



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

Committee Draft 03, 14 December 2004

**Document identifier:**

sstc-saml-core-2.0-cd-03

**Location:**

[http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

**Editors:**

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

**SAML V2.0 Contributors:**

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rebekah Metz, Booz Allen Hamilton  
Rick Randall, Booz, Allen, Hamilton  
Tim Alsop, CyberSafe Limited  
Paul Madsen, 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  
Charles Knouse, Oblix  
Steve Anderson, OpenNetwork  
Prateek Mishra, Principal Identity  
John Linn, RSA Security  
Rob Philpott, RSA Security  
Jahan Moreh, Sigaba

42 Anne Anderson, Sun Microsystems  
43 Gary Ellison, Sun Microsystems  
44 Eve Maler, Sun Microsystems  
45 Ron Monzillo, Sun Microsystems  
46 Greg Whitehead, Trustgenix

47 **Abstract:**

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

50 **Status:**

51 This is a **Committee Draft** approved by the Security Services Technical Committee on 14  
52 December 2004.

53 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)  
54 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located  
55 at [http://www.oasis-open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security). The  
56 committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog  
57 of any changes made to this document as a result of comments.

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

# Table of Contents

63	1 Introduction.....	7
64	1.1 Notation.....	7
65	1.2 Schema Organization and Namespaces.....	8
66	1.3 Common Data Types.....	8
67	1.3.1 String Values.....	8
68	1.3.2 URI Values.....	9
69	1.3.3 Time Values.....	9
70	1.3.4 ID and ID Reference Values.....	9
71	2 SAML Assertions.....	11
72	2.1 Schema Header and Namespace Declarations.....	11
73	2.2 Name Identifiers.....	12
74	2.2.1 Element <BaseID>.....	12
75	2.2.2 Complex Type NameIDType.....	13
76	2.2.3 Element <NameID>.....	14
77	2.2.4 Element <EncryptedID>.....	14
78	2.2.5 Element <Issuer>.....	14
79	2.3 Assertions.....	15
80	2.3.1 Element <AssertionIDRef>.....	15
81	2.3.2 Element <AssertionURIRef>.....	15
82	2.3.3 Element <Assertion>.....	15
83	2.3.4 Element <EncryptedAssertion>.....	17
84	2.4 Subjects.....	17
85	2.4.1 Element <Subject>.....	17
86	2.4.1.1 Element <SubjectConfirmation>.....	18
87	2.4.1.2 Element <SubjectConfirmationData>.....	19
88	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	20
89	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
90	2.5 Conditions.....	21
91	2.5.1 Element <Conditions>.....	21
92	2.5.1.1 General Processing Rules.....	22
93	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
94	2.5.1.3 Element <Condition>.....	23
95	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
96	2.5.1.5 Element <OneTimeUse>.....	24
97	2.5.1.6 Element <ProxyRestriction>.....	25
98	2.6 Advice.....	25
99	2.6.1 Element <Advice>.....	26
100	2.7 Statements.....	26
101	2.7.1 Element <Statement>.....	26
102	2.7.2 Element <AuthnStatement>.....	26
103	2.7.2.1 Element <SubjectLocality>.....	28
104	2.7.2.2 Element <AuthnContext>.....	28
105	2.7.3 Element <AttributeStatement>.....	29
106	2.7.3.1 Element <Attribute>.....	29
107	2.7.3.1.1 Element <AttributeValue>.....	30
108	2.7.3.2 Element <EncryptedAttribute>.....	31
109	2.7.4 Element <AuthzDecisionStatement>.....	31
110	2.7.4.1 Simple Type DecisionType.....	33
111	2.7.4.2 Element <Action>.....	33

112	2.7.4.3 Element <Evidence>.....	34
113	3 SAML Protocols.....	35
114	3.1 Schema Header and Namespace Declarations.....	35
115	3.2 Requests and Responses.....	36
116	3.2.1 Complex Type RequestAbstractType.....	36
117	3.2.2 Complex Type StatusResponseType.....	38
118	3.2.2.1 Element <Status>.....	39
119	3.2.2.2 Element <StatusCode>.....	40
120	3.2.2.3 Element <StatusMessage>.....	42
121	3.2.2.4 Element <StatusDetail>.....	42
122	3.3 Assertion Query and Request Protocol.....	42
123	3.3.1 Element <AssertionIDRequest>.....	42
124	3.3.2 Queries.....	42
125	3.3.2.1 Element <SubjectQuery>.....	42
126	3.3.2.2 Element <AuthnQuery>.....	43
127	3.3.2.3 Element <RequestedAuthnContext>.....	44
128	3.3.2.4 Element <AttributeQuery>.....	45
129	3.3.2.5 Element <AuthzDecisionQuery>.....	46
130	3.3.3 Element <Response>.....	46
131	3.3.4 Processing Rules.....	47
132	3.4 Authentication Request Protocol.....	47
133	3.4.1 Element <AuthnRequest>.....	48
134	3.4.1.1 Element <NameIDPolicy>.....	50
135	3.4.1.2 Element <Scoping>.....	51
136	3.4.1.3 Element <IDPList>.....	52
137	3.4.1.3.1 Element <IDPEntry>.....	52
138	3.4.1.4 Processing Rules.....	53
139	3.4.1.5 Proxying.....	53
140	3.4.1.5.1 Proxying Processing Rules.....	54
141	3.5 Artifact Resolution Protocol.....	55
142	3.5.1 Element <ArtifactResolve>.....	55
143	3.5.2 Element <ArtifactResponse>.....	56
144	3.5.3 Processing Rules.....	56
145	3.6 Name Identifier Management Protocol.....	57
146	3.6.1 Element <ManageNameIDRequest>.....	57
147	3.6.2 Element <ManageNameIDResponse>.....	58
148	3.6.3 Processing Rules.....	58
149	3.7 Single Logout Protocol.....	59
150	3.7.1 Element <LogoutRequest>.....	59
151	3.7.2 Element <LogoutResponse>.....	60
152	3.7.3 Processing Rules.....	60
153	3.7.3.1 Session Participant Rules.....	61
154	3.7.3.2 Session Authority Rules.....	61
155	3.8 Name Identifier Mapping Protocol.....	62
156	3.8.1 Element <NameIDMappingRequest>.....	63
157	3.8.2 Element <NameIDMappingResponse>.....	63
158	3.8.3 Processing Rules.....	64
159	4 SAML Versioning.....	65
160	4.1 SAML Specification Set Version.....	65
161	4.1.1 Schema Version.....	65
162	4.1.2 SAML Assertion Version.....	65
163	4.1.3 SAML Protocol Version.....	66
164	4.1.3.1 Request Version.....	66

