

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

Working Draft ~~076~~, ~~1328~~ February-
~~December~~September ~~2015~~2019~~07~~

Document identifier:

sstc-saml-core-errata-2.0-wd-~~076~~

Location:

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

Editors:

Scott Cantor, Internet2
John Kemp, Nokia
Rob Philpott, RSA Security
Eve Maler, Individual (errata editor)
Eric Goodman, Individual (errata editor)

Contributors to the Errata:

Rob Philpott, EMC Corporation
Nick Ragouzis, Enosis Group
Thomas Wisniewski, Entrust
Greg Whitehead, HP
Heather Hinton, IBM
Connor P. Cahill, Intel
Scott Cantor, Internet2
Nate Klingenstein, Internet2
RL 'Bob' Morgan, Internet2
John Bradley, Individual
Jeff Hodges, Individual
Joni Brennan, Liberty Alliance
Eric Tiffany, Liberty Alliance
Thomas Hardjono, M.I.T.
Tom Scavo, NCSA
Peter Davis, NeuStar, Inc.
Frederick Hirsch, Nokia Corporation
Paul Madsen, NTT Corporation
Ari Kermaier, Oracle Corporation
Hal Lockhart, Oracle Corporation
Prateek Mishra, Oracle Corporation
Brian Campbell, Ping Identity
Anil Saldhana, Red Hat Inc.
Jim Lien, RSA Security
Jahan Moreh, Sigaba
Kent Spaulding, Skyworth TTG Holdings Limited
Emily Xu, Sun Microsystems
David Staggs, Veteran's Health Administration

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:

The SAML V2.0 Assertions and Protocols specification defines the syntax and semantics for XML-encoded assertions about authentication, attributes, and authorization, and for the protocols that convey this information. This document, known as an “errata composite”, combines corrections to reported errata with the original specification text. By design, the corrections are limited to clarifications of ambiguous or conflicting specification text. This document shows deletions from the original specification as struck-through text, and additions as colored underlined text. The “[*Errn*]” designations embedded in the text refer to particular errata and their dispositions.

Status:

This errata composite document is a **working draft** based on the [original](#) OASIS Standard document that had been produced by the Security Services Technical Committee and approved by the OASIS membership on 1 March 2005. While the errata corrections appearing here are non-normative, they reflect changes specified by the Approved Errata document (currently at Working Draft revision 02), which is on an OASIS standardization track. In case of any discrepancy between this document and the Approved Errata, the latter has precedence. [See also the Errata Working Document \(currently at revision 39\), which provides background on the changes specified here.](#)

This document includes corrections for errata E0, E6, E8, E10, E12, E13, E14, E15, E30, E36, E38, E43, E45, E46, E47, E49, E55, E60, E61, E65, E74, E75, E78, ~~and E79~~, [E81](#), [E83](#), [E84](#), [E86](#), [E91](#), [E92](#) and [E93](#).

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them by following the instructions at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

Table of Contents

105	1 Introduction.....	8
106	1.1 Notation.....	8
107	1.2 Schema Organization and Namespaces.....	9
108	1.3 Common Data Types.....	9
109	1.3.1 String Values.....	9
110	1.3.2 URI Values.....	10
111	1.3.3 Time Values.....	10
112	1.3.4 ID and ID Reference Values.....	10
113	2 SAML Assertions.....	12
114	2.1 Schema Header and Namespace Declarations.....	12
115	2.2 Name Identifiers.....	13
116	2.2.1 Element <BaseID>.....	13
117	2.2.2 Complex Type NameIDType.....	14
118	2.2.3 Element <NameID>.....	15
119	2.2.4 Element <EncryptedID>.....	15
120	2.2.5 Element <Issuer>.....	16
121	2.3 Assertions.....	16
122	2.3.1 Element <AssertionIDRef>.....	16
123	2.3.2 Element <AssertionURIRef>.....	16
124	2.3.3 Element <Assertion>.....	16
125	2.3.4 Element <EncryptedAssertion>.....	18
126	2.4 Subjects.....	18
127	2.4.1 Element <Subject>.....	19
128	2.4.1.1 Element <SubjectConfirmation>.....	19
129	2.4.1.2 Element <SubjectConfirmationData>.....	20
130	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	21
131	2.4.1.4 Example of a Key-Confirmed <Subject>.....	22
132	2.5 Conditions.....	22
133	2.5.1 Element <Conditions>.....	22
134	2.5.1.1 General Processing Rules.....	23
135	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	24
136	2.5.1.3 Element <Condition>.....	24
137	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	24
138	2.5.1.5 Element <OneTimeUse>.....	25
139	2.5.1.6 Element <ProxyRestriction>.....	26
140	2.6 Advice.....	27
141	2.6.1 Element <Advice>.....	27
142	2.7 Statements.....	27
143	2.7.1 Element <Statement>.....	27
144	2.7.2 Element <AuthnStatement>.....	28
145	2.7.2.1 Element <SubjectLocality>.....	29
146	2.7.2.2 Element <AuthnContext>.....	29
147	2.7.3 Element <AttributeStatement>.....	30
148	2.7.3.1 Element <Attribute>.....	31
149	2.7.3.1.1 Element <AttributeValue>.....	32
150	2.7.3.2 Element <EncryptedAttribute>.....	32
151	2.7.4 Element <AuthzDecisionStatement>.....	33
152	2.7.4.1 Simple Type DecisionType.....	34
153	2.7.4.2 Element <Action>.....	35

154	2.7.4.3 Element <Evidence>.....	35
155	3 SAML Protocols.....	37
156	3.1 Schema Header and Namespace Declarations.....	37
157	3.2 Requests and Responses.....	38
158	3.2.1 Complex Type RequestAbstractType.....	38
159	3.2.2 Complex Type StatusResponseType.....	40
160	3.2.2.1 Element <Status>.....	41
161	3.2.2.2 Element <StatusCode>.....	41
162	3.2.2.3 Element <StatusMessage>.....	44
163	3.2.2.4 Element <StatusDetail>.....	44
164	3.3 Assertion Query and Request Protocol.....	44
165	3.3.1 Element <AssertionIDRequest>.....	44
166	3.3.2 Queries.....	44
167	3.3.2.1 Element <SubjectQuery>.....	44
168	3.3.2.2 Element <AuthnQuery>.....	45
169	3.3.2.2.1 Element <RequestedAuthnContext>.....	46
170	3.3.2.3 Element <AttributeQuery>.....	47
171	3.3.2.4 Element <AuthzDecisionQuery>.....	48
172	3.3.3 Element <Response>.....	48
173	3.3.4 Processing Rules.....	49
174	3.4 Authentication Request Protocol.....	49
175	3.4.1 Element <AuthnRequest>.....	50
176	3.4.1.1 Element <NameIDPolicy>.....	52
177	3.4.1.2 Element <Scoping>.....	54
178	3.4.1.3 Element <IDPList>.....	55
179	3.4.1.3.1 Element <IDPEntry>.....	55
180	3.4.1.4 Processing Rules.....	56
181	3.4.1.5 Proxying.....	57
182	3.4.1.5.1 Proxying Processing Rules.....	57
183	3.5 Artifact Resolution Protocol.....	58
184	3.5.1 Element <ArtifactResolve>.....	58
185	3.5.2 Element <ArtifactResponse>.....	59
186	3.5.3 Processing Rules.....	59
187	3.6 Name Identifier Management Protocol.....	60
188	3.6.1 Element <ManageNameIDRequest>.....	60
189	3.6.2 Element <ManageNameIDResponse>.....	61
190	3.6.3 Processing Rules.....	61
191	3.7 Single Logout Protocol.....	63
192	3.7.1 Element <LogoutRequest>.....	63
193	3.7.2 Element <LogoutResponse>.....	64
194	3.7.3 Processing Rules.....	64
195	3.7.3.1 Session Participant Rules.....	65
196	3.7.3.2 Session Authority Rules.....	65
197	3.8 Name Identifier Mapping Protocol.....	66
198	3.8.1 Element <NameIDMappingRequest>.....	67
199	3.8.2 Element <NameIDMappingResponse>.....	67
200	3.8.3 Processing Rules.....	68
201	4 SAML Versioning.....	69
202	4.1 SAML Specification Set Version.....	69
203	4.1.1 Schema Version.....	69
204	4.1.2 SAML Assertion Version.....	69
205	4.1.3 SAML Protocol Version.....	70
206	4.1.3.1 Request Version.....	70

207	4.1.3.2 Response Version.....	70
208	4.1.3.3 Permissible Version Combinations.....	71
209	4.2 SAML Namespace Version.....	71
210	4.2.1 Schema Evolution.....	71
211	5 SAML and XML Signature Syntax and Processing.....	72
212	5.1 Signing Assertions.....	72
213	5.2 Request/Response Signing.....	72
214	5.3 Signature Inheritance.....	72
215	5.4 XML Signature Profile.....	73
216	5.4.1 Signing Formats and Algorithms.....	73
217	5.4.2 References.....	73
218	5.4.3 Canonicalization Method.....	73
219	5.4.4 Transforms.....	74
220	5.4.5 [E91] Object.....	74
221	5.4.6 KeyInfo.....	74
222	5.4.7 Example.....	74
223	6 SAML and XML Encryption Syntax and Processing.....	77
224	6.1 General Considerations.....	77
225	6.2 [E93] Encryption and Integrity Protection.....	77
226	6.3 [E43] Key and Data Referencing Guidelines.....	78
227	6.4 Examples.....	78
228	7 SAML Extensibility.....	81
229	7.1 Schema Extension.....	81
230	7.1.1 Assertion Schema Extension.....	81
231	7.1.2 Protocol Schema Extension.....	81
232	7.2 Schema Wildcard Extension Points.....	82
233	7.2.1 Assertion Extension Points.....	82
234	7.2.2 Protocol Extension Points.....	82
235	7.3 Identifier Extension.....	82
236	8 SAML-Defined Identifiers.....	83
237	8.1 Action Namespace Identifiers.....	83
238	8.1.1 Read/Write/Execute/Delete/Control.....	83
239	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	83
240	8.1.3 Get/Head/Put/Post.....	84
241	8.1.4 UNIX File Permissions.....	84
242	8.2 Attribute Name Format Identifiers.....	84
243	8.2.1 Unspecified.....	84
244	8.2.2 URI Reference.....	85
245	8.2.3 Basic.....	85
246	8.3 Name Identifier Format Identifiers.....	85
247	8.3.1 Unspecified.....	85
248	8.3.2 Email Address.....	85
249	8.3.3 X.509 Subject Name.....	85
250	8.3.4 Windows Domain Qualified Name.....	85
251	8.3.5 Kerberos Principal Name.....	86
252	8.3.6 Entity Identifier.....	86
253	8.3.7 Persistent Identifier.....	86
254	8.3.8 Transient Identifier.....	87
255	8.4 Consent Identifiers.....	87
256	8.4.1 Unspecified.....	87

257	8.4.2 Obtained.....	87
258	8.4.3 Prior.....	88
259	8.4.4 Implicit.....	88
260	8.4.5 Explicit.....	88
261	8.4.6 Unavailable.....	88
262	8.4.7 Inapplicable.....	88
263	9 References.....	89
264	9.1 Normative References.....	89
265	9.2 Non-Normative References.....	89
266	Appendix A. Acknowledgments.....	91
267	Appendix B. Notices.....	93
268		

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 [SAMPLP-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 [SAMPLP-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax

Prefix	XML Namespace	Comments
		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.

302 This specification uses the following typographical conventions in text: <SAML**E**lement>,
303 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

304 1.2 Schema Organization and Namespaces

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

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

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

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

311 The assertion schema is imported into the protocol schema. See Section 4.2 for information on SAML
312 namespace versioning.

313 Also imported into both schemas is the schema for XML Signature [XMLSig], which is associated with the
314 following XML namespace:

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

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

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

319 1.3 Common Data Types

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

322 1.3.1 String Values

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

327 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that
328 have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact
329 binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-
330 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.

[E92] SAML system entities SHOULD allow for reasonable clock skew between systems when interpreting time instants and enforcing security policies based on them. Tolerances of 3-5 minutes are reasonable defaults, but allowing for configurability is a suggested practice in implementations

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.

373 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the
374 implementation. In the case that a random or pseudorandom technique is employed, the probability of two
375 randomly chosen identifiers being identical MUST be less than or equal to 2^{-128} and SHOULD be less than
376 or equal to 2^{-160} . This requirement MAY be met by encoding a randomly chosen value between 128 and
377 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom
378 generator MUST be seeded with unique material in order to ensure the desired uniqueness properties
379 between different systems.

380 The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs:IDREF**
381 cannot be used for this purpose. In SAML, the element referred to by a SAML identifier reference might
382 actually be defined in a document separate from that in which the identifier reference is used. Using
383 **xs:IDREF** would violate the requirement that its value match the value of an ID attribute on some element
384 in the same XML document.

385 **Note:** It is anticipated that the World Wide Web Consortium will standardize a global
386 attribute for holding ID-typed values, called `xml:id` [XML-ID]. The Security Services
387 Technical Committee plans to move away from SAML-specific ID attributes to this style of
388 assigning unique identifiers as soon as practicable after the `xml:id` attribute is
389 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 [\[E13\]](#) or is indeterminate.

