.)

`<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:EncryptedElementType"/>
<element name="Terminate" type="samlp:TerminateType"/>
<complexType name="TerminateType"/>
```

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 for the principal 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 for the principal 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.

Note that neither, either, or both of the original and new identifier MAY be encrypted (using the `<EncryptedID>` and `<NewEncryptedID>` elements).

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`, the `NameQualifier` attribute MUST contain the unique identifier of the identity provider that created the identifier. 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. These attributes MAY be omitted if they would otherwise match the value of the containing protocol message's `<Issuer>` element, but this is NOT RECOMMENDED due to the opportunity for confusion.

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 SHOULD 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. It SHOULD attempt to contact as many of these participants as it can using this protocol, terminate its own session with the principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

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, after which the recipient may discard the message. The time value is encoded in UTC, as described in Section 1.3.3.

`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. (For more information on this element, see Section 2.2.)

`<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="string" use="optional"/>
                <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
            </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 subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in Section 3.3.4.
- The `SessionIndex` attribute of one of the assertion's authentication statements matches one of the `<SessionIndex>` elements specified in the logout request, or the logout request contains no `<SessionIndex>` elements.
- The assertion would otherwise be valid, based on the time conditions specified in the assertion itself (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject confirmation data).
- The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on the message).

> **Note:** This rule is intended to prevent a situation in which a session participant receives a logout request targeted at a single, or multiple, assertion(s) (as identified by the `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid - assertion(s) targeted by the logout request. It should honor the logout request until the logout request itself may be discarded (the `NotOnOrAfter` value on the request has been exceeded) or the assertion targeted by the logout request has been received and has been handled appropriately.

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 assertion 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 principal'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 the session authority successfully terminates the principal's session with respect to itself, then it MUST respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it MUST respond with a `<LogoutResponse>` message containing a top-level status code indicating the error. Thus, the top-level status indicates the state of the logout operation only with respect to the session authority itself.

The session authority SHOULD attempt to contact each session participant using any applicable/usable protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because the original request takes place using a protocol binding that does not enable the logout to be propagated to all participants).

In the event that not all session participants successfully respond to these `<LogoutRequest>` messages (or if not all participants can be contacted), then the session authority MUST include in its `<LogoutResponse>` message a second-level status code of `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session participants successfully responded with confirmation of the logout.

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, after which the logout request may be discarded by the recipient. This value SHOULD be set to a time value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout request).

In addition to the values specified in Section 3.6.3 for 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 session 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 unnecessary.

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. (For more information on this element, see Section 2.2.)

`<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 **NameIDMappingResponseType**, which extends **StatusResponseType** 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. (For more information on this element, see Section 2.2.)

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 is expressed in the form *Major.Minor*. The version number $\mathit{Major}_B.\mathit{Minor}_B$ is higher than the version number $\mathit{Major}_A.\mathit{Minor}_A$ if and only if:

$(\text{Major}_B > \text{Major}_{A)}$ OR (($\text{Major}_B = \text{Major}_A$) AND ($\text{Minor}_B > \text{Minor}_A$))

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 `<xs: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 an attribute for expressing the major and minor version of the assertion in a string of the form *Major.Minor*. 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 asserting party 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 an attribute for expressing the major and minor version of the request or response message using a string of the form *Major.Minor*. 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.)

4.1.3.2 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`.

4.1.3.3 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 asserting party supports assertion integrity, authentication of the asserting party 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 asserting party or SAML responder directly (with no intermediaries) through a secure channel, with the asserting party 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 (see [RFC 2246]/[SSL3]), 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 asserting party SHOULD be signed by the SAML asserting party.
- 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 **xs:ID**-typed attributes 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.

Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to SAML content are free to define other approaches.

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

SAML assertions and protocol messages MUST supply a value for the `ID` attribute on the root element of the assertion or protocol message being signed. 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 (e.g., it might be contained within a SOAP envelope).

Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the `ID` attribute value of the root element of the assertion or protocol message being signed. For example, if the `ID` 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 Example

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

```
<Response
  IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
  ID="_c7055387-af61-4fce-8b98-e2927324b306"
  xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
   <saml:Issuer>https://www.opensaml.org/IDP"</saml:Issuer>
   <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 ds xs 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" Version="2.0"
      xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
      <Issuer>https://www.opensaml.org/IDP</Issuer>
      <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 ds xs 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>
        <Subject>
            <NameID
                Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
                scott@example.org
            </NameID>
            <SubjectConfirmation
                Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
        </Subject>
        <Conditions NotBefore="2003-04-17T00:46:02Z"
                NotOnOrAfter="2003-04-17T00:51:02Z">
            <AudienceRestriction>
                <Audience>http://www.opensaml.org/SP</Audience>
            </AudienceRestriction>
        </Conditions>
        <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
            <AuthnContext>
                <AuthnContextClassRef>
                    urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                </AuthnContextClassRef>
            </AuthnContext>
        </AuthnStatement>
    </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 ensure 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 (see [RFC 2246]/[SSL3]) 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>` element may be encrypted, as described in Section 2.3.4.
- The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

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 plaintext 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 Encryption

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 a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and placed within the `<Assertion>` element before the element is 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.

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, 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. As a practical matter, this means that extensions are typically defined only as types rather than elements, and are included in SAML instances by means of an `xsi:type` attribute.

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 (see [SAML-XSD]) 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**
- **StatusResponseType**

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 **xs:anyType**, which allows any sub-elements and attributes.
- `<AuthnContextDecl>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<AttributeValue>`: Uses **xs: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:

- `<Attribute>` and **AttributeType**

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. In no case should the meaning of a given URI used as such an identifier significantly change, or be used to mean two different things.