165	4.1.3.2 Response Version.....	66
166	4.1.3.3 Permissible Version Combinations.....	67
167	4.2 SAML Namespace Version.....	67
168	4.2.1 Schema Evolution.....	67
169	5 SAML and XML Signature Syntax and Processing.....	68
170	5.1 Signing Assertions.....	68
171	5.2 Request/Response Signing.....	68
172	5.3 Signature Inheritance.....	68
173	5.4 XML Signature Profile.....	69
174	5.4.1 Signing Formats and Algorithms.....	69
175	5.4.2 References.....	69
176	5.4.3 Canonicalization Method.....	69
177	5.4.4 Transforms.....	69
178	5.4.5 KeyInfo.....	70
179	5.4.6 Example.....	70
180	6 SAML and XML Encryption Syntax and Processing.....	73
181	6.1 General Considerations.....	73
182	6.2 Combining Signatures and Encyption.....	73
183	7 SAML Extensibility.....	74
184	7.1 Schema Extension.....	74
185	7.1.1 Assertion Schema Extension.....	74
186	7.1.2 Protocol Schema Extension.....	74
187	7.2 Schema Wildcard Extension Points.....	75
188	7.2.1 Assertion Extension Points.....	75
189	7.2.2 Protocol Extension Points.....	75
190	7.3 Identifier Extension.....	75
191	8 SAML-Defined Identifiers.....	77
192	8.1 Action Namespace Identifiers.....	77
193	8.1.1 Read/Write/Execute/Delete/Control.....	77
194	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	77
195	8.1.3 Get/Head/Put/Post.....	78
196	8.1.4 UNIX File Permissions.....	78
197	8.2 Attribute Name Format Identifiers.....	78
198	8.2.1 Unspecified.....	78
199	8.2.2 URI Reference.....	79
200	8.2.3 Basic.....	79
201	8.3 Name Identifier Format Identifiers.....	79
202	8.3.1 Unspecified.....	79
203	8.3.2 Email Address.....	79
204	8.3.3 X.509 Subject Name.....	79
205	8.3.4 Windows Domain Qualified Name.....	79
206	8.3.5 Kerberos Principal Name.....	80
207	8.3.6 Entity Identifier.....	80
208	8.3.7 Persistent Identifier.....	80
209	8.3.8 Transient Identifier.....	81
210	8.4 Consent Identifiers.....	81
211	8.4.1 Unspecified.....	81
212	8.4.2 Obtained.....	81
213	8.4.3 Prior.....	81
214	8.4.4 Implicit.....	81
215	8.4.5 Explicit.....	82

216	8.4.6 Unavailable.....	82
217	8.4.7 Inapplicable.....	82
218	9 References.....	83
219	9.1 Normative References.....	83
220	9.2 Non-Normative References.....	83
221	Appendix A. Acknowledgments.....	86
222	Appendix B. Notices.....	88
223		

# 1 Introduction

224

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

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

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

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

## 1.1 Notation

241

242 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD  
243 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as  
244 described in IETF RFC 2119 [RFC 2119].

245 `Listings of SAML schemas appear like this.`

246

247 `Example code listings appear like this.`

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

249 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative  
250 text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In  
251 cases of disagreement between the SAML schema documents and schema listings in this specification,  
252 the schema documents take precedence. Note that in some cases the normative text of this specification  
253 imposes constraints beyond those indicated by the schema documents.

254 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for  
255 their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is  
256 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 . The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].

Prefix	XML Namespace	Comments
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

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

## 259 1.2 Schema Organization and Namespaces

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

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

263 The SAML request-response protocol structures are defined in a schema associated with the following  
264 XML namespace:

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

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

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

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

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

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

## 274 1.3 Common Data Types

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

### 277 1.3.1 String Values

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

282 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that  
283 have the XML Schema **xs:string** type, or a type derived from that, **MUST** be compared using an exact  
284 binary comparison. In particular, SAML implementations and deployments **MUST NOT** depend on case-  
285 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific



286 formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft  
287 Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

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

293 Applications that compare data received in SAML documents to data from external sources MUST take  
294 into account the normalization rules specified for XML. Text contained within elements is normalized so  
295 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the XML  
296 Recommendation [XML] §2.11. XML attribute values defined as strings (or types derived from strings) are  
297 normalized as described in [XML] §3.3.3. All whitespace characters are replaced with blanks (ASCII code  
298 32<sub>Decimal</sub>).