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/xmenc#"
  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/xmenc#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-
20021210/xenc-schema.xsd"/>
  <annotation>
    <documentation>
```

```

437         Document identifier: saml-schema-assertion-2.0
438         Location: http://docs.oasis-open.org/security/saml/v2.0/
439         Revision history:
440         V1.0 (November, 2002):
441             Initial Standard Schema.
442         V1.1 (September, 2003):
443             Updates within the same V1.0 namespace.
444         V2.0 (March, 2005):
445             New assertion schema for SAML V2.0 namespace.
446     </documentation>
447 </annotation>
448 ...
449 </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 [\[E60\]](#)-[urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified](#)~~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:

```

529 <complexType name="NameIDType">
530   <simpleContent>
531     <extension base="string">
532       <attributeGroup ref="saml:IDNameQualifiers"/>
533       <attribute name="Format" type="anyURI" use="optional"/>
534       <attribute name="SPProvidedID" type="string" use="optional"/>
535     </extension>
536   </simpleContent>
537 </complexType>

```

538 2.2.3 Element <NameID>

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

542 The following schema fragment defines the <NameID> element:

```

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

```

544 2.2.4 Element <EncryptedID>

545 The <EncryptedID> element is of type **EncryptedElementType**, and carries the content of an
546 unencrypted identifier element in encrypted fashion, as defined by the XML Encryption Syntax and
547 Processing specification [XMLEnc]. The <EncryptedID> element contains the following elements:

548 <xenc:EncryptedData> [Required]

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

554 <xenc:EncryptedKey> [Zero or More]

555 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
556 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of
557 the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity, as defined by
558 Section 8.3.6.

559 Encrypted identifiers are intended as a privacy protection mechanism when the plain-text value passes
560 through an intermediary. As such, the ciphertext MUST be unique to any given encryption operation. For
561 more on such issues, see [XMLEnc] Section 6.3.

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

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

```

567 <complexType name="EncryptedElementType">
568   <sequence>
569     <element ref="xenc:EncryptedData"/>
570     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
571   </sequence>
572 </complexType>
573 <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.

611 <Issuer> [Required]
 612 The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be
 613 unambiguous to the intended relying parties.

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

617 <ds:Signature> [Optional]
 618 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as
 619 described below and in Section 5.

620 <Subject> [Optional]
 621 The subject of the statement(s) in the assertion.

622 <Conditions> [Optional]
 623 Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion.
 624 See Section 2.5 for additional information on how to evaluate conditions.

625 <Advice> [Optional]
 626 Additional information related to the assertion that assists processing in certain situations but which
 627 MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

628 Zero or more of the following statement elements:

629 <Statement>
 630 A statement of a type defined in an extension schema. An xsi:type attribute MUST be used to
 631 indicate the actual statement type.

632 <AuthnStatement>
 633 An authentication statement.

634 <AuthzDecisionStatement>
 635 An authorization decision statement.

636 <AttributeStatement>
 637 An attribute statement.

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

641 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If
 642 <Subject> is omitted, then the statements in the assertion apply to a subject or subjects identified in an
 643 application- or profile-specific manner. SAML itself defines no such statements, and an assertion without
 644 a subject has no defined meaning in this specification.

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

650 If such a signature is used, then the <ds:Signature> element MUST be present, and a relying party
 651 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in
 652 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the
 653 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.

[E47] If the <SubjectConfirmation> element in an assertion subject contains an identifier the issuer authorizes the attesting entity to wield the assertion on behalf of that subject. A relying party MAY apply additional constraints on the use of such an assertion at its discretion, based upon the identities of both the subject and the attesting entity.

If an assertion is issued for use by an entity other than the subject, then that entity SHOULD be identified in the <SubjectConfirmation> 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.

792 Address [Optional]

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

799 Arbitrary attributes

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

807 Arbitrary elements

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

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

815 Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present,
816 SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's
817 `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore`
818 MUST be less than (earlier than) the value for `NotOnOrAfter`. [\[E92\] As noted in section 1.3.3, relying](#)
819 [parties SHOULD allow for reasonable clock skew in the interpretation of both values.](#)

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

```
822 <element name="SubjectConfirmationData"  
823   type="saml:SubjectConfirmationDataType"/>  
824 <complexType name="SubjectConfirmationDataType" mixed="true">  
825   <complexContent>  
826     <restriction base="anyType">  
827       <sequence>  
828         <any namespace="##any" processContents="lax" minOccurs="0"  
829         maxOccurs="unbounded"/>  
830       </sequence>  
831       <attribute name="NotBefore" type="dateTime" use="optional"/>  
832       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
833       <attribute name="Recipient" type="anyURI" use="optional"/>  
834       <attribute name="InResponseTo" type="NCName" use="optional"/>  
835       <attribute name="Address" type="string" use="optional"/>  
836       <anyAttribute namespace="##other" processContents="lax"/>  
837     </restriction>  
838   </complexContent>  
839 </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. [\[E92\] As noted in section 1.3.3, relying parties SHOULD allow for reasonable clock skew in the interpretation of both values.](#)

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 [E46]<AudienceRestrictions>condition, the <Audience> elementsaudiences form a disjunction (an "OR") while multiple <AudienceRestrictions> elementseconditions 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.

1016 The `<OneTimeUse>` element indicates that the assertion SHOULD be used immediately by the relying
1017 party and MUST NOT be retained for future use. Relying parties are always free to request a fresh
1018 assertion for every use. However, implementations that choose to retain assertions for future use MUST
1019 observe the `<OneTimeUse>` element. This condition is independent from the `NotBefore` and
1020 `NotOnOrAfter` condition information.

1021 To support the single use constraint, a relying party should maintain a cache of the assertions it has
1022 processed containing such a condition. Whenever an assertion with this condition is processed, the cache
1023 should be checked to ensure that the same assertion has not been previously received and processed by
1024 the relying party.

1025 A SAML authority MUST NOT include more than one `<OneTimeUse>` element within a `<Conditions>`
1026 element of an assertion.

1027 For the purposes of determining the validity of the `<Conditions>` element, the `<OneTimeUse>` is
1028 considered to always be valid. That is, this condition does not affect validity but is a condition on use.

1029 The following schema fragment defines the `<OneTimeUse>` element and its **OneTimeUseType** complex
1030 type:

```
1031 <element name="OneTimeUse" type="saml:OneTimeUseType"/>
1032 <complexType name="OneTimeUseType">
1033   <complexContent>
1034     <extension base="saml:ConditionAbstractType"/>
1035   </complexContent>
1036 </complexType>
```

1037 2.5.1.6 Element `<ProxyRestriction>`

1038 Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as
1039 asserting parties and issue subsequent assertions of their own on the basis of the information contained
1040 in the original assertion. A relying party acting as an asserting party MUST NOT issue an assertion that
1041 itself violates the restrictions specified in this condition on the basis of an assertion containing such a
1042 condition.

1043 The `<ProxyRestriction>` element contains the following elements and attributes:

1044 `Count` [Optional]

1045 Specifies the maximum number of indirections that the asserting party permits to exist between this
1046 assertion and an assertion which has ultimately been issued on the basis of it.

1047 `<Audience>` [Zero or More]

1048 Specifies the set of audiences to whom the asserting party permits new assertions to be issued on
1049 the basis of this assertion.

1050 A `Count` value of zero indicates that a relying party MUST NOT issue an assertion to another relying
1051 party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves
1052 contain a `<ProxyRestriction>` element with a `Count` value of at most one less than this value.

1053 If no `<Audience>` elements are specified, then no audience restrictions are imposed on the relying
1054 parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST
1055 themselves contain an `<AudienceRestriction>` element with at least one of the `<Audience>`
1056 elements present in the previous `<ProxyRestriction>` element, and no `<Audience>` elements
1057 present that were not in the previous `<ProxyRestriction>` element.

1058 A SAML authority MUST NOT include more than one `<ProxyRestriction>` element within a
1059 `<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.~~[\[E79\]](#)~~Indicates an upper bound on sessions with the subject derived from the enclosing assertion.~~ 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. [It's left to profiles to provide specific processing rules for relying parties based on this attribute.](#)

<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:

1239 <Attribute> or <EncryptedAttribute> [One or More]

1240 The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML
1241 attribute may be included with the <EncryptedAttribute> element.

1242 The following schema fragment defines the <AttributeStatement> element and its
1243 **AttributeStatementType** complex type:

```
1244 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1245 <complexType name="AttributeStatementType">
1246   <complexContent>
1247     <extension base="saml:StatementAbstractType">
1248       <choice maxOccurs="unbounded">
1249         <element ref="saml:Attribute"/>
1250         <element ref="saml:EncryptedAttribute"/>
1251       </choice>
1252     </extension>
1253   </complexContent>
1254 </complexType>
```

1255 2.7.3.1 Element <Attribute>

1256 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the
1257 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and
1258 values associated with an assertion subject, as described in the previous section. It is also used in an
1259 attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for
1260 more information). The <Attribute> element contains the following XML attributes:

1261 Name [Required]

1262 The name of the attribute.

1263 NameFormat [Optional]

1264 A URI reference representing the classification of the attribute name for purposes of interpreting the
1265 name. See Section 8.2 for some URI references that MAY be used as the value of the NameFormat
1266 attribute and their associated descriptions and processing rules. If no NameFormat value is provided,
1267 the identifier urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified (see Section
1268 8.2.1) is in effect.

1269 FriendlyName [Optional]

1270 A string that provides a more human-readable form of the attribute's name, which may be useful in
1271 cases in which the actual Name is complex or opaque, such as an OID or a UUID. This attribute's
1272 value MUST NOT be used as a basis for formally identifying SAML attributes.

1273 Arbitrary attributes

1274 This complex type uses an <xs:anyAttribute> extension point to allow arbitrary XML attributes to
1275 be added to <Attribute> constructs without the need for an explicit schema extension. This allows
1276 additional fields to be added as needed to supply additional parameters to be used, for example, in
1277 an attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes
1278 or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a
1279 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1280 <AttributeValue> [Any Number]

1281 Contains a value of the attribute. If an attribute contains more than one discrete value, it is
1282 RECOMMENDED that each value appear in its own <AttributeValue> element. If more than
1283 one <AttributeValue> element is supplied for an attribute, and any of the elements have a
1284 datatype assigned through xsi:type, then all of the <AttributeValue> elements must have
1285 the identical datatype assigned.

[E49]Attributes are identified/named by the combination of the `NameFormat` and `Name` XML attributes described above. Neither one in isolation can be assumed to be unique, but taken together, they ought to be unambiguous within a given deployment.

The SAML profiles specification [SAMLProf] includes a number of attribute profiles designed to improve the interoperability of attribute usage in some identified scenarios. Such profiles typically include constraints on attribute naming and value syntax. There is no explicit indicator when an attribute profile is in use, and it is assumed that deployments can establish this out of band, based on the combination of `NameFormat` and `Name`.

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.

1417 Indeterminate

1418 The SAML authority cannot determine whether the specified action is permitted or denied.

1419 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to
1420 provide an affirmative statement but where it is not able to issue a decision. Additional information as to
1421 the reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>`
1422 elements in the enclosing `<Response>`.

1423 The following schema fragment defines the **DecisionType** simple type:

```
1424 <simpleType name="DecisionType">  
1425   <restriction base="string">  
1426     <enumeration value="Permit"/>  
1427     <enumeration value="Deny"/>  
1428     <enumeration value="Indeterminate"/>  
1429   </restriction>  
1430 </simpleType>
```

1431 2.7.4.2 Element `<Action>`

1432 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its
1433 string-data content provides the label for an action sought to be performed on the specified resource, and
1434 it has the following attribute:

1435 Namespace [[E36]RequiredOptional]

1436 A URI reference representing the namespace in which the name of the specified action is to be
1437 interpreted. If this element is absent, the namespace
1438 ~~urn:oasis:names:tc:SAML:1.0:action:rwede-negation specified in Section 8.1.2 is in~~
1439 ~~effect.~~

1440 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```
1441 <element name="Action" type="saml:ActionType"/>  
1442 <complexType name="ActionType">  
1443   <simpleContent>  
1444     <extension base="string">  
1445       <attribute name="Namespace" type="anyURI" use="required"/>  
1446     </extension>  
1447   </simpleContent>  
1448 </complexType>
```

1449 2.7.4.3 Element `<Evidence>`

1450 The `<Evidence>` element contains one or more assertions or assertion references that the SAML
1451 authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It
1452 contains a mixture of one or more of the following elements:

1453 `<AssertionIDRef>` [Any number]

1454 Specifies an assertion by reference to the value of the assertion's `ID` attribute.

1455 `<AssertionURIRef>` [Any number]

1456 Specifies an assertion by means of a URI reference.

1457 `<Assertion>` [Any number]

1458 Specifies an assertion by value.

1459 `<EncryptedAssertion>` [Any number]

1460 Specifies an encrypted assertion by value.

1461 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1462 and the SAML authority making the authorization decision. For example, in the case that the SAML
1463 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use that
1464 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's
1465 assertion as valid either to the relying party or any other third party.

1466 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1467 <element name="Evidence" type="saml:EvidenceType"/>
1468 <complexType name="EvidenceType">
1469   <choice maxOccurs="unbounded">
1470     <element ref="saml:AssertionIDRef"/>
1471     <element ref="saml:AssertionURIRef"/>
1472     <element ref="saml:Assertion"/>
1473     <element ref="saml:EncryptedAssertion"/>
1474   </choice>
1475 </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"
```

```

1514     version="2.0">
1515     <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1516           schemaLocation="saml-schema-assertion-2.0.xsd"/>
1517     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1518           schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
1519 20020212/xmldsig-core-schema.xsd"/>
1520     <annotation>
1521       <documentation>
1522         Document identifier: saml-schema-protocol-2.0
1523         Location: http://docs.oasis-open.org/security/saml/v2.0/
1524         Revision history:
1525         V1.0 (November, 2002):
1526           Initial Standard Schema.
1527         V1.1 (September, 2003):
1528           Updates within the same V1.0 namespace.
1529         V2.0 (March, 2005):
1530           New protocol schema based in a SAML V2.0 namespace.
1531       </documentation>
1532     </annotation>
1533     ...
1534   </schema>

```

1535 3.2 Requests and Responses

1536 The following sections define the SAML constructs and basic requirements that underlie all of the request
 1537 and response messages used in SAML protocols.