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 8.1.1 are interpreted in the same manner described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action `~Read` is affirmatively denied read permission.

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

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 **AttributeType** 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 **xs:Name** as defined in [Schema2] Section 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 SAML V1.1 have been removed for SAML V2.0.

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] Section 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 IETF RFC 1510 [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, requester, or responder) 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.

The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

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 a protocol message or an assertion containing the identifier in its subject. Note that a different system entity might later issue its own protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be omitted in such a case).

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 unique identifier 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, then the attribute MUST be omitted.

Persistent identifiers are intended as a privacy protection mechanism; 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.

Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of creation, but not of use. If a persistent identifier is created by a particular identity provider, the `NameQualifier` attribute value is permanently established at that time. If a service provider that receives such an identifier takes on the role of an identity provider and issues its own assertion containing that identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It might alternatively choose to create its own persistent identifier to represent the principal and link the two values. This is a deployment decision.

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 1.3.4), 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** and **StatusResponseType** complex types to communicate whether a principal gave consent, and under what conditions, for the message.

8.4.1 Unspecified

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

No claim as to principal consent is being made.

8.4.2 Obtained

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

Indicates that a principal’s consent has been obtained by the issuer of the message.

8.4.3 Prior

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

Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to the action that initiated the message.

8.4.4 Implicit

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

Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the action that initiated the message, 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 principal's consent has been explicitly obtained by the issuer of the message during the action that initiated the message.

8.4.6 Unavailable

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

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

8.4.7 Inapplicable

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

Indicates that the issuer of the message 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. See 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. See 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. See 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. See http://www.w3.org/TR/REC-xml.

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

[XMLEnc-XSD] XML Encryption Schema. World Wide Web Consortium. See 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, January 1999. See http://www.w3.org/TR/REC-xml-names.

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing.* World Wide Web Consortium, February 2002. See 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. See 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. See http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.

[RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5).* IETF RFC 1510, September 1993. See 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. See http://www.ietf.org/rfc/rfc2119.txt.

[RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See 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. See 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. See http://www.ietf.org/rfc/rfc2396.txt.

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

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

[RFC 3513] R. Hinden, S.Deering, *Internet Protocol Version 6 (IPv6) Addressing Architecture.* IETF RFC 3513, April 2003. See http://www.ietf.org/rfc/rfc3513.txt.

[SAMLAuthnCxt] J. Kemp et al. *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. http://www.oasis-open.org/committees/security/.

[SAMLGloss] J. Hodges et al. *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLMeta] S. Cantor et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLP-XSD] S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See http://www.oasis-open.org/committees/security/.

[SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLSecure] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLTechOvw] J. Hughes et al. SAML Technical Overview. OASIS, February 2005. Document ID sstc-saml-tech-overview-2.0-draft-03. See http://www.oasis-open.org/committees/security/.

[SAML-XSD] S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See http://www.oasis-open.org/committees/security/.

[SSL3] A. Frier et al. *The SSL 3.0 Protocol.* Netscape Communications Corp, November 1996.

[UNICODE-C] M. Davis, M. J. Dürst. *Unicode Normalization Forms.* UNICODE Consortium, March 2001. See 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. See http://www.w3.org/TR/WD-charreq.

[W3C-CharMod] M. J. Dürst. *Character Model for the World Wide Web 1.0: Normalization.* World Wide Web Consortium, February 2004. See http://www.w3.org/TR/charmod-norm/.

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

[XML-ID] J. Marsh et al. *xml:id Version 1.0*, World Wide Web Consortium, April 2004. See http://www.w3.org/TR/xml-id/.

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:

- Conor Cahill, AOL
- John Hughes, Atos Origin
- Hal Lockhart, BEA Systems
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiester, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems

- Eve Maler, Sun Microsystems
- Ronald Monzillo, Sun Microsystems
- Emily Xu, Sun Microsystems
- Greg Whitehead, Trustgenix

The editors also would like to acknowledge the following former SSTC members for their contributions to this or previous versions of the OASIS Security Assertions Markup Language Standard:

- Stephen Farrell, Baltimore Technologies
- David Orchard, BEA Systems
- Krishna Sankar, Cisco Systems
- Zahid Ahmed, CommerceOne
- Tim Alsop, CyberSafe Limited
- Carlisle Adams, Entrust
- Tim Moses, Entrust
- Nigel Edwards, Hewlett-Packard
- Joe Pato, Hewlett-Packard
- Bob Blakley, IBM
- Marlena Erdos, IBM
- Marc Chanliau, Netegrity
- Chris McLaren, Netegrity
- Lynne Rosenthal, NIST
- Mark Skall, NIST
- Charles Knouse, Oblix
- Simon Godik, Overxeer
- Charles Norwood, SAIC
- Evan Prodromou, Securant
- Robert Griffin, RSA Security (former editor)
- Sai Allarvarpu, Sun Microsystems
- Gary Ellison, Sun Microsystems
- Chris Ferris, Sun Microsystems
- Mike Myers, Traceroute Security
- Phillip Hallam-Baker, VeriSign (former editor)
- James Vanderbeek, Vodafone
- Mark O’Neill, Vordel
- Tony Palmer, Vordel

Finally, the editors wish to acknowledge the following people for their contributions of material used as input to the OASIS Security Assertions Markup Language specifications:

- Thomas Gross, IBM
- Birgit Pfitzmann, IBM

Appendix B. Notices