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

### 302 1.3.2 URI Values

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

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

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

### 312 1.3.3 Time Values

313 All SAML time values have the type **xs:dateTime**, which is built in to the W3C XML Schema Datatypes  
314 specification [Schema2], and MUST be expressed in UTC form, with no time zone component.

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

### 317 1.3.4 ID and ID Reference Values

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

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

325 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the  
326 implementation. In the case that a random or pseudorandom technique is employed, the probability of two  
327 randomly chosen identifiers being identical MUST be less than or equal to  $2^{-128}$  and SHOULD be less than  
328 or equal to  $2^{-160}$ . This requirement MAY be met by encoding a randomly chosen value between 128 and

329 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom  
330 generator **MUST** be seeded with unique material in order to ensure the desired uniqueness properties  
331 between different systems.

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

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

---

## 2 SAML Assertions

342

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

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

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

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

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

364 As described in Section 7, extensions are permitted by the SAML assertion schema, allowing user-defined  
365 extensions to assertions and statements, as well as allowing the definition of new kinds of assertions and  
366 statements.

367 The SAML technical overview [SAML-TechOvw] and glossary [SAMLGloss] provide more detailed  
368 explanation of SAML terms and concepts.

### 2.1 Schema Header and Namespace Declarations

369

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

```
372 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
373   xmlns="http://www.w3.org/2001/XMLSchema"  
374   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
375   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
376   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
377   elementFormDefault="unqualified"  
378   attributeFormDefault="unqualified"  
379   blockDefault="substitution"  
380   version="2.0">  
381   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
382     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-  
383 20020212/xmldsig-core-schema.xsd"/>  
384   <import namespace="http://www.w3.org/2001/04/xmlenc#"  
385     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-  
386 20021210/xenc-schema.xsd"/>  
387   <annotation>  
388     <documentation>  
389       Document identifier: sstc-saml-schema-assertion-2.0
```

```

390         Location: http://www.oasis-
391 open.org/committees/documents.php?wg_abbrev=security
392         Revision history:
393         V1.0 (November, 2002):
394             Initial Standard Schema.
395         V1.1 (September, 2003):
396             Updates within the same V1.0 namespace.
397         V2.0 CD-03 (December, 2004):
398             New assertion schema based in a SAML V2.0 namespace.
399     </documentation>
400 </annotation>
401 ...
402 </schema>

```

## 403 2.2 Name Identifiers

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

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

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

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

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

### 423 2.2.1 Element <BaseID>

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

427 **NameQualifier** [Optional]

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

430 **SPNameQualifier** [Optional]

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

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

```

436 <attributeGroup name="IDNameQualifiers">
437   <attribute name="NameQualifier" type="string" use="optional"/>
438   <attribute name="SPNameQualifier" type="string" use="optional"/>
439 </attributeGroup>
440 <element name="BaseID" type="saml:BaseIDAbstractType"/>
441 <complexType name="BaseIDAbstractType" abstract="true">
442   <attributeGroup ref="saml:IDNameQualifiers"/>
443 </complexType>

```

## 444 2.2.2 Complex Type NameIDType

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

### 449 NameQualifier [Optional]

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

### 452 SPNameQualifier [Optional]

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

### 456 Format [Optional]

457         A URI reference representing the classification of string-based identifier information. See Section  
458         8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute  
459         and their associated descriptions and processing rules. Unless otherwise specified by an element  
460         based on this type, if no `Format` value is provided, then the value  
461         `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in  
462         effect.

463         When a `Format` value other than one specified in Section 8.3 is used, the content of an element  
464         of this type is to be interpreted according to the definition of that format as provided outside of this  
465         specification. If not otherwise indicated by the definition of the format, issues of anonymity,  
466         pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties  
467         are implementation-specific.

### 468 SPProvidedID [Optional]

469         A name identifier established by a service provider or affiliation of providers for the entity, if  
470         different from the primary name identifier given in the content of the element. This attribute  
471         provides a means of integrating the use of SAML with existing identifiers already in use by a  
472         service provider. For example, an existing identifier can be "attached" to the entity using the Name  
473         Identifier Management protocol defined in Section 3.6.

474 Additional rules for the content of (or the omission of) these attributes can be defined by elements that  
475 make use of this type, and by specific `Format` definitions.

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

```

477 <complexType name="NameIDType">
478   <simpleContent>
479     <extension base="string">
480       <attributeGroup ref="saml:IDNameQualifiers"/>
481       <attribute name="Format" type="anyURI" use="optional"/>
482       <attribute name="SPProvidedID" type="string" use="optional"/>

```

```
483     </extension>
484     </simpleContent>
485 </complexType>
```

### 486 2.2.3 Element <NameID>

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

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

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

### 492 2.2.4 Element <EncryptedID>

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

496 <xenc:EncryptedData> [Required]

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

502 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

```
515 <complexType name="EncryptedElementType">
516   <sequence>
517     <element ref="xenc:EncryptedData"/>
518     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
519   </sequence>
520 </complexType>
521 <element name="EncryptedID" type="saml:EncryptedElementType"/>
```

### 522 2.2.5 Element <Issuer>

523 The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a  
524 SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name,  
525 but permits various pieces of descriptive data (see Section 2.2.2).