1538 3.2.1 Complex Type RequestAbstractType

1539 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
 1540 This type defines common attributes and elements that are associated with all SAML requests:

1541 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**
 1542 for V2.0 of SAML.

1543 ID [Required]

1544 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section
 1545 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo
 1546 attribute in the corresponding response MUST match.

1547 Version [Required]

1548 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".
 1549 SAML versioning is discussed in Section 4.

1550 IssueInstant [Required]

1551 The time instant of issue of the request. The time value is encoded in UTC, as described in Section
 1552 1.3.3.

1553 Destination [Optional]

1554 A URI reference indicating the address to which this request has been sent. This is useful to prevent
 1555 malicious forwarding of requests to unintended recipients, a protection that is required by some
 1556 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies
 1557 the location at which the message was received. If it does not, the request MUST be discarded.
 1558 Some protocol bindings may require the use of this attribute (see [SAMLBind]).

1559 Consent [Optional]

1560 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
1561 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value
1562 of the Consent attribute and their associated descriptions. If no Consent value is provided, the
1563 identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
1564 effect.

1565 <saml:Issuer> [Optional]

1566 Identifies the entity that generated the request message. (For more information on this element, see
1567 Section 2.2.5.)

1568 <ds:Signature> [Optional]

1569 An XML Signature that authenticates the requester and provides message integrity, as described
1570 below and in Section 5.

1571 <Extensions> [Optional]

1572 This extension point contains optional protocol message extension elements that are agreed on
1573 between the communicating parties. No extension schema is required in order to make use of this
1574 extension point, and even if one is provided, the lax validation setting does not impose a requirement
1575 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1576 SAML-defined namespace.

1577 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to
1578 authenticate itself, and message integrity may often be required. Authentication and message integrity
1579 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request
1580 MAY be signed, which provides both authentication of the requester and message integrity.

1581 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1582 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)
1583 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the
1584 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the
1585 signature to determine the identity and appropriateness of the signer and may continue to process the
1586 request or respond with an error (if the request is invalid for some other reason).

1587 If a Consent attribute is included and the value indicates that some form of principal consent has been
1588 obtained, then the request SHOULD be signed.

1589 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if
1590 it responds, it MUST return a SAML response message with a <StatusCode> element with the value
1591 urn:oasis:names:tc:SAML:2.0:status:Requester. In some cases, for example during a
1592 suspected denial-of-service attack, not responding at all may be warranted.

1593 The following schema fragment defines the **RequestAbstractType** complex type:

```
1594   <complexType name="RequestAbstractType" abstract="true">
1595     <sequence>
1596       <element ref="saml:Issuer" minOccurs="0"/>
1597       <element ref="ds:Signature" minOccurs="0"/>
1598       <element ref="samlp:Extensions" minOccurs="0"/>
1599     </sequence>
1600     <attribute name="ID" type="ID" use="required"/>
1601     <attribute name="Version" type="string" use="required"/>
1602     <attribute name="IssueInstant" type="dateTime" use="required"/>
1603     <attribute name="Destination" type="anyURI" use="optional"/>
1604     <attribute name="Consent" type="anyURI" use="optional"/>
1605   </complexType>
1606   <element name="Extensions" type="samlp:ExtensionsType"/>
1607   <complexType name="ExtensionsType">
```

```

1608     <sequence>
1609         <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1610     </sequence>
1611 </complexType>

```

1612 3.2.2 Complex Type StatusResponseType

1613 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type
 1614 defines common attributes and elements that are associated with all SAML responses:

1615 ID [Required]

1616 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in
 1617 Section 1.3.4 for identifier uniqueness.

1618 InResponseTo [Optional]

1619 A reference to the identifier of the request to which the response corresponds, if any. If the response
 1620 is not generated in response to a request, or if the ID attribute value of a request cannot be
 1621 determined (for example, the request is malformed), then this attribute MUST NOT be present.
 1622 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's
 1623 ID attribute.

1624 Version [Required]

1625 The version of this response. The identifier for the version of SAML defined in this specification is
 1626 "2.0". SAML versioning is discussed in Section 4.

1627 IssueInstant [Required]

1628 The time instant of issue of the response. The time value is encoded in UTC, as described in Section
 1629 1.3.3.

1630 Destination [Optional]

1631 A URI reference indicating the address to which this response has been sent. This is useful to prevent
 1632 malicious forwarding of responses to unintended recipients, a protection that is required by some
 1633 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies
 1634 the location at which the message was received. If it does not, the response MUST be discarded.
 1635 Some protocol bindings may require the use of this attribute (see [SAMLBind]).

1636 Consent [Optional]

1637 Indicates whether or not (and under what conditions) consent has been obtained from a principal in
 1638 the sending of this response. See Section 8.4 for some URI references that MAY be used as the
 1639 value of the Consent attribute and their associated descriptions. If no Consent value is provided,
 1640 the identifier urn:oasis:names:tc:SAML:2.0:consent:unspecified (see Section 8.4.1) is in
 1641 effect.

1642 <saml:Issuer> [Optional]

1643 Identifies the entity that generated the response message. (For more information on this element, see
 1644 Section 2.2.5.)

1645 <ds:Signature> [Optional]

1646 An XML Signature that authenticates the responder and provides message integrity, as described
 1647 below and in Section 5.

1648 <Extensions> [Optional]

1649 This extension point contains optional protocol message extension elements that are agreed on
 1650 between the communicating parties. . No extension schema is required in order to make use of this
 1651 extension point, and even if one is provided, the lax validation setting does not impose a requirement

1652 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-
1653 SAML-defined namespace.

1654 <Status> [Required]

1655 A code representing the status of the corresponding request.

1656 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to
1657 authenticate itself, and message integrity may often be required. Authentication and message integrity
1658 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML
1659 response MAY be signed, which provides both authentication of the responder and message integrity.

1660 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML
1661 requester receiving the response MUST verify that the signature is valid (that is, that the message has not
1662 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on
1663 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD
1664 evaluate the signature to determine the identity and appropriateness of the signer and may continue to
1665 process the response as it deems appropriate.

1666 If a Consent attribute is included and the value indicates that some form of principal consent has been
1667 obtained, then the response SHOULD be signed.

1668 The following schema fragment defines the **StatusResponseType** complex type:

```
1669 <complexType name="StatusResponseType">  
1670   <sequence>  
1671     <element ref="saml:Issuer" minOccurs="0"/>  
1672     <element ref="ds:Signature" minOccurs="0"/>  
1673     <element ref="samlp:Extensions" minOccurs="0"/>  
1674     <element ref="samlp:Status"/>  
1675   </sequence>  
1676   <attribute name="ID" type="ID" use="required"/>  
1677   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1678   <attribute name="Version" type="string" use="required"/>  
1679   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1680   <attribute name="Destination" type="anyURI" use="optional"/>  
1681   <attribute name="Consent" type="anyURI" use="optional"/>  
1682 </complexType>
```

1683 3.2.2.1 Element <Status>

1684 The <Status> element contains the following elements:

1685 <StatusCode> [Required]

1686 A code representing the status of the activity carried out in response to the corresponding request.

1687 <StatusMessage> [Optional]

1688 A message which MAY be returned to an operator.

1689 <StatusDetail> [Optional]

1690 Additional information concerning the status of the request.

1691 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1692 <element name="Status" type="samlp:StatusType"/>  
1693 <complexType name="StatusType">  
1694   <sequence>  
1695     <element ref="samlp:StatusCode"/>  
1696     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1697     <element ref="samlp:StatusDetail" minOccurs="0"/>
```

1698 </sequence>
1699 </complexType>

1700 3.2.2.2 Element <StatusCode>

1701 The <StatusCode> element specifies a code or a set of nested codes representing the status of the
1702 corresponding request. The <StatusCode> element has the following element and attribute:

1703 Value [Required]

1704 The status code value. This attribute contains a URI reference. The value of the topmost
1705 <StatusCode> element MUST be from the top-level list provided in this section.

1706 <StatusCode> [Optional]

1707 A subordinate status code that provides more specific information on an error condition. Note that
1708 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for
1709 additional information by intentionally presenting erroneous requests.

1710 The permissible top-level <StatusCode> values are as follows:

1711 urn:oasis:names:tc:SAML:2.0:status:Success

1712 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or
1713 <StatusDetail> elements.

1714 urn:oasis:names:tc:SAML:2.0:status:Requester

1715 The request could not be performed due to an error on the part of the requester.

1716 urn:oasis:names:tc:SAML:2.0:status:Responder

1717 The request could not be performed due to an error on the part of the SAML responder or SAML
1718 authority.

1719 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1720 The SAML responder could not process the request because the version of the request message was
1721 incorrect.

1722 The following second-level status codes are referenced at various places in this specification. Additional
1723 second-level status codes MAY be defined in future versions of the SAML specification. System entities
1724 are free to define more specific status codes by defining appropriate URI references.

1725 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1726 The responding provider was unable to successfully authenticate the principal.

1727 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1728 Unexpected or invalid content was encountered within a <saml:Attribute> or
1729 <saml:AttributeValue> element.

1730 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1731 The responding provider cannot or will not support the requested name identifier policy.

1732 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1733 The specified authentication context requirements cannot be met by the responder.

1734 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1735 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in
1736 an <IDPList> can be resolved or that none of the supported identity providers are available.

1737 urn:oasis:names:tc:SAML:2.0:status:NoPassive
 1738 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1739 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP
 1740 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
 1741 supported by the intermediary.

1742 urn:oasis:names:tc:SAML:2.0:status:PartialLogout
 1743 Used by a session authority to indicate to a session participant that it was not able to propagate
 1744 logout to all other session participants.

1745 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded
 1746 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to
 1747 proxy the request further.

1748 urn:oasis:names:tc:SAML:2.0:status:RequestDenied
 1749 The SAML responder or SAML authority is able to process the request but has chosen not to
 1750 respond. This status code MAY be used when there is concern about the security context of the
 1751 request message or the sequence of request messages received from a particular requester.

1752 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported
 1753 The SAML responder or SAML authority does not support the request.

1754 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated
 1755 The SAML responder cannot process any requests with the protocol version specified in the request.

1756 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh
 1757 The SAML responder cannot process the request because the protocol version specified in the
 1758 request message is a major upgrade from the highest protocol version supported by the responder.

1759 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow
 1760 The SAML responder cannot process the request because the protocol version specified in the
 1761 request message is too low.

1762 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized
 1763 The resource value provided in the request message is invalid or unrecognized.

1764 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses
 1765 The response message would contain more elements than the SAML responder is able to return.

1766 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile
 1767 An entity that has no knowledge of a particular attribute profile has been presented with an attribute
 1768 drawn from that profile.

1769 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal
 1770 The responding provider does not recognize the principal specified or implied by the request.

1771 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding
 1772 The SAML responder cannot properly fulfill the request using the protocol binding specified in the
 1773 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>
```

```
</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. [E45]If ordering is relevant to the evaluation of the request, then 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. For example, ordering is significant when using this element in an `<AuthnRequest>` message but not in an `<AuthnQuery>` message.

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 [E65]top-level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:Responder` and MAY return 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 [E45]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"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

    <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

2046 <Assertion> element and MUST include a <StatusCode> element with the value
2047 urn:oasis:names:tc:SAML:2.0:status:Success.

2048 All other processing rules associated with the underlying request and response messages MUST be
2049 observed.

2050 **3.4 Authentication Request Protocol**

2051 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
2052 authentication statements to establish a security context at one or more relying parties, it can use the
2053 authentication request protocol to send an <AuthnRequest> message element to a SAML authority and
2054 request that it return a <Response> message containing one or more such assertions. Such assertions
2055 MAY contain additional statements of any type, but at least one assertion MUST contain at least one
2056 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

2057 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
2058 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider
2059 is not specified, though the means of authentication might impact the content of the response. Other
2060 issues related to the validation of authentication credentials by the identity provider or any communication
2061 between the identity provider and any other entities involved in the authentication process are also out of
2062 scope of this protocol.

2063 The descriptions and processing rules in the following sections reference the following actors, many of
2064 whom might be the same entity in a particular profile of use:

2065 Requester

2066 The entity who creates the authentication request and to whom the response is to be returned.

2067 Presenter

2068 The entity who presents the request to the identity provider and either authenticates itself during
2069 the transmission of the message, or relies on an existing security context to establish its identity.
2070 If not the requester, the presenter acts as an intermediary between the requester and the
2071 responding identity provider.

2072 Requested Subject

2073 The entity about whom one or more assertions are being requested.

2074 Attesting Entity

2075 The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>
2076 elements of the resulting assertion(s).

2077 Relying Party

2078 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
2079 the profile or context of use, generally to establish a security context.

2080 Identity Provider

2081 The entity to whom the presenter gives the request and from whom the presenter receives the
2082 response.

2083 **3.4.1 Element <AuthnRequest>**

2084 To request that an identity provider issue an assertion with an authentication statement, a presenter
2085 authenticates to that identity provider (or relies on an existing security context) and sends it an
2086 <AuthnRequest> message that describes the properties that the resulting assertion needs to have to

2087 satisfy its purpose. Among these properties may be information that relates to the content of the assertion
2088 and/or information that relates to how the resulting `<Response>` message should be delivered to the
2089 requester. The process of authentication of the presenter may take place before, during, or after the initial
2090 delivery of the `<AuthnRequest>` message.

2091 The requester might not be the same as the presenter of the request if, for example, the requester is a
2092 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject
2093 so that the relying party can decide whether to provide a service.

2094 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected
2095 by the protocol binding used to deliver the message.

2096 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
2097 adds the following elements and attributes, all of which are optional in general, but may be required by
2098 specific profiles:

2099 `<saml:Subject>` [Optional]

2100 Specifies the requested subject of the resulting assertion(s). This may include one or more
2101 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions
2102 can be confirmed. For more information on this element, see Section 2.4.

2103 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the
2104 requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the
2105 presenter is presumed to be the only attesting entity required and the method is implied by the profile
2106 of use and/or the policies of the identity provider.

2107 `<NameIDPolicy>` [Optional]

2108 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
2109 then any type of identifier supported by the identity provider for the requested subject can be used,
2110 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2111 `<saml:Conditions>` [Optional]

2112 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting
2113 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The
2114 information in this element is used as input to the process of constructing the assertion, rather than as
2115 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2116 `<RequestedAuthnContext>` [Optional]

2117 Specifies the requirements, if any, that the requester places on the authentication context that applies
2118 to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing
2119 rules regarding this element.

2120 `<Scoping>` [Optional]

2121 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as
2122 limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity
2123 providers by the responder.

2124 `ForceAuthn` [Optional]

2125 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
2126 rely on a previous security context. If a value is not provided, the default is "false". However, if both
2127 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the
2128 presenter unless the constraints of `IsPassive` can be met.

2129 `IsPassive` [Optional]

2130 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take
2131 control of the user interface from the requester and interact with the presenter in a noticeable fashion.

2132 If a value is not provided, the default is "false".

2133 `AssertionConsumerServiceIndex` [Optional]

2134 Indirectly identifies the location to which the `<Response>` message should be returned to the
2135 requester. It applies only to profiles in which the requester is different from the presenter, such as the
2136 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map
2137 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one
2138 possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message to
2139 the default location associated with the requester for the profile of use. If the index specified is invalid,
2140 then the identity provider MAY return an error `<Response>` or it MAY use the default location. This
2141 attribute is mutually exclusive with the `AssertionConsumerServiceURL` and `ProtocolBinding`
2142 attributes.

2143 `AssertionConsumerServiceURL` [Optional]

2144 Specifies by value the location to which the `<Response>` message MUST be returned to the
2145 requester. The responder MUST ensure by some means that the value specified is in fact associated
2146 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing
2147 `<AuthnRequest>` message is another. This attribute is mutually exclusive with the
2148 `AssertionConsumerServiceIndex` attribute and is typically accompanied by the
2149 `ProtocolBinding` attribute.

2150 `ProtocolBinding` [Optional]

2151 A URI reference that identifies a SAML protocol binding to be used when returning the `<Response>`
2152 message. See [SAMLBind] for more information about protocol bindings and URI references defined
2153 for them. This attribute is mutually exclusive with the `AssertionConsumerServiceIndex` attribute
2154 and is typically accompanied by the `AssertionConsumerServiceURL` attribute.

2155 `AttributeConsumingServiceIndex` [Optional]

2156 Indirectly identifies information associated with the requester describing the SAML attributes the
2157 requester desires or requires to be supplied by the identity provider in the `<Response>` message.
2158 The identity provider MUST have a trusted means to map the index value in the attribute to
2159 information associated with the requester. [SAMLMeta] provides one possible mechanism. The
2160 identity provider MAY use this information to populate one or more `<saml:AttributeStatement>`
2161 elements in the assertion(s) it returns.

2162 `ProviderName` [Optional]

2163 Specifies the human-readable name of the requester for use by the presenter's user agent or the
2164 identity provider.

2165 See Section 3.4.1.4 for general processing rules regarding this message.

2166 The following schema fragment defines the `<AuthnRequest>` element and its **AuthnRequestType**
2167 complex type:

```
2168 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2169 <complexType name="AuthnRequestType">
2170   <complexContent>
2171     <extension base="samlp:RequestAbstractType">
2172       <sequence>
2173         <element ref="saml:Subject" minOccurs="0"/>
2174         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2175         <element ref="saml:Conditions" minOccurs="0"/>
2176         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2177         <element ref="samlp:Scoping" minOccurs="0"/>
2178       </sequence>
2179       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2180       <attribute name="IsPassive" type="boolean" use="optional"/>

```

```

2181         <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2182         <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2183 use="optional"/>
2184         <attribute name="AssertionConsumerServiceURL" type="anyURI"
2185 use="optional"/>
2186         <attribute name="AttributeConsumingServiceIndex"
2187 type="unsignedShort" use="optional"/>
2188         <attribute name="ProviderName" type="string" use="optional"/>
2189     </extension>
2190 </complexContent>
2191 </complexType>

```

2192 3.4.1.1 Element <NameIDPolicy>

2193 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an
 2194 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2195 Format [Optional]

2196 Specifies the URI reference corresponding to a name identifier format defined in this or another
 2197 specification (see Section 8.3 for examples). The additional value of
 2198 `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` is defined specifically for use
 2199 within this attribute to indicate a request that the resulting identifier be encrypted.

2200 SPNameQualifier [Optional]

2201 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of
 2202 a service provider other than the requester, or in the namespace of an affiliation group of service
 2203 providers. See for example the definition of [E84] `urn:oasis:names:tc:SAML:2.0:nameid-`
 2204 `format:persistent` `urn:oasis:names:tc:SAML:1.1:nameid-format:persistent` in
 2205 Section 8.3.7.

2206 AllowCreate [Optional]

2207 A Boolean value used to indicate whether [E14] the requester grants to the identity provider-is allowed,
 2208 in the course of fulfilling the request, permission to create a new identifier-to represent the principal or
 2209 to associate an existing identifier representing the principal with the relying party. Defaults to "false".
 2210 When "false", the requester constrains the identity provider to only issue an assertion to it if an
 2211 acceptable identifier for the principal has already been established. Note that this does not prevent
 2212 the identity provider from creating such identifiers outside the context of this specific request (for
 2213 example, in advance for a large number of principals)-if not present or the entire element is omitted.

2214 The AllowCreate attribute may be used by some deployments to influence the creation of state
 2215 maintained by the identity provider pertaining to the use of a name identifier (or any other persistent,
 2216 uniquely identifying attributes) by a particular relying party, for purposes such as dynamic identifier or
 2217 attribute creation, tracking of consent, subsequent use of the Name Identifier Management protocol
 2218 (see Section 3.6), or other related purposes.

2219 When "false", the requester tries to constrain the identity provider to issue an assertion only if such
 2220 state has already been established or is not deemed applicable by the identity provider to the use of
 2221 an identifier. Thus, this does not prevent the identity provider from assuming such information exists
 2222 outside the context of this specific request (for example, establishing it in advance for a large number
 2223 of principals).

2224 A value of "true" permits the identity provider to take any related actions it wishes to fulfill the request,
 2225 subject to any other constraints imposed by the request and policy (the IsPassive attribute, for
 2226 example).

2227 Generally, requesters cannot assume specific behavior from identity providers regarding the initial
 2228 creation or association of identifiers on their behalf, as these are details left to implementations or

2229 | deployments. Absent specific profiles governing the use of this attribute, it might be used as a hint to
 2230 | identity providers about the requester's intention to store the identifier or link it to a local value.

2231 | A value of "false" might be used to indicate that the requester is not prepared or able to do so and
 2232 | save the identity provider wasted effort.

2233 | Requesters that do not make specific use of this attribute SHOULD generally set it to "true" to
 2234 | maximize interoperability.

2235 | The use of the AllowCreate attribute MUST NOT be used and SHOULD be ignored in conjunction
 2236 | with requests for or assertions issued with name identifiers with a Format of
 2237 | urn:oasis:names:tc:SAML:2.0:nameid-format:transient (they preclude any such state in
 2238 | and of themselves).

2239 | When this element is used, if the content is not understood by or acceptable to the identity provider, then
 2240 | a <Response> message element MUST be returned with an error <Status>, and MAY contain a
 2241 | second-level <StatusCode> of
 2242 | urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

2243 | If the Format value is omitted or set to urn:oasis:names:tc:SAML:2.0:nameid-
 2244 | format:unspecified, then the identity provider is free to return any kind of identifier, subject to any
 2245 | additional constraints due to the content of this element or the policies of the identity provider or principal.

2246 | The special Format value urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates
 2247 | that the resulting assertion(s) MUST contain <EncryptedID> elements instead of plaintext. The
 2248 | underlying name identifier's unencrypted form can be of any type supported by the identity provider for
 2249 | the requested subject. [E6]It is not possible for the service provider to specifically request that a particular
 2250 | kind of identifier be returned if it asks for encryption. The <md:NameIDFormat> metadata element (see
 2251 | [SAMLMeta]) or other out-of-band means MAY be used to determine what kind of identifier to encrypt and
 2252 | return.

2253 | [E15]When a Format defined in Section 8.3 other than urn:oasis:names:tc:SAML:1.1:nameid-
 2254 | format:unspecified or urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is used,
 2255 | then if the identity provider returns any assertions:

- 2256 | • the Format value of the <NameID> within the <Subject> of any <Assertion> MUST be
 2257 | identical to the Format value supplied in the <NameIDPolicy>, and
- 2258 | • if SPNameQualifier is not omitted in <NameIDPolicy>, the SPNameQualifier value of the
 2259 | <NameID> within the <Subject> of any <Assertion> MUST be identical to the
 2260 | SPNameQualifier value supplied in the <NameIDPolicy>.

2261 | Regardless of the Format in the <NameIDPolicy>, the identity provider MAY return an
 2262 | <EncryptedID> in the resulting assertion subject if the policies in effect at the identity provider (possibly
 2263 | specific to the service provider) require that an encrypted identifier be used.

2264 | [E14]Note that if the requester wishes to permit the identity provider to establish a new identifier for the
 2265 | principal if none exists, it MUST include this element with the AllowCreate attribute set to "true".
 2266 | Otherwise, only a principal for whom the identity provider has previously established an identifier usable
 2267 | by the requester can be authenticated successfully. This is primarily useful in conjunction with the
 2268 | urn:oasis:names:tc:SAML:2.0:nameid-format:persistent Format value (see Section 8.3.7).

2269 | The following schema fragment defines the <NameIDPolicy> element and its **NameIDPolicyType**
 2270 | complex type:

```

2271 | <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
2272 | <complexType name="NameIDPolicyType">
2273 |   <attribute name="Format" type="anyURI" use="optional"/>
2274 |   <attribute name="SPNameQualifier" type="string" use="optional"/>
  
```

```

2275     <attribute name="AllowCreate" type="boolean" use="optional"/>
2276 </complexType>

```

2277 3.4.1.2 Element <Scoping>

2278 The <Scoping> element specifies the identity providers trusted by the requester to authenticate the
 2279 presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to
 2280 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
 2281 elements and attribute:

2282 ProxyCount [Optional]

2283 Specifies the number of proxying indirections permissible between the identity provider that receives
 2284 this <AuthnRequest> and the identity provider who ultimately authenticates the principal. A count of
 2285 zero permits no proxying, while omitting this attribute expresses no such restriction.

2286 <IDPList> [Optional]

2287 An advisory list of identity providers and associated information that the requester deems acceptable
 2288 to respond to the request.

2289 <RequesterID> [Zero or More]

2290 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate
 2291 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for
 2292 a description of entity identifiers.

2293 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
 2294 <Response> message with an error <Status> and [E65]optionally a second-level <StatusCode> of
 2295 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or
 2296 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support
 2297 any of the specified identity providers.

2298 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```

2299 <element name="Scoping" type="saml:ScopingType"/>
2300 <complexType name="ScopingType">
2301   <sequence>
2302     <element ref="saml:IDPList" minOccurs="0"/>
2303     <element ref="saml:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2304   </sequence>
2305   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2306 </complexType>
2307 <element name="RequesterID" type="anyURI"/>

```

2308 3.4.1.3 Element <IDPList>

2309 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the
 2310 presenter. Its **IDPListType** complex type defines the following elements:

2311 <IDPEntry> [One or More]

2312 Information about a single identity provider.

2313 <GetComplete> [Optional]

2314 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to
 2315 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML
 2316 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>
 2317 element.

2318 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```

2319 <element name="IDPList" type="samlp:IDPListType"/>
2320 <complexType name="IDPListType">
2321   <sequence>
2322     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2323     <element ref="samlp:GetComplete" minOccurs="0"/>
2324   </sequence>
2325 </complexType>
2326 <element name="GetComplete" type="anyURI"/>

```

2327 3.4.1.3.1 Element <IDPEntry>

2328 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the
 2329 presenter. Its **IDPEntryType** complex type defines the following attributes:

2330 **ProviderID** [Required]

2331 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2332 **Name** [Optional]

2333 A human-readable name for the identity provider.

2334 **Loc** [Optional]

2335 A URI reference representing the location of a profile-specific endpoint supporting the authentication
 2336 request protocol. The binding to be used must be understood from the profile of use.

2337 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```

2338 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2339 <complexType name="IDPEntryType">
2340   <attribute name="ProviderID" type="anyURI" use="required"/>
2341   <attribute name="Name" type="string" use="optional"/>
2342   <attribute name="Loc" type="anyURI" use="optional"/>
2343 </complexType>

```

2344 3.4.1.4 Processing Rules

2345 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
 2346 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
 2347 kinds of input and output are required or prohibited. The following processing rules apply as invariant
 2348 behavior across any profile of this protocol exchange. All other processing rules associated with the
 2349 underlying request and response messages MUST also be observed.

2350 The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing
 2351 one or more assertions that meet the specifications defined by the request, or with a <Response>
 2352 message containing a <Status> describing the error that occurred. The responder MAY conduct
 2353 additional message exchanges with the presenter as needed to initiate or complete the authentication
 2354 process, subject to the nature of the protocol binding and the authentication mechanism. As described in
 2355 the next section, this includes proxying the request by directing the presenter to another identity provider
 2356 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to
 2357 authenticate the presenter to the original responder, in effect using SAML as the authentication
 2358 mechanism.

2359 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if
 2360 prevented from providing an assertion by policies in effect at the identity provider (for example the
 2361 intended subject has prohibited the identity provider from providing assertions to the relying party), then it
 2362 MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> of
 2363 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or
 2364 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2365 If the `<saml:Subject>` element in the request is present, then the resulting assertions'
2366 `<saml:Subject>` MUST **strongly match** the request `<saml:Subject>`, as described in Section 3.3.4,
2367 except that the identifier MAY be in a different format if specified by `<NameIDPolicy>`. In such a case,
2368 the identifier's physical content MAY be different, but it MUST refer to the same principal. [\[E75\]If the](#)
2369 [identity provider cannot or will not produce assertions with a strongly matching subject, then it MUST](#)
2370 [return a `<Response>` with an error `<Status>`, and MAY return a second-level `<StatusCode>` that](#)
2371 [reflects the reason for the failure.](#)

2372 All of the content defined specifically within `<AuthnRequest>` is optional, although some may be
2373 required by certain profiles. In the absence of any specific content at all, the following behavior is implied:

- 2374 • The assertion(s) returned MUST contain a `<saml:Subject>` element that represents the
2375 presenter. The identifier type and format are determined by the identity provider. At least one
2376 statement in at least one assertion MUST be a `<saml:AuthnStatement>` that describes the
2377 authentication performed by the responder or authentication service associated with it.
- 2378 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the
2379 `<saml:SubjectConfirmation>` of the assertion(s). In the case of weaker confirmation
2380 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2381 • The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element
2382 referencing the requester as an acceptable relying party. Other audiences MAY be included as
2383 deemed appropriate by the identity provider.

2384 3.4.1.5 Proxying

2385 If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or
2386 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to
2387 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new
2388 `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in
2389 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying
2390 identity provider.

2391 Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the
2392 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the
2393 proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`,
2394 completing the overall message exchange. Both the proxying and authenticating identity providers MAY
2395 include constraints on proxying activity in the messages and assertions they issue, as described in
2396 previous sections and below.

2397 The requester can influence proxy behavior by including a `<Scoping>` element where the provider sets a
2398 desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by
2399 including an ordered `<IDPList>` of preferred providers.

2400 An identity provider can control secondary use of its assertions by proxying identity providers using a
2401 `<ProxyRestriction>` element in the assertions it issues.

2402 3.4.1.5.1 Proxying Processing Rules

2403 An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is
2404 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
2405 choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

2406 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. [\[65\]Unless the](#)
2407 [identity provider can directly authenticate the presenter, it](#)~~The identity provider~~ MUST return a
2408 [message with an error top level `<StatusCode>` value of](#)

2409 [urn:oasis:names:tc:SAML:2.0:status:Responder](#) containing and may return a second-level
 2410 <StatusCode> value of urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded,
 2411 ~~unless it can directly authenticate the presenter.~~

2412 If it chooses to proxy to a SAML identity provider, when creating the new <AuthnRequest>, the proxying
 2413 identity provider MUST include equivalent or stricter forms of all the information included in the original
 2414 request (such as authentication context policy). Note, however, that the proxying provider is free to
 2415 specify whatever <NameIDPolicy> it wishes to maximize the chances of a successful response.

2416 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
 2417 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for
 2418 example) will be honored by the authenticating provider.

2419 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less
 2420 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new
 2421 request SHOULD contain a <ProxyCount> attribute.

2422 If an <IDPList> was specified in the original request, the new request MUST also contain an
 2423 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the
 2424 <IDPList>, but MUST NOT remove any from the list.

2425 The authentication request and response are processed in normal fashion, in accordance with the rules
 2426 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity
 2427 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2428 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
 2429 relevant information from the original assertion or non-SAML equivalent.
- 2430 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original
 2431 requester's preferences, as defined by its <NameIDPolicy> element.
- 2432 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>
 2433 element containing a <saml:AuthenticatingAuthority> element referencing the identity
 2434 provider to which the proxying identity provider referred the presenter. If the original assertion
 2435 contains <saml:AuthnContext> information that includes one or more
 2436 <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the
 2437 new assertion, with the new element placed after them.
- 2438 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
 2439 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
 2440 consistent over time across different requests. The value MUST not conflict with values used or
 2441 generated by other SAML providers.
- 2442 • Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in
 2443 accordance with the policies of the proxying identity provider, provided that the original
 2444 requirements dictated by the requester are met.

2445 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,
 2446 and this request is equally or less strict than the original request (as determined by the proxying identity
 2447 provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating
 2448 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML
 2449 equivalent it received is still valid).

2450 3.5 Artifact Resolution Protocol

2451 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be
 2452 transported in a SAML binding by reference instead of by value. Both requests and responses can be
 2453 obtained by reference using this specialized protocol. A message sender, instead of binding a message to

2454 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a
2455 variety of forms, but must support a means by which the receiver can determine who sent it. If the
2456 receiver wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML
2457 binding protocol to resolve the artifact into the original protocol message.

2458 The most common use for this mechanism is with bindings that cannot easily carry a message because of
2459 size constraints, or to enable a message to be communicated via a secure channel between the SAML
2460 requester and responder, avoiding the need for a signature.

2461 Depending on the characteristics of the underlying message being passed by reference, the artifact
2462 resolution protocol MAY require protections such as mutual authentication, integrity protection,
2463 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST
2464 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used
2465 by any party.

2466 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly
2467 as if the message so obtained had been sent originally in place of the artifact.

2468 3.5.1 Element <ArtifactResolve>

2469 The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an
2470 <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message.
2471 The original transmission of the artifact is governed by the specific protocol binding that is being used;
2472 see [SAMLBind] for more information on the use of artifacts in bindings.

2473 The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity
2474 protected by the protocol binding used to deliver the message.

2475 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and
2476 adds the following element:

2477 <Artifact> [Required]

2478 The artifact value that the requester received and now wishes to translate into the protocol message it
2479 represents. See [SAMLBind] for specific artifact format information.

2480 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**
2481 complex type:

```
2482 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2483 <complexType name="ArtifactResolveType">
2484   <complexContent>
2485     <extension base="samlp:RequestAbstractType">
2486       <sequence>
2487         <element ref="samlp:Artifact"/>
2488       </sequence>
2489     </extension>
2490   </complexContent>
2491 </complexType>
2492 <element name="Artifact" type="string"/>
```

2493 3.5.2 Element <ArtifactResponse>

2494 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>
2495 message element. This element is of complex type **ArtifactResponseType**, which extends
2496 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol
2497 message being returned. This wrapped message element can be a request or a response.

2498 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2499 protected by the protocol binding used to deliver the message.

2500 The following schema fragment defines the <ArtifactResponse> element and its
2501 **ArtifactResponseType** complex type:

```
2502 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2503 <complexType name="ArtifactResponseType">
2504   <complexContent>
2505     <extension base="samlp:StatusResponseType">
2506       <sequence>
2507         <any namespace="##any" processContents="lax" minOccurs="0"/>
2508       </sequence>
2509     </extension>
2510   </complexContent>
2511 </complexType>
```

2512 3.5.3 Processing Rules

2513 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in
2514 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>
2515 element with no embedded message. In both cases, the <Status> element MUST include a
2516 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A
2517 response message with no embedded message inside it is termed an empty response in the remainder of
2518 this section.

2519 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent
2520 request with the same artifact by any requester results in an empty response as described above.

2521 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY
2522 be intended for consumption by any party that receives it and can respond appropriately. In most other
2523 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST
2524 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer
2525 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the
2526 original intended recipient, then the artifact issuer MUST return an empty response.

2527 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2528 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2529 return it in an <ArtifactResolve> message to the issuer.

2530 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of
2531 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message
2532 will contain its own message identifier, and in the case of an embedded response, may contain a different
2533 InResponseTo value that corresponds to the original request message to which the embedded message
2534 is responding.

2535 All other processing rules associated with the underlying request and response messages MUST be
2536 observed.

2537 3.6 Name Identifier Management Protocol

2538 After establishing a name identifier for a principal, an identity provider wishing to change the value
2539 ~~[E12]and/or format~~ of the identifier that it will use when referring to the principal, or to indicate that a name
2540 identifier will no longer be used to refer to the principal, informs service providers of the change by
2541 sending them a <ManageNameIDRequest> message.

2542 A service provider also uses this message to register or change the `SPProvidedID` value to be included
2543 when the underlying name identifier is used to communicate with it, or to terminate the use of a name
2544 identifier between itself and the identity provider.

2545 ~~[E14]Note that this protocol is typically not used with "transient" name identifiers, since their value is not~~
2546 ~~intended to be managed on a long term basis.This protocol MUST NOT be used in conjunction with the~~
2547 ~~urn:oasis:names:tc:SAML:2.0:nameidformat:transient <NameID> Format.~~

2548 3.6.1 Element **<ManageNameIDRequest>**

2549 A provider sends a **<ManageNameIDRequest>** message to inform the recipient of a changed name
2550 identifier or to indicate the termination of the use of a name identifier.

2551 The **<ManageNameIDRequest>** message SHOULD be signed or otherwise authenticated and integrity
2552 protected by the protocol binding used to deliver the message.

2553 This message has the complex type **ManageNameIDRequestType**, which extends
2554 **RequestAbstractType** and adds the following elements:

2555 **<saml:NameID>** or **<saml:EncryptedID>** [Required]

2556 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the
2557 principal as currently recognized by the identity and service providers prior to this request. (For more
2558 information on these elements, see Section 2.2.)

2559 **<NewID>** or **<NewEncryptedID>** or **<Terminate>** [Required]

2560 The new identifier value (in plaintext or encrypted form) to be used when communicating with the
2561 requesting provider concerning this principal, or an indication that the use of the old identifier has
2562 been terminated. In the former case, if the requester is the service provider, the new identifier MUST
2563 appear in subsequent **<NameID>** elements in the `SPProvidedID` attribute. If the requester is the
2564 identity provider, the new value will appear in subsequent **<NameID>** elements as the element's
2565 content. ~~[E12]In either case, if the <NewEncryptedID> is used, its encrypted content is just a~~
2566 ~~<NewID> element containing only the new value for the identifier (format and qualifiers cannot be~~
2567 ~~changed once established).~~