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

528 The following schema fragment defines the `<Issuer>` element:

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

## 530 2.3 Assertions

531 The following sections define the SAML constructs that either contain assertion information or provide a  
532 means to refer to an existing assertion.

### 533 2.3.1 Element `<AssertionIDRef>`

534 The `<AssertionIDRef>` element makes a reference to a SAML assertion by its unique identifier. The  
535 specific authority who issued the assertion or from whom the assertion can be obtained is not specified as  
536 part of the reference. See Section 3.3.1 for a protocol element that uses such a reference to ask for the  
537 corresponding assertion.

538 The following schema fragment defines the `<AssertionIDRef>` element:

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

### 540 2.3.2 Element `<AssertionURIRef>`

541 The `<AssertionURIRef>` element makes a reference to a SAML assertion by URI reference. The URI  
542 reference MAY be used to retrieve the corresponding assertion in a manner specific to the URI reference.  
543 See Section 3.7 of the Bindings specification [SAMLBind] for information on how this element is used in a  
544 protocol binding to accomplish this.

545 The following schema fragment defines the `<AssertionURIRef>` element:

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

### 547 2.3.3 Element `<Assertion>`

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

550 `Version` [Required]

551 The version of this assertion. The identifier for the version of SAML defined in this specification is  
552 "2.0". SAML versioning is discussed in Section 4.

553 `ID` [Required]

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

556 `IssueInstant` [Required]

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

558 `<Issuer>` [Required]

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

561 This specification defines no particular relationship between the entity represented by this element  
562 and the signer of the assertion (if any). Any such requirements imposed by a relying party that

563 consumes the assertion or by specific profiles are application-specific.

564 `<ds:Signature>` [Optional]

565 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as  
566 described below and in section 5.

567 `<Subject>` [Optional]

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

569 `<Conditions>` [Optional]

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

572 `<Advice>` [Optional]

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

575 Zero or more of the following statement elements:

576 `<Statement>`

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

579 `<AuthnStatement>`

580 An authentication statement.

581 `<AuthzDecisionStatement>`

582 An authorization decision statement.

583 `<AttributeStatement>`

584 An attribute statement.

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

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

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

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

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



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

```
608 <element name="Assertion" type="saml:AssertionType"/>
609 <complexType name="AssertionType">
610   <sequence>
611     <element ref="saml:Issuer"/>
612     <element ref="ds:Signature" minOccurs="0"/>
613     <element ref="saml:Subject" minOccurs="0"/>
614     <element ref="saml:Conditions" minOccurs="0"/>
615     <element ref="saml:Advice" minOccurs="0"/>
616     <choice minOccurs="0" maxOccurs="unbounded">
617       <element ref="saml:Statement"/>
618       <element ref="saml:AuthnStatement"/>
619       <element ref="saml:AuthzDecisionStatement"/>
620       <element ref="saml:AttributeStatement"/>
621     </choice>
622   </sequence>
623   <attribute name="Version" type="string" use="required"/>
624   <attribute name="ID" type="ID" use="required"/>
625   <attribute name="IssueInstant" type="dateTime" use="required"/>
626 </complexType>
```

## 627 2.3.4 Element <EncryptedAssertion>

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

631 <xenc:EncryptedData> [Required]

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

636 <xenc:EncryptedKey> [Zero or More]

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

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

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

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

## 645 2.4 Subjects

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

### 647 2.4.1 Element <Subject>

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

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

652 Identifies the subject.

653 <SubjectConfirmation> [Zero or More]

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

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

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

664 A <Subject> element SHOULD NOT identify more than one principal.

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

```
666 <element name="Subject" type="saml:SubjectType"/>
667 <complexType name="SubjectType">
668   <choice>
669     <sequence>
670       <choice>
671         <element ref="saml:BaseID"/>
672         <element ref="saml:NameID"/>
673         <element ref="saml:EncryptedID"/>
674       </choice>
675       <element ref="saml:SubjectConfirmation" minOccurs="0"
676 maxOccurs="unbounded"/>
677     </sequence>
678     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
679   </choice>
680 </complexType>
```

#### 681 **2.4.1.1 Element <SubjectConfirmation>**

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

685 Method [Required]

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

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

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

692 <SubjectConfirmationData> [Optional]

693 Additional confirmation information to be used by a specific confirmation method. For example, typical  
694 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax  
695 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section  
696 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,  
697 attributes, or content that may appear in the <SubjectConfirmationData> element.

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

```
700 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
701 <complexType name="SubjectConfirmationType">
702   <sequence>
703     <choice minOccurs="0">
704       <element ref="saml:BaseID"/>
705       <element ref="saml:NameID"/>
706       <element ref="saml:EncryptedID"/>
707     </choice>
708     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
709   </sequence>
710   <attribute name="Method" type="anyURI" use="required"/>
711 </complexType>
```

### 712 2.4.1.2 Element `<SubjectConfirmationData>`

713 The `<SubjectConfirmationData>` element has the **SubjectConfirmationDataType** complex type. It  
714 specifies additional data that allows the subject to be confirmed or constrains the circumstances under  
715 which the act of subject confirmation can take place. Subject confirmation takes place when a relying  
716 party seeks to verify the relationship between an entity presenting the assertion (that is, the confirming  
717 entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply  
718 to any method:

719 `NotBefore` [Optional]

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

722 `NotOnOrAfter` [Optional]

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

725 `Recipient` [Optional]

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

729 `InResponseTo` [Optional]

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

733 `Address` [Optional]

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

740 Arbitrary attributes

741 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary namespace-  
742 qualified XML attributes to be added to `<SubjectConfirmationData>` constructs without the need  
743 for an explicit schema extension. This allows additional fields to be added as needed to supply  
744 additional confirmation-related information. SAML extensions MUST NOT add local (non-namespace-  
745 qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the

746 **SubjectConfirmationDataType** complex type or a derivation of it; such attributes are reserved for  
747 future maintenance and enhancement of SAML itself.