2568 The following schema fragment defines the **<ManageNameIDRequest>** element and its
2569 **ManageNameIDRequestType** complex type:

```
2570 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2571 <complexType name="ManageNameIDRequestType">
2572   <complexContent>
2573     <extension base="samlp:RequestAbstractType">
2574       <sequence>
2575         <choice>
2576           <element ref="saml:NameID"/>
2577           <element ref="saml:EncryptedID"/>
2578         </choice>
2579         <choice>
2580           <element ref="samlp:NewID"/>
2581           <element ref="samlp:NewEncryptedID"/>
2582           <element ref="samlp:Terminate"/>
2583         </choice>
2584       </sequence>
2585     </extension>
2586   </complexContent>
2587 </complexType>
2588 <element name="NewID" type="string"/>
2589 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2590 <element name="Terminate" type="samlp:TerminateType"/>
2591 <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 [E55]using that identifierabout the principal. The receiving provider can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated. [E8] In general it SHOULD NOT invalidate any active session(s) of the principal for whom the relationship has been terminated. If the receiving provider is an identity provider, it SHOULD NOT invalidate any active session(s) of the principal established with other service providers. A requesting provider MAY send a <LogoutRequest> message prior to initiating a name identifier termination by sending a <ManageNameIDRequest> message if that is the requesting provider's intent (e.g., the name identifier termination is initiated via an administrator who wished to terminate all user activity). The requesting provider MUST NOT send a <LogoutRequest> message after the <ManageNameIDRequest> message is sent. It can choose to invalidate the active session(s) of a principal for whom a relationship has been terminated.

[E14] If the receiving provider is maintaining state associated with the name identifier, such as the value of the identifier itself (in the case of a pair-wise identifier), an SPProvidedID value, the sender's consent to the identifier's creation/use, etc., then the receiver can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated.

Any subsequent operations performed by the receiver on behalf of the sender regarding the principal (for example, a subsequent <AuthnRequest>) SHOULD be carried out in a manner consistent with the absence of any previous state.

Termination is potentially the cleanup step for any state management behavior triggered by the use of the AllowCreate attribute in the Authentication Request protocol (see Section 3.4). Deployments that do not make use of that attribute are likely to avoid the use of the <Terminate> element or would treat it as a purely advisory matter.

Note that in most cases (a notable exception being the rules surrounding the SPProvidedID attribute), there are no requirements on either identity providers or service providers regarding the creation or use of persistent state. Therefore, no explicit behavior is mandated when the <Terminate> element is received. However, if persistent state is present pertaining to the use of an identifier (such as if an SPProvidedID attribute was attached), the <Terminate> element provides a clear indication that this state SHOULD be deleted (or marked as obsolete in some fashion).

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

2636 | SPProvidedID when subsequently communicating to the service provider **regarding this**
 2637 | **principal[E55]using the primary identifier.**

2638 | If the identity provider requests that its identifier for the principal be changed by including a <NewID> (or
 2639 | <NewEncryptedID>) element, the service provider MUST use the element's content as the
 2640 | <saml:NameID> element content when subsequently communicating with the identity provider **[E55]in**
 2641 | **any case where the identifier being changed would have been usedregarding this principal.**

2642 | Note that neither, either, or both of the original and new identifier MAY be encrypted (using the
 2643 | <EncryptedID> and <NewEncryptedID> elements).

2644 | In any case, the <saml:NameID> content in the request and its associated SPProvidedID attribute
 2645 | MUST contain the most recent name identifier information established between the providers for the
 2646 | principal.

2647 | In the case of an identifier with a Format of urn:oasis:names:tc:SAML:2.0:nameid-
 2648 | format:persistent, the NameQualifier attribute MUST contain the unique identifier of the identity
 2649 | provider that created the identifier. If the identifier was established between the identity provider and an
 2650 | affiliation group of which the service provider is a member, then the SPNameQualifier attribute MUST
 2651 | contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the
 2652 | service provider. These attributes MAY be omitted if they would otherwise match the value of the
 2653 | containing protocol message's <Issuer> element, but this is NOT RECOMMENDED due to the
 2654 | opportunity for confusion.

2655 | Changes to these identifiers may take a potentially significant amount of time to propagate through the
 2656 | systems at both the requester and the responder. Implementations might wish to allow each party to
 2657 | accept either identifier for some period of time following the successful completion of a name identifier
 2658 | change. Not doing so could result in the inability of the principal to access resources.

2659 | All other processing rules associated with the underlying request and response messages MUST be
 2660 | observed.

2661 | 3.7 Single Logout Protocol

2662 | The single logout protocol provides a message exchange protocol by which all sessions provided by a
 2663 | particular session authority are near-simultaneously terminated. The single logout protocol is used either
 2664 | when a principal logs out at a session participant or when the principal logs out directly at the
 2665 | session authority. This protocol may also be used to log out a principal due to a timeout. The reason for
 2666 | the logout event can be indicated through the Reason attribute.

2667 | The principal may have established authenticated sessions with both the session authority and individual
 2668 | session participants, based on assertions containing authentication statements supplied by the session
 2669 | authority.

2670 | When the principal invokes the single logout process at a session participant, the session participant
 2671 | MUST send a <LogoutRequest> message to the session authority that provided the assertion
 2672 | containing the authentication statement related to that session at the session participant.

2673 | When either the principal invokes a logout at the session authority, or a session participant sends a logout
 2674 | request to the session authority specifying that principal, the session authority SHOULD send a
 2675 | <LogoutRequest> message to each session participant to which it provided assertions containing
 2676 | authentication statements under its current session with the principal, with the exception of the session
 2677 | participant that sent the <LogoutRequest> message to the session authority. It SHOULD attempt to
 2678 | contact as many of these participants as it can using this protocol, terminate its own session with the
 2679 | 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. [\[E92\] As noted in section 1.3.3, relying parties SHOULD allow for reasonable clock skew in the interpretation of both values.](#)

Reason [Optional]

An indication of the reason for the logout, in the form of a URI reference. [\[E10\] The Reason attribute is specified as a string in the schema. This specification further restricts the schema by requiring that the Reason attribute MUST be 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]

[\[E38\] The index of the session between the principal identified by the <saml:BaseID>, <saml:NameID>, or <saml:EncryptedID> element, and the session authority. This must correlate to the SessionIndex attribute, if any, in the <saml:AuthnStatement> of the assertion used to establish the session that is being terminated. 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).

2771 **Note:** This rule is intended to prevent a situation in which a session participant receives a
 2772 logout request targeted at a single, or multiple, assertion(s) (as identified by the
 2773 <SessionIndex> element(s)) *before* it receives the actual – and possibly still valid –
 2774 assertion(s) targeted by the logout request. It should honor the logout request until the
 2775 logout request itself may be discarded (the *NotOnOrAfter* value on the request has
 2776 been exceeded) or the assertion targeted by the logout request has been received and
 2777 has been handled appropriately.

2778 3.7.3.2 Session Authority Rules

2779 When a session authority receives a <LogoutRequest> message, the session authority **MUST**
 2780 authenticate the sender. If the sender is a session participant to which the session authority provided an
 2781 assertion containing an authentication statement for the current session, then the session authority
 2782 **SHOULD** do the following in the specified order:

- 2783 • Send a <LogoutRequest> message to any session authority on behalf of whom the session
 2784 authority proxied the principal's authentication, unless the second authority is the originator of the
 2785 <LogoutRequest>.
- 2786 • Send a <LogoutRequest> message to each session participant for which the session authority
 2787 provided assertions in the current session, *other than* the originator of a current
 2788 <LogoutRequest>.
- 2789 • Terminate the principal's current session as specified by the <saml:BaseID>, <saml:NameID>,
 2790 or <saml:EncryptedID> element, and any <SessionIndex> elements present in the logout
 2791 request message.

2792 If the session authority successfully terminates the principal's session with respect to itself, then it **MUST**
 2793 respond to the original requester, if any, with a <LogoutResponse> message containing a top-level
 2794 status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it **MUST**
 2795 respond with a <LogoutResponse> message containing a top-level status code indicating the error.
 2796 Thus, the top-level status indicates the state of the logout operation only with respect to the session
 2797 authority itself.

2798 The session authority **SHOULD** attempt to contact each session participant using any applicable/usable
 2799 protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because
 2800 the original request takes place using a protocol binding that does not enable the logout to be propagated
 2801 to all participants).

2802 In the event that not all session participants successfully respond to these <LogoutRequest> messages
 2803 (or if not all participants can be contacted), then the session authority **MUST** include in its
 2804 <LogoutResponse> message a second-level status code of
 2805 `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session
 2806 participants successfully responded with confirmation of the logout.

2807 Note that a session authority **MAY** initiate a logout for reasons other than having received a
 2808 <LogoutRequest> from a session participant – these include, but are not limited to:

- 2809 • If some timeout period was agreed out-of-band with an individual session participant, the session
 2810 authority **MAY** send a <LogoutRequest> to that individual participant alone.
- 2811 • An agreed global timeout period has been exceeded.
- 2812 • The principal or some other trusted entity has requested logout of the principal directly at the
 2813 session authority.
- 2814 • The session authority has determined that the principal's credentials may have been compromised.

2815 When constructing a logout request message, the session authority MUST set the value of the
2816 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,
2817 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time
2818 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most
2819 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout
2820 request).

2821 In addition to the values specified in Section [E0] 3.7.33-6.3 for the `Reason` attribute, the following values
2822 are also available for use by the session authority only:

2823 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2824 Specifies that the message is being sent because of the global session timeout interval period
2825 being exceeded.

2826 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

2827 Specifies that the message is being sent because a timeout interval period agreed between a
2828 participant and the session authority has been exceeded.

2829 3.8 Name Identifier Mapping Protocol

2830 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2831 identifier for the same principal in a particular format or federation namespace, it can send a request to
2832 the identity provider using this protocol.

2833 For example, a service provider that wishes to communicate with another service provider with whom it
2834 does not share an identifier for the principal can use an identity provider that shares an identifier for the
2835 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,
2836 with which it can communicate with the second service provider.

2837 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a
2838 `<saml:EncryptedID>` element unless a specific deployment dictates such protection is unnecessary.

2839 3.8.1 Element `<NameIDMappingRequest>`

2840 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2841 `<NameIDMappingRequest>` message. This message has the complex type
2842 **`NameIDMappingRequestType`**, which extends **`RequestAbstractType`** and adds the following elements:

2843 `<saml:BaseID>` or `<saml:NameID>` or `<saml:EncryptedID>` [Required]

2844 The identifier and associated descriptive data that specify the principal as currently recognized by the
2845 requester and the responder. (For more information on this element, see Section 2.2.)

2846 `<NameIDPolicy>` [Required]

2847 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2848 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2849 binding used to deliver the message.

2850 The following schema fragment defines the `<NameIDMappingRequest>` element and its
2851 **`NameIDMappingRequestType`** complex type:

```
2852 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2853 <complexType name="NameIDMappingRequestType">
2854   <complexContent>
2855     <extension base="samlp:RequestAbstractType">
```

```

2856         <sequence>
2857             <choice>
2858                 <element ref="saml:BaseID"/>
2859                 <element ref="saml:NameID"/>
2860                 <element ref="saml:EncryptedID"/>
2861             </choice>
2862             <element ref="samlp:NameIDPolicy"/>
2863         </sequence>
2864     </extension>
2865 </complexContent>
2866 </complexType>

```

2867 3.8.2 Element <NameIDMappingResponse>

2868 The recipient of a <NameIDMappingRequest> message MUST respond with a
2869 <NameIDMappingResponse> message. This message has the complex type
2870 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2871 <saml:NameID> or <saml:EncryptedID> [Required]

2872 The identifier and associated attributes that specify the principal in the manner requested, usually in
2873 encrypted form. (For more information on this element, see Section 2.2.)

2874 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2875 binding used to deliver the message.

2876 The following schema fragment defines the <NameIDMappingResponse> element and its
2877 **NameIDMappingResponseType** complex type:

```

2878 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2879 <complexType name="NameIDMappingResponseType">
2880     <complexContent>
2881         <extension base="samlp:StatusResponseType">
2882             <choice>
2883                 <element ref="saml:NameID"/>
2884                 <element ref="saml:EncryptedID"/>
2885             </choice>
2886         </extension>
2887     </complexContent>
2888 </complexType>

```

2889 3.8.3 Processing Rules

2890 If the responder does not recognize the principal identified in the request, it MAY respond with an error
2891 <Status>, [E65]optionally containing a second-level <StatusCode> of
2892 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2893 At the responder's discretion, the
2894 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to
2895 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2896 All other processing rules associated with the underlying request and response messages MUST be
2897 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 *Major_B.Minor_B* is higher than the version number *Major_A.Minor_A* if and only if:

(Major_B > Major_A) OR ((Major_B = Major_A) AND (Minor_B > Minor_A))

4.1 SAML Specification Set Version

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

The overall size and scope of changes to the specification set documents will informally dictate whether a set of changes constitutes a major or minor revision. In general, if the specification set is backwards compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major revision.

4.1.1 Schema Version

As a non-normative documentation mechanism, any XML schema documents published as part of the specification set will contain a `version` attribute on the `<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

2940 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For
2941 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the
2942 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the
2943 likely effects of schema evolution.)

2944 4.1.3 SAML Protocol Version

2945 The various SAML protocols' request and response elements contain an attribute for expressing the major
2946 and minor version of the request or response message using a string of the form *Major.Minor*. Each
2947 version of the SAML specification set will be construed so as to document the syntax, semantics, and
2948 processing rules of the protocol messages of the same version. That is, specification set version 1.0
2949 describes request and response version V1.0, and so on.

2950 There is explicitly NO relationship between the protocol version and the target XML namespace specified
2951 for the schema definitions for that protocol version.

2952 The version numbers used in SAML protocol request and response elements will match for any particular
2953 revision of the SAML specification set.

2954 4.1.3.1 Request Version

2955 The following processing rules apply to requests:

- 2956 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2957 SAML requester and the SAML responder.
- 2958 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD
2959 assume that the responder supports requests with the highest request version supported by the
2960 requester.
- 2961 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version
2962 number matching a response version number that the requester does not support.
- 2963 • A SAML responder MUST reject any request with a major request version number not supported by
2964 the responder.
- 2965 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2966 higher than the highest supported request version that it supports. However, all requests that share
2967 a major request version number MUST share the same general processing rules and semantics,
2968 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the
2969 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill
2970 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

2971 4.1.3.2 Response Version

2972 The following processing rules apply to responses:

- 2973 • A SAML responder MUST NOT issue a response message with a response version number higher
2974 than the request version number of the corresponding request message.
- 2975 • A SAML responder MUST NOT issue a response message with a major response version number
2976 lower than the major request version number of the corresponding request message except to
2977 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2978 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a
2979 top-level `<StatusCode>` value of
2980 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting
2981 one of the following second-level values:

2982 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,
2983 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or
2984 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

2985 4.1.3.3 Permissible Version Combinations

2986 Assertions of a particular major version appear only in response messages of the same major version, as
2987 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For
2988 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1
2989 response message, if the appropriate assertion schema is referenced during namespace importation. But
2990 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major
2991 versions.

2992 4.2 SAML Namespace Version

2993 XML schema documents published as part of the specification set contain one or more target
2994 namespaces into which the type, element, and attribute definitions are placed. Each namespace is
2995 distinct from the others, and represents, in shorthand, the structural and syntactic definitions that make up
2996 that part of the specification.

2997 The namespace URI references defined by the specification set will generally contain version information
2998 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST
2999 correspond to the major and minor version of the specification set in which the namespace is first
3000 introduced and defined. This information is not typically consumed by an XML processor, which treats the
3001 namespace opaquely, but is intended to communicate the relationship between the specification set and
3002 the namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that
3003 are listed in Section 8.

3004 As a general rule, implementers can expect the namespaces and the associated schema definitions
3005 defined by a major revision of the specification set to remain valid and stable across minor revisions of
3006 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,
3007 but this is expected to be rare. In such cases, the older namespaces and their associated definitions
3008 should be expected to remain valid until a major specification set revision.

3009 4.2.1 Schema Evolution

3010 In general, maintaining namespace stability while adding or changing the content of a schema are
3011 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
3012 older implementations will react to any given change, making forward compatibility difficult to achieve.
3013 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
3014 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),
3015 implementations should expect forward-compatible schema changes in minor revisions, allowing new
3016 messages to validate against older schemas.

3017 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
3018 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
3019 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such
3020 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
3021 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. [\[E81\]Any algorithm defined for use with the XML Signature specification MAY be used. 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. [\[E83\]Use of Exclusive Canonicalization facilitates the verification of signatures created over SAML messages when placed into a different XML context than present during signing.](#)

[Note that use of this algorithm alone does not guarantee that a particular signed object can be moved from one context to another safely, nor is that a requirement of signed SAML objects in general, though it](#)

3106 | MAY be required by particular profiles Use of Exclusive Canonicalization ensures that signatures created
3107 | over SAML messages embedded in an XML context can be verified independent of that context.

3108 | 5.4.4 Transforms

3109 | Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
3110 | transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
3111 | canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
3112 | <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

3113 | Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
3114 | not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can
3115 | be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
3116 | applying the transforms manually to the content and reverifying the result as consisting of the same SAML
3117 | message.

3118 | 5.4.5 [E91] Object

3119 | The <ds:Object> element is not defined for use with SAML signatures, and SHOULD NOT be present.
3120 | Since it can be used in service of an attacker by carrying unsigned data, verifiers SHOULD reject
3121 | signatures that contain a <ds:Object> element.

3122 | 5.4.6 KeyInfo

3123 | XML Signature defines usage of the <ds:KeyInfo> element. SAML does not require the use of
3124 | <ds:KeyInfo>, nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY be
3125 | absent.

3126 | 5.4.7 Example

3127 | Following is an example of a signed response containing a signed assertion. Line breaks have been
3128 | added for readability; the signatures are not valid and cannot be successfully verified.

```
3129 | <Response  
3130 |   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"  
3131 |   ID="_c7055387-af61-4fce-8b98-e2927324b306"  
3132 |   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"  
3133 |   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">  
3134 |   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>  
3135 |   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
3136 |     <ds:SignedInfo>  
3137 |       <ds:CanonicalizationMethod  
3138 |         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
3139 |       <ds:SignatureMethod  
3140 |         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
3141 |       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
3142 |         <ds:Transforms>  
3143 |           <ds:Transform  
3144 |             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-  
3145 | signature" />  
3146 |           <ds:Transform  
3147 |             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">  
3148 |             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"  
3149 |               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />  
3150 |           </ds:Transform>  
3151 |         </ds:Transforms>  
3152 |       <ds:DigestMethod
```

```

3153         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3154         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
3155     </ds:Reference>
3156 </ds:SignedInfo>
3157 <ds:SignatureValue>
3158     x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
3159     EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaKlywS7gFgsD01qjyen3CP+m3D
3160     w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
3161 </ds:SignatureValue>
3162 <ds:KeyInfo>
3163     <ds:X509Data>
3164         <ds:X509Certificate>
3165             MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTA1VT
3166             MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3167             FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
3168             bmZvcmlhdGlvb2ZlbnNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgcQ0Eg
3169             LS0gMjAwMjA3MDFBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
3170             CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbGJESMBAGAlUEBxMJQW5uIEFy
3171             Ym9yMQ4wDAYDVQQKEwVVQ0FJRDECMBoGAlUEAxMTc2hpYjEuaW50ZXJ1ZXQyLmVh
3172             dTEuMCUGCSqGSIB3DQEBQUAA4GNADCBiQKBgQDZSAb2sxvhaXNlVt8vUray+x50z7GJj
3173             CSqGSIB3DQEBQUAA4GNADCBiQKBgQDZSAb2sxvhaXNlVt8vUray+x50z7GJj
3174             IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrCP+
3175             c2M2VeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
3176             pmqOI fGTWQIDAQABox0wGzAMBGNVHRMBAf8EAjAAMAsGAlUdDwQEAWIFoDANBgkq
3177             hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3178             qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
3179             8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
3180         </ds:X509Certificate>
3181     </ds:X509Data>
3182 </ds:KeyInfo>
3183 </ds:Signature>
3184 <Status>
3185     <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3186 </Status>
3187 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3188     IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3189     xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3190     <Issuer>https://www.opensaml.org/IDP</Issuer>
3191     <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3192         <ds:SignedInfo>
3193             <ds:CanonicalizationMethod>
3194                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
3195             <ds:SignatureMethod>
3196                 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3197             <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3198                 <ds:Transforms>
3199                     <ds:Transform
3200                         Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3201 signature"/>
3202                     <ds:Transform
3203                         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3204                         <InclusiveNamespaces
3205                             PrefixList="#default saml ds xs xsi"
3206                             xmlns="http://www.w3.org/2001/10/xml-exc-c14n#">
3207                         </ds:Transform>
3208                     </ds:Transforms>
3209                     <ds:DigestMethod>
3210                         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3211                     <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3212                 </ds:Reference>
3213             </ds:SignedInfo>
3214             <ds:SignatureValue>
3215                 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n

```

```

3216         7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmt3TD
3217         MwUL/cBUj2OtBZQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
3218     </ds:SignatureValue>
3219     <ds:KeyInfo>
3220         <ds:X509Data>
3221             <ds:X509Certificate>
3222                 MIICyCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTA1VT
3223                 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
3224                 FlVuaXZlcnpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lubiBvZiBJ
3225                 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3226                 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoxDTA2MDkwNDA3Mjc1MVowYysx
3227                 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
3228                 Ym9yMQ4wDAYDVQQKEwVWQ0FJRDECMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVh
3229                 dTEuMCUGCSqGSIB3DQEJARYYcm9vdEBzaGliMS5pbmRlcm5ldDIuZWZWR1MIGfMAOG
3230                 CSqGSIB3DQEBAAQUAA4GNADCBiQKBgQDZSAb2sxvvhAXnXVIVTxs8vuRay+x50z7GJj
3231                 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZB113R6+KYIE7x4XAWIrCP+
3232                 c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
3233                 pmqOI fGTWQIDAQABox0wGzAMBGNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3234                 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
3235                 qgi7lFV6MDkkmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
3236                 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpRlylGPdiowMMNTrEG8cCx3w/w==
3237             </ds:X509Certificate>
3238         </ds:X509Data>
3239     </ds:KeyInfo>
3240 </ds:Signature>
3241 <Subject>
3242     <NameID
3243         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3244         scott@example.org
3245     </NameID>
3246     <SubjectConfirmation
3247         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3248 </Subject>
3249 <Conditions NotBefore="2003-04-17T00:46:02Z"
3250     NotOnOrAfter="2003-04-17T00:51:02Z">
3251     <AudienceRestriction>
3252         <Audience>http://www.opensaml.org/SP</Audience>
3253     </AudienceRestriction>
3254 </Conditions>
3255 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3256     <AuthnContext>
3257         <AuthnContextClassRef>
3258             urn:oasis:names:tc:SAML:2.0:ac:classes>Password
3259         </AuthnContextClassRef>
3260     </AuthnContext>
3261 </AuthnStatement>
3262 </Assertion>
3263 </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 [E30]optionally zeroone 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/xmenc#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 [E93] Encryption and Integrity Protection

SAML allows for assertions containing encrypted elements to be integrity protected, and allows for encrypted assertions to be included inside protocol response elements that are themselves integrity protected (typically via XML Signature, or in some cases through binding-specific mechanisms such as TLS).

Recent practical attacks against the most common algorithms (at the time of this writing) used for bulk data encryption in [XMLEnc], which operate in CBC-mode, necessitate the enforcement of integrity protection by a relying party prior to processing encrypted data. As a result, when CBC-mode algorithms are used for data encryption, relying parties SHOULD require the presence of integrity protection before processing encrypted SAML assertions or assertions containing encrypted data. The most appropriate means of achieving this will vary by profile, but may involve the use of authenticated TLS requests, or a requirement for an authenticated digital signature at a layer above that of the encrypted elements.

The ability to protect the encryption layer via a signature or TLS is limited by the fact that one typically does not have the ability to relate the asserting party's key to the cipher key. Thus, while one can limit exposure to only trusted asserting parties (via their key), it will often be the case that any trusted party's key will be accepted for the purposes of exploiting this issue.

Other countermeasures, such as attempting to mitigate timing attacks, or limiting reuse of encryption keys, tend to be impractical for most implementations and the use of integrity protection, when properly implemented, is the suggested solution if authenticated encryption modes are unavailable.

6.3 [E43] Key and Data Referencing Guidelines

If an encrypted key is NOT included in the XML instance, then the relying party must be able to locally determine the decryption key, per [XMLEnc].

Implementations of SAML MAY implicitly associate keys with the corresponding data they are used to encrypt, through the positioning of `<xenc:EncryptedKey>` elements next to the associated `<xenc:EncryptedData>` element, within the enclosing SAML parent element. However, the following set of explicit referencing guidelines are suggested to facilitate interoperability.

If the encrypted key is included in the XML instance, then it SHOULD be referenced within the associated `<xenc:EncryptedData>` element, or alternatively embedded within the `<xenc:EncryptedData>` element. When an `<xenc:EncryptedKey>` element is used, the `<ds:KeyInfo>` element within `<xenc:EncryptedData>` SHOULD reference the `<xenc:EncryptedKey>` element using a `<ds:RetrievalMethod>` element of Type `http://www.w3.org/2001/04/xmenc#EncryptedKey`.

In addition, an `<xenc:EncryptedKey>` element SHOULD contain an `<xenc:ReferenceList>` element containing a `<xenc:DataReference>` that references the corresponding `<xenc:EncryptedData>` element(s) that the key was used to encrypt.

In scenarios where the encrypted element is being "multicast" to multiple recipients, and the key used to encrypt the message must be in turn encrypted individually and independently for each of the multiple recipients, the `<xenc:CarriedKeyName>` element SHOULD be used to assign a common name to each of the `<xenc:EncryptedKey>` elements so that a `<ds:KeyName>` can be used from within the `<xenc:EncryptedData>` element's `<ds:KeyInfo>` element.

Within the `<xenc:EncryptedData>` element, the `<ds:KeyName>` can be thought of as an "alias" that is used for backwards referencing from the `<xenc:CarriedKeyName>` element in each individual `<xenc:EncryptedKey>` element. While this accommodates a "multicast" approach, each recipient must be able to understand (at least one) `<ds:KeyName>`. The `Recipient` attribute is used to provide a hint as to which key is meant for which recipient.

The SAML implementation has the discretion to accept or reject a message where multiple `Recipient` attributes or `<ds:KeyName>` elements are understood. It is RECOMMENDED that implementations simply use the first key they understand and ignore any additional keys. 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.

Combining Signatures and Encryption

6.4 Examples

In the following example, the parent element (`<EncryptedID>`) contains `<xenc:EncryptedData>` and (referenced) `<xenc:EncryptedKey>` elements as siblings (note that the key can in fact be anywhere in the same instance, and the key references the `<xenc:EncryptedData>` element):

```
<saml:EncryptedID xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
    Id="Encrypted_DATA_ID"
    Type="http://www.w3.org/2001/04/xmenc#Element">
    <xenc:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmenc#aes128-cbc"/>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:RetrievalMethod URI="#Encrypted_KEY_ID">
```

```

3354         Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
3355     </ds:KeyInfo>
3356     <xenc:CipherData>
3357         <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3358     </xenc:CipherData>
3359 </xenc:EncryptedData>
3360
3361     <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3362         Id="Encrypted_KEY_ID">
3363         <xenc:EncryptionMethod
3364             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3365         <xenc:CipherData>
3366             <xenc:CipherValue>PzA5X...</xenc:CipherValue>
3367         </xenc:CipherData>
3368         <xenc:ReferenceList>
3369             <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3370         </xenc:ReferenceList>
3371     </xenc:EncryptedKey>
3372 </saml:EncryptedID>

```

3373 In the following `<EncryptedAttribute>` example, the `<xenc:EncryptedKey>` element is contained
3374 within the `<xenc:EncryptedData>` element, so there is no explicit referencing:

```

3375 <saml:EncryptedAttribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3376     <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3377         Id="Encrypted_DATA_ID"
3378         Type="http://www.w3.org/2001/04/xmlenc#Element">
3379         <xenc:EncryptionMethod
3380             Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
3381         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3382             <xenc:EncryptedKey Id="Encrypted_KEY_ID">
3383                 <xenc:EncryptionMethod
3384                     Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3385                 <xenc:CipherData>
3386                     <xenc:CipherValue>SDFSDF... </xenc:CipherValue>
3387                 </xenc:CipherData>
3388             </xenc:EncryptedKey>
3389         </ds:KeyInfo>
3390         <xenc:CipherData>
3391             <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3392         </xenc:CipherData>
3393     </xenc:EncryptedData>
3394 </saml:EncryptedAttribute>