748 Arbitrary elements

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

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

756 Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present,  
757 SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's  
758 `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore`  
759 MUST be less than (earlier than) the value for `NotOnOrAfter`.

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

```
762 <element name="SubjectConfirmationData"  
763 type="saml:SubjectConfirmationDataType"/>  
764 <complexType name="SubjectConfirmationDataType" mixed="true">  
765 <complexContent>  
766 <restriction base="anyType">  
767 <sequence>  
768 <any namespace="##any" processContents="lax" minOccurs="0"  
769 maxOccurs="unbounded"/>  
770 </sequence>  
771 <attribute name="NotBefore" type="dateTime" use="optional"/>  
772 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
773 <attribute name="Recipient" type="anyURI" use="optional"/>  
774 <attribute name="InResponseTo" type="NCName" use="optional"/>  
775 <attribute name="Address" type="string" use="optional"/>  
776 <anyAttribute namespace="##other" processContents="lax"/>  
777 </restriction>  
778 </complexContent>  
779 </complexType>
```

### 780 2.4.1.3 Complex Type **KeyInfoConfirmationDataType**

781 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`  
782 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in  
783 some way to authenticate the principal identified by the subject. The particular confirmation method MUST  
784 define the exact mechanism by which the confirmation data can be used. The optional attributes defined  
785 by the **SubjectConfirmationDataType** complex type MAY also appear.

786 This complex type, or a type derived from it, SHOULD be used by any confirmation method that defines its  
787 confirmation data in terms of the `<ds:KeyInfo>` element.

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

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

```
792 <complexType name="KeyInfoConfirmationDataType" mixed="false">  
793 <complexContent>  
794 <restriction base="saml:SubjectConfirmationDataType">  
795 <sequence>
```

```
796         <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
797     </sequence>
798 </restriction>
799 </complexContent>
800 </complexType>
```

#### 801 2.4.1.4 Example of a Key-Confirmed <Subject>

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

```
806 <Subject>
807   <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
808     scott@example.org
809   </NameID>
810   <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
811     <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
812       <ds:KeyInfo>
813         <ds:KeyName>Scott's Key</ds:KeyName>
814       </ds:KeyInfo>
815     </SubjectConfirmationData>
816   </SubjectConfirmation>
817 </Subject>
```

## 818 2.5 Conditions

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

### 821 2.5.1 Element <Conditions>

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

823 NotBefore [Optional]

824 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as  
825 described in Section 1.3.3.

826 NotOnOrAfter [Optional]

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

829 <Condition> [Any Number]

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

832 <AudienceRestriction> [Any Number]

833 Specifies that the assertion is addressed to a particular audience.

834 <OneTimeUse> [Optional]

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

838 <ProxyRestriction> [Optional]

839 Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act

840 as asserting parties themselves and issue assertions of their own on the basis of the information  
841 contained in the original assertion. Although the schema permits multiple occurrences, there MUST  
842 be at most one instance of this element.

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

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

```
849 <element name="Conditions" type="saml:ConditionsType"/>
850 <complexType name="ConditionsType">
851   <choice minOccurs="0" maxOccurs="unbounded">
852     <element ref="saml:Condition"/>
853     <element ref="saml:AudienceRestriction"/>
854     <element ref="saml:OneTimeUse"/>
855     <element ref="saml:ProxyRestriction"/>
856   </choice>
857   <attribute name="NotBefore" type="dateTime" use="optional"/>
858   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
859 </complexType>
```

### 860 2.5.1.1 General Processing Rules

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

863 Note that an assertion that has condition validity status **Valid** may nonetheless be untrustworthy or invalid  
864 for reasons such as not being well-formed or schema-valid, not being issued by a trustworthy SAML  
865 authority, or not being authenticated by a trustworthy means.

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

- 868 1. If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is  
869 considered to be **Valid** with respect to condition processing.
- 870 2. If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the  
871 assertion is considered to be **Invalid**.
- 872 3. If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, or if an element is  
873 encountered that is not understood, then the validity of the assertion cannot be determined and is  
874 considered to be **Indeterminate**.
- 875 4. If all sub-elements and attributes of the `<Conditions>` element are determined to be **Valid**, then the  
876 assertion is considered to be **Valid** with respect to condition processing.

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

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



### 882 **2.5.1.2 Attributes NotBefore and NotOnOrAfter**

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

886 The `NotBefore` attribute specifies the time instant at which the validity interval begins. The  
887 `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

888 If the value for either `NotBefore` or `NotOnOrAfter` is omitted, then it is considered unspecified. If the  
889 `NotBefore` attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then  
890 the assertion is **Valid** with respect to conditions at any time before the time instant specified by the  
891 `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if all other conditions that are  
892 supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant specified  
893 by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other conditions that  
894 are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any time.