```

3395 The final example shows an assertion encrypted for multiple recipients, using the
3396 `<xenc:CarriedKeyName>` approach:

```

3397 <saml:EncryptedAssertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
3398     <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3399         Id="Encrypted_DATA_ID"
3400         Type="http://www.w3.org/2001/04/xmlenc#Element">
3401         <xenc:EncryptionMethod
3402             Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
3403         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3404             <ds:KeyName>MULTICAST_KEY_NAME</ds:KeyName>
3405         </ds:KeyInfo>
3406         <xenc:CipherData>
3407             <xenc:CipherValue>Nk4W4mx...</xenc:CipherValue>
3408         </xenc:CipherData>
3409     </xenc:EncryptedData>
3410
3411     <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3412         Id="Encrypted_KEY_ID_1" Recipient="https://sp1.org">

```

```

3413     <xenc:EncryptionMethod
3414         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3415     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3416 <ds:KeyName>KEY_NAME_1</ds:KeyName>
3417     </ds:KeyInfo>
3418     <xenc:CipherData>
3419 <xenc:CipherValue>xyzABC...</xenc:CipherValue>
3420 </xenc:CipherData>
3421     <xenc:ReferenceList>
3422         <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3423     </xenc:ReferenceList>
3424     <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
3425 </xenc:EncryptedKey>
3426
3427 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
3428 Id="Encrypted_KEY_ID_2" Recipient="https://sp2.org">
3429     <xenc:EncryptionMethod
3430         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
3431     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3432 <ds:KeyName>KEY_NAME_2</ds:KeyName>
3433     </ds:KeyInfo>
3434     <xenc:CipherData>
3435 <xenc:CipherValue>abcXYZ...</xenc:CipherValue>
3436 </xenc:CipherData>
3437     <xenc:ReferenceList>
3438         <xenc:DataReference URI="#Encrypted_DATA_ID"/>
3439     </xenc:ReferenceList>
3440     <xenc:CarriedKeyName>MULTICAST_KEY_NAME</xenc:CarriedKeyName>
3441 </xenc:EncryptedKey>
3442 </saml:EncryptedAssertion>

```

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 [\[E61\]element constructs](#) are intended specifically for use as extension points in an extension schema; their types [listed](#) are set to `abstract`, and are thus usable only as the base of a derived type:

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

3480 The following constructs that are directly usable as part of SAML are particularly interesting targets for
3481 extension:

- 3482 • `<AuthnQuery>` and **AuthnQueryType**
- 3483 • `<AuthzDecisionQuery>` and **AuthzDecisionQueryType**
- 3484 • `<AttributeQuery>` and **AttributeQueryType**
- 3485 • **StatusResponseType**

3486 7.2 Schema Wildcard Extension Points

3487 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes
3488 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension
3489 schema.

3490 7.2.1 Assertion Extension Points

3491 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 3492 • `<SubjectConfirmationData>`: Uses **xs:anyType**, which allows any sub-elements and
3493 attributes.
- 3494 • `<AuthnContextDecl>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3495 • `<AttributeValue>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3496 • `<Advice>` and **AdviceType**: In addition to SAML-native elements, allows elements from other
3497 namespaces with lax schema validation processing.

3498 The following constructs in the assertion schema allow arbitrary global attributes:

- 3499 • `<Attribute>` and **AttributeType**

3500 7.2.2 Protocol Extension Points

3501 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 3502 • `<Extensions>` and **ExtensionsType**: Allows elements from other namespaces with lax schema
3503 validation processing.
- 3504 • `<StatusDetail>` and **StatusDetailType**: Allows elements from other namespaces with lax
3505 schema validation processing.
- 3506 • `<ArtifactResponse>` and **ArtifactResponseType**: Allows elements from any namespaces with
3507 lax schema validation processing. (It is specifically intended to carry a SAML request or response
3508 message element, however.)

3509 7.3 Identifier Extension

3510 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier
3511 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.
3512 However, it is always possible to define additional URI-based identifiers for these purposes. It is
3513 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should
3514 the meaning of a given URI used as such an identifier significantly change, or be used to mean two
3515 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.

3552 A SAML authority MUST NOT authorize both an action and its negated form.

3553 8.1.3 Get/Head/Put/Post

3554 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3555 Defined actions:

3556 GET HEAD PUT POST

3557 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
3558 the GET action on a resource is authorized to retrieve it.

3559 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
3560 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
3561 operation may cause data to be modified and a POST operation may cause modification to a resource
3562 other than the one specified in the request. For this reason a separate Action URI reference specifier is
3563 provided.

3564 8.1.4 UNIX File Permissions

3565 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3566 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3567 The action string is a four-digit numeric code:

3568 *extended user group world*

3569 Where the *extended* access permission has the value

3570 +2 if sgid is set

3571 +4 if suid is set

3572 The *user group* and *world* access permissions have the value

3573 +1 if execute permission is granted

3574 +2 if write permission is granted

3575 +4 if read permission is granted

3576 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read
3577 and execute; and world read.

3578 8.2 Attribute Name Format Identifiers

3579 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType**
3580 complex type to refer to the classification of the attribute name for purposes of interpreting the name.

3581 8.2.1 Unspecified

3582 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3583 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

3617 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3618 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"
3619 separator MAY be omitted.

3620 8.3.5 Kerberos Principal Name

3621 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3622 Indicates that the content of the element is in the form of a Kerberos principal name using the format
3623 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and
3624 realm are described in IETF RFC 1510 [RFC 1510].

3625 8.3.6 Entity Identifier

3626 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3627 Indicates that the content of the element is the identifier of an entity that provides SAML-based services
3628 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service
3629 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to
3630 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make
3631 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be
3632 used in other elements and attributes whose purpose is to identify a system entity in various protocol
3633 exchanges.

3634 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is
3635 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3636 The NameQualifier, SPNameQualifier, and SPProvidedID attributes MUST be omitted.

3637 8.3.7 Persistent Identifier

3638 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3639 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to
3640 an identity provider and a service provider or affiliation of service providers. [E86]Persistent name
3641 identifiers generated by identity providers MUST be constructed using pseudo-random values that have
3642 no discernible correspondence with the subject's actual identifier (for example, username). Persistent
3643 name identifiers generated by identity providers MUST be constructed using values that have no
3644 discernible correspondence with the subject's actual identity (for example, username). They MAY be
3645 pseudo-random values, or generated in any other manner, provided there is no guessable relationship
3646 between the value and the subject's underlying identity, and that they are unique within the range of
3647 values generated by a given identity provider for a given service provider or affiliation of providers. The
3648 intent is to create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or
3649 activities. Persistent name identifier values MUST NOT exceed a length of 256 characters. [E78]A given
3650 value, once associated with a principal, MUST NOT be assigned to a different principal at any time in the
3651 future.

3652 The element's NameQualifier attribute, if present, MUST contain the unique identifier of the identity
3653 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived
3654 from the context of the message containing the element, such as the issuer of a protocol message or an
3655 assertion containing the identifier in its subject. Note that a different system entity might later issue its own
3656 protocol message or assertion containing the identifier; the NameQualifier attribute does not change in
3657 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be
3658 omitted in such a case).

3659 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service
3660 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be
3661 omitted if the element is contained in a message intended only for consumption directly by the service
3662 provider, and the value would be the unique identifier of that service provider.

3663 ~~[E55]The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most~~
3664 ~~recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been~~
3665 ~~established, then the attribute MUST be omitted.~~

3666 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be
3667 shared in clear text with providers other than the providers that have established the shared identifier.
3668 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and
3669 protections. Deployments without such requirements are free to use other kinds of identifiers in their
3670 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

3671 Note also that while persistent identifiers are typically used to reflect an account linking relationship
3672 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3673 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly
3674 different and does not affect the behavior of the identity provider or any processing rules specific to
3675 persistent identifiers in the protocols defined in this specification.

3676 Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of
3677 creation, but not of use. If a persistent identifier is created by a particular identity provider, the
3678 `NameQualifier` attribute value is permanently established at that time. If a service provider that
3679 receives such an identifier takes on the role of an identity provider and issues its own assertion containing
3680 that identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted).
3681 It might alternatively choose to create its own persistent identifier to represent the principal and link the
3682 two values. This is a deployment decision.

3683 8.3.8 Transient Identifier

3684 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3685 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3686 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in
3687 accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of
3688 256 characters.

3689 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3690 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in
3691 accordance with the rules specified in Section 8.3.7.

3692 8.4 Consent Identifiers

3693 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType**
3694 and **StatusResponseType** complex types to communicate whether a principal gave consent, and under
3695 what conditions, for the message.

3696 8.4.1 Unspecified

3697 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

3698 No claim as to principal consent is being made.

3699 **8.4.2 Obtained**

3700 **URI:** urn:oasis:names:tc:SAML:2.0:consent:obtained

3701 Indicates that a principal's consent has been obtained by the issuer of the message.

3702 **8.4.3 Prior**

3703 **URI:** urn:oasis:names:tc:SAML:2.0:consent:prior

3704 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to
3705 the action that initiated the message.

3706 **8.4.4 Implicit**

3707 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3708 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the
3709 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically
3710 more proximal to the action in time and presentation than prior consent, such as part of a session of
3711 activities.

3712 **8.4.5 Explicit**

3713 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3714 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the
3715 action that initiated the message.

3716 **8.4.6 Unavailable**

3717 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3718 Indicates that the issuer of the message did not obtain consent.

3719 **8.4.7 Inapplicable**

3720 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3721 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*. [\[E74\]Second Edition](#). World Wide Web Consortium, [February 2002](#)[June 2008](#). 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>.

3766	[E74][RFC 3075]	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. See http://www.ietf.org/rfc/rfc3075.txt .
3767		
3768	[RFC 3513]	R. Hinden, S. Deering. <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. See http://www.ietf.org/rfc/rfc3513.txt .
3769		
3770	[SAMLAuthnCxt]	J. Kemp et al. <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See http://www.oasis-open.org/committees/security/ .
3771		
3772		
3773	[SAMLBind]	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/ .
3774		
3775		
3776	[SAMLConform]	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. http://www.oasis-open.org/committees/security/ .
3777		
3778		
3779	[SAMLGloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/ .
3780		
3781		
3782	[SAMLMeta]	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/ .
3783		
3784		
3785	[SAMLXSD]	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/ .
3786		
3787		
3788	[SAMLProf]	S. Cantor et al. <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See http://www.oasis-open.org/committees/security/ .
3789		
3790		
3791	[SAMLSecure]	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/ .
3792		
3793		
3794		
3795	[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/ .
3796		
3797		
3798	[SAMLXSD]	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/ .
3799		
3800		
3801	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
3802		
3803	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. See http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
3804		
3805	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. See http://www.w3.org/TR/WD-charreq .
3806		
3807	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. See http://www.w3.org/TR/charmod-norm/ .
3808		
3809		
3810	[XACML]	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See http://www.oasis-open.org/committees/xacml .
3811		
3812	[XML-ID]	J. Marsh et al. <i>xml:id Version 1.0</i> , World Wide Web Consortium, April 2004. See http://www.w3.org/TR/xml-id/ .
3813		

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 Kiestler, 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

- 3856 • Eve Maler, Sun Microsystems
- 3857 • Ronald Monzillo, Sun Microsystems
- 3858 • Emily Xu, Sun Microsystems
- 3859 • Greg Whitehead, Trustgenix

3860

3861 The editors also would like to acknowledge the following former SSTC members for their contributions to
3862 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 3863 • Stephen Farrell, Baltimore Technologies
- 3864 • David Orchard, BEA Systems
- 3865 • Krishna Sankar, Cisco Systems
- 3866 • Zahid Ahmed, CommerceOne
- 3867 • Tim Alsop, CyberSafe Limited
- 3868 • Carlisle Adams, Entrust
- 3869 • Tim Moses, Entrust
- 3870 • Nigel Edwards, Hewlett-Packard
- 3871 • Joe Pato, Hewlett-Packard
- 3872 • Bob Blakley, IBM
- 3873 • Marlena Erdos, IBM
- 3874 • Marc Chanliau, Netegrity
- 3875 • Chris McLaren, Netegrity
- 3876 • Lynne Rosenthal, NIST
- 3877 • Mark Skall, NIST
- 3878 • Charles Knouse, Oblix
- 3879 • Simon Godik, Overxeer
- 3880 • Charles Norwood, SAIC
- 3881 • Evan Prodromou, Securant
- 3882 • Robert Griffin, RSA Security (former editor)
- 3883 • Sai Allarvarpu, Sun Microsystems
- 3884 • Gary Ellison, Sun Microsystems
- 3885 • Chris Ferris, Sun Microsystems
- 3886 • Mike Myers, Traceroute Security
- 3887 • Phillip Hallam-Baker, VeriSign (former editor)
- 3888 • James Vanderbeek, Vodafone
- 3889 • Mark O'Neill, Vordel
- 3890 • Tony Palmer, Vordel

3891

3892 Finally, the editors wish to acknowledge the following people for their contributions of material used as
3893 input to the OASIS Security Assertions Markup Language specifications:

- 3894 • Thomas Gross, IBM
- 3895 • Birgit Pfitzmann, IBM

3896 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on
3897 the SSTC was the primary editor of the errata working document and who made major substantive
3898 contributions to all of the errata materials.

Appendix B. Notices

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

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

Copyright © OASIS Open 2005. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

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