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

### 897 **2.5.1.3 Element <Condition>**

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

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

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

### 904 **2.5.1.4 Elements <AudienceRestriction> and <Audience>**

905 The `<AudienceRestriction>` element specifies that the assertion is addressed to one or more  
906 specific audiences identified by `<Audience>` elements. Although a SAML relying party that is outside the  
907 audiences specified is capable of drawing conclusions from an assertion, the SAML asserting party  
908 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the  
909 following element:

910 `<Audience>`

911 A URI reference that identifies an intended audience. The URI reference MAY identify a document  
912 that describes the terms and conditions of audience membership. It MAY also contain the unique  
913 identifier of a system entity that has a SAML name identifier (see Section 8.3.6).

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

916 The SAML asserting party cannot prevent a party to whom the assertion is disclosed from taking action on  
917 the basis of the information provided. However, the `<AudienceRestriction>` element allows the  
918 SAML asserting party to state explicitly that no warranty is provided to such a party in a machine- and  
919 human-readable form. While there can be no guarantee that a court would uphold such a warranty  
920 exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably  
921 improved.

922 Note that multiple `<AudienceRestriction>` elements MAY be included in a single assertion, and each  
923 MUST be evaluated independently. The effect of this requirement and the preceding definition is that

924 within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a  
925 conjunction (an "AND").

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

```
928 <element name="AudienceRestriction"  
929 type="saml:AudienceRestrictionType"/>  
930 <complexType name="AudienceRestrictionType">  
931 <complexContent>  
932 <extension base="saml:ConditionAbstractType">  
933 <sequence>  
934 <element ref="saml:Audience" maxOccurs="unbounded"/>  
935 </sequence>  
936 </extension>  
937 </complexContent>  
938 </complexType>  
939 <element name="Audience" type="anyURI"/>
```

### 940 **2.5.1.5 Element <OneTimeUse>**

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

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

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

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

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

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

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

```
966 <element name="OneTimeUse" type="saml:OneTimeUseType"/>  
967 <complexType name="OneTimeUseType">  
968 <complexContent>  
969 <extension base="saml:ConditionAbstractType"/>  
970 </complexContent>  
971 </complexType>
```



## 972 2.5.1.6 Element <ProxyRestriction>

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

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

978 Count [Optional]

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

981 <Audience> [Zero or More]

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

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

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

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

994 For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction>  
995 condition is considered to always be valid. That is, this condition does not affect validity but is a condition  
996 on use.

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

```
999 <element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
1000 <complexType name="ProxyRestrictionType">
1001   <complexContent>
1002     <extension base="saml:ConditionAbstractType">
1003       <sequence>
1004         <element ref="saml:Audience" minOccurs="0"
1005 maxOccurs="unbounded"/>
1006       </sequence>
1007       <attribute name="Count" type="nonNegativeInteger" use="optional"/>
1008     </extension>
1009   </complexContent>
1010 </complexType>
```

## 1011 2.6 Advice

1012 This section defines the SAML constructs that contain additional information about an assertion that an  
1013 asserting party wishes to provide to a relying party.

## 1014 2.6.1 Element <Advice>

1015 The <Advice> element contains any additional information that the SAML authority wishes to provide.  
1016 This information MAY be ignored by applications without affecting either the semantics or the validity of  
1017 the assertion.

1018 The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>,  
1019 <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in  
1020 other non-SAML namespaces.

1021 Following are some potential uses of the <Advice> element:

- 1022 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating  
1023 the claims) or indirectly (by reference to the supporting assertions).
- 1024 • State a proof of the assertion claims.
- 1025 • Specify the timing and distribution points for updates to the assertion.

1026 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
1027 <element name="Advice" type="saml:AdviceType"/>  
1028 <complexType name="AdviceType">  
1029   <choice minOccurs="0" maxOccurs="unbounded">  
1030     <element ref="saml:AssertionIDRef"/>  
1031     <element ref="saml:AssertionURIRef"/>  
1032     <element ref="saml:Assertion"/>  
1033     <element ref="saml:EncryptedAssertion"/>  
1034     <any namespace="##other" processContents="lax"/>  
1035   </choice>  
1036 </complexType>
```

## 1037 2.7 Statements

1038 The following sections define the SAML constructs that contain statement information.

### 1039 2.7.1 Element <Statement>

1040 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
1041 the SAML assertion framework. SAML itself derives its core statements from this extension point. Its  
1042 **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

1043 The following schema fragment defines the <Statement> element and its **StatementAbstractType**  
1044 complex type:

```
1045 <element name="Statement" type="saml:StatementAbstractType"/>  
1046 <complexType name="StatementAbstractType" abstract="true"/>
```

### 1047 2.7.2 Element <AuthnStatement>

1048 The <AuthnStatement> element describes a statement by the SAML authority asserting that the  
1049 assertion subject was authenticated by a particular means at a particular time. Assertions containing  
1050 <AuthnStatement> elements MUST contain a <Subject> element.

1051 It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the  
1052 following elements and attributes:

1053 **Note:** The <AuthorityBinding> element and its corresponding type were removed  
1054 from <AuthnStatement> for V2.0 of SAML.

- 1055 **AuthnInstant** [Required]
- 1056        Specifies the time at which the authentication took place. The time value is encoded in UTC, as  
1057        described in Section 1.3.3.
- 1058 **SessionIndex** [Optional]
- 1059        Specifies the index of a particular session between the principal identified by the subject and the  
1060        authenticating authority.
- 1061 **SessionNotOnOrAfter** [Optional]
- 1062        Specifies a time instant at which the session between the principal identified by the subject and the  
1063        SAML authority issuing this statement **MUST** be considered ended. The time value is encoded in  
1064        UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a  
1065        **NotOnOrAfter** condition attribute that may be present in the assertion.
- 1066 **<SubjectLocality>** [Optional]
- 1067        Specifies the DNS domain name and IP address for the system from which the assertion subject was  
1068        apparently authenticated.
- 1069 **<AuthnContext>** [Required]
- 1070        The context used by the authenticating authority up to and including the authentication event that  
1071        yielded this statement. Contains an authentication context class reference, an authentication context  
1072        declaration or declaration reference, or both. See the Authentication Context specification  
1073        [SAMLAuthnCxt] for a full description of authentication context information.
- 1074 In general, any string value **MAY** be used as a **SessionIndex** value. However, when privacy is a  
1075 consideration, care must be taken to ensure that the **SessionIndex** value does not invalidate other  
1076 privacy mechanisms. Accordingly, the value **SHOULD NOT** be usable to correlate activity by a principal  
1077 across different session participants. Two solutions that achieve this goal are provided below and are  
1078 **RECOMMENDED**:
- 1079 • Use small positive integers (or reoccurring constants in a list) for the **SessionIndex**. The SAML  
1080 authority **SHOULD** choose the range of values such that the cardinality of any one integer will be  
1081 sufficiently high to prevent a particular principal's actions from being correlated across multiple session  
1082 participants. The SAML authority **SHOULD** choose values for **SessionIndex** randomly from within  
1083 this range (except when required to ensure unique values for subsequent statements given to the  
1084 same session participant but as part of a distinct session).
  - 1085 • Use the enclosing assertion's **ID** value in the **SessionIndex**.

1086 The following schema fragment defines the **<AuthnStatement>** element and its **AuthnStatementType**  
1087 complex type:

```

1088 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1089 <complexType name="AuthnStatementType">
1090   <complexContent>
1091     <extension base="saml:StatementAbstractType">
1092       <sequence>
1093         <element ref="saml:SubjectLocality" minOccurs="0"/>
1094         <element ref="saml:AuthnContext"/>
1095       </sequence>
1096       <attribute name="AuthnInstant" type="dateTime" use="required"/>
1097       <attribute name="SessionIndex" type="string" use="optional"/>
1098       <attribute name="SessionNotOnOrAfter" type="dateTime"
1099 use="optional"/>
1100     </extension>
1101   </complexContent>
1102 </complexType>

```

### 1103 **2.7.2.1 Element <SubjectLocality>**

1104 The <SubjectLocality> element specifies the DNS domain name and IP address for the system from  
1105 which the assertion subject was authenticated. It has the following attributes:

1106 Address [Optional]

1107 The network address of the system from which the principal identified by the subject was  
1108 authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4").  
1109 IPv6 addresses SHOULD be represented as defined by section 2.2 of [RFC 3513] (e.g.,  
1110 "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

1111 DNSName [Optional]

1112 The DNS name of the system from which the principal identified by the subject was authenticated.

1113 This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful  
1114 information in some applications.

1115 The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**  
1116 complex type:

```
1117 <element name="SubjectLocality"  
1118         type="saml: SubjectLocalityType"/>  
1119 <complexType name="SubjectLocalityType">  
1120   <attribute name="Address" type="string" use="optional"/>  
1121   <attribute name="DNSName" type="string" use="optional"/>  
1122 </complexType>
```

### 1123 **2.7.2.2 Element <AuthnContext>**

1124 The <AuthnContext> element specifies the context of an authentication event. The element can contain  
1125 an authentication context class reference, an authentication context declaration or declaration reference,  
1126 or both. Its complex **AuthnContextType** has the following elements:

1127 <AuthnContextClassRef> [Optional]

1128 A URI reference identifying an authentication context class that describes the authentication context  
1129 declaration that follows.

1130 <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

1131 Either an authentication context declaration provided by value, or a URI reference that identifies such  
1132 a declaration. The URI reference MAY directly resolve into an XML document containing the  
1133 referenced declaration.

1134 <AuthenticatingAuthority> [Zero or More]

1135 Zero or more unique identifiers of authentication authorities that were involved in the authentication of  
1136 the principal (not including the assertion issuer, who is presumed to have been involved without being  
1137 explicitly named here).

1138 See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication  
1139 context information.

1140 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**  
1141 complex type:

```
1142 <element name="AuthnContext" type="saml:AuthnContextType"/>  
1143 <complexType name="AuthnContextType">  
1144   <sequence>  
1145     <choice>  
1146       <sequence>  
1147         <element ref="saml:AuthnContextClassRef"/>
```

```

1148         <choice minOccurs="0">
1149             <element ref="saml:AuthnContextDecl"/>
1150             <element ref="saml:AuthnContextDeclRef"/>
1151         </choice>
1152     </sequence>
1153     <choice>
1154         <element ref="saml:AuthnContextDecl"/>
1155         <element ref="saml:AuthnContextDeclRef"/>
1156     </choice>
1157 </choice>
1158 <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1159 maxOccurs="unbounded"/>
1160 </sequence>
1161 </complexType>
1162 <element name="AuthnContextClassRef" type="anyURI"/>
1163 <element name="AuthnContextDeclRef" type="anyURI"/>
1164 <element name="AuthnContextDecl" type="anyType"/>
1165 <element name="AuthenticatingAuthority" type="anyURI"/>

```

## 1166 2.7.3 Element <AttributeStatement>

1167 The <AttributeStatement> element describes a statement by the SAML authority asserting that the  
1168 assertion subject is associated with the specified attributes. Assertions containing  
1169 <AttributeStatement> elements MUST contain a <Subject> element.

1170 It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the  
1171 following elements:

1172 <Attribute> or <EncryptedAttribute> [One or More]

1173 The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML  
1174 attribute may be included with the <EncryptedAttribute> element.

1175 The following schema fragment defines the <AttributeStatement> element and its  
1176 **AttributeStatementType** complex type:

```

1177 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1178 <complexType name="AttributeStatementType">
1179     <complexContent>
1180         <extension base="saml:StatementAbstractType">
1181             <choice maxOccurs="unbounded">
1182                 <element ref="saml:Attribute"/>
1183                 <element ref="saml:EncryptedAttribute"/>
1184             </choice>
1185         </extension>
1186     </complexContent>
1187 </complexType>

```

### 1188 2.7.3.1 Element <Attribute>

1189 The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the  
1190 **AttributeType** complex type. It is used within an attribute statement to express particular attributes and  
1191 values associated with an assertion subject, as described in the previous section. It is also used in an  
1192 attribute query to request that the values of specific SAML attributes be returned (see section 3.3.2.4 for  
1193 more information). The <Attribute> element contains the following XML attributes:

1194 Name [Required]

1195 The name of the attribute.

1196 NameFormat [Optional]

1197 A URI reference representing the classification of the attribute name for purposes of interpreting the

1198 name. See Section 8.2 for some URI references that MAY be used as the value of the `NameFormat`  
1199 attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided,  
1200 the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section  
1201 8.2.1) is in effect.

1202 `FriendlyName` [Optional]

1203 A string that provides a more human-readable form of the attribute's name, which may be useful in  
1204 cases in which the actual `Name` is complex or opaque, such as an OID or a UUID. This attribute's  
1205 value MUST NOT be used as a basis for formally identifying SAML attributes.

1206 Arbitrary attributes

1207 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to  
1208 be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows  
1209 additional fields to be added as needed to supply additional parameters to be used, for example, in an  
1210 attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or  
1211 XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a  
1212 derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1213 `<AttributeValue>` [Any Number]

1214 Contains a value of the attribute. If an attribute contains more than one discrete value, it is  
1215 RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than  
1216 one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a  
1217 datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have  
1218 the identical datatype assigned.

1219 The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on  
1220 its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the  
1221 `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of  
1222 values indicates that the requester is interested in any or all of the named attribute's values (see also  
1223 Section 3.3.2.4).

1224 Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the  
1225 semantics of specifying or omitting `<AttributeValue>` elements.

1226 The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
1227 <element name="Attribute" type="saml:AttributeType"/>
1228 <complexType name="AttributeType">
1229   <sequence>
1230     <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
1231   </sequence>
1232   <attribute name="Name" type="string" use="required"/>
1233   <attribute name="NameFormat" type="anyURI" use="optional"/>
1234   <attribute name="FriendlyName" type="string" use="optional"/>
1235   <anyAttribute namespace="##other" processContents="lax"/>
1236 </complexType>
```

### 1237 2.7.3.1.1 Element `<AttributeValue>`

1238 The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the  
1239 **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

1240 If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as  
1241 **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration  
1242 in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data  
1243 elements MAY be defined in an extension schema.

1244 **Note:** Specifying a datatype other than an XML Schema simple type on  
1245 `<AttributeValue>` using `xsi:type` will require the presence of the extension schema  
1246 that defines the datatype in order for schema processing to proceed.

1247 If a SAML attribute includes an empty value, such as the empty string, the corresponding  
1248 `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`).  
1249 This overrides the requirement in section 1.3.1 that string values in SAML content contain at least one  
1250 non-whitespace character.

1251 If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be  
1252 empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

1253 The following schema fragment defines the `<AttributeValue>` element:

```
1254 <element name="AttributeValue" type="anyType" nillable="true"/>
```

### 1255 2.7.3.2 Element `<EncryptedAttribute>`

1256 The `<EncryptedAttribute>` element represents a SAML attribute in encrypted fashion, as defined by  
1257 the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAttribute>`  
1258 element contains the following elements:

1259 `<xenc:EncryptedData>` [Required]

1260 The encrypted content and associated encryption details, as defined by the XML Encryption  
1261 Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if  
1262 present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The  
1263 encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

1264 `<xenc:EncryptedKey>` [Zero or More]

1265 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a  
1266 `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of  
1267 the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name  
1268 identifier, as defined by Section 8.3.6.

1269 Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through  
1270 an intermediary.

1271 The following schema fragment defines the `<EncryptedAttribute>` element:

```
1272 <element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

### 1273 2.7.4 Element `<AuthzDecisionStatement>`

1274 **Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0,  
1275 with no future enhancements planned. Users who require additional functionality may  
1276 want to consider the eXtensible Access Control Markup Language [XACML], which offers  
1277 enhanced authorization decision features.

1278 The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that  
1279 a request for access by the assertion subject to the specified resource has resulted in the specified  
1280 authorization decision on the basis of some optionally specified evidence. Assertions containing  
1281 `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

1282 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
1283 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in  
1284 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different



1285 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
1286 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1287 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
1288 dependent. When a scheme uses elements of the common syntax, it will also use the common  
1289 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
1290 with an explicit ".port", where the port is the default for the scheme, is equivalent to one where  
1291 the port is elided.

1292 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the  
1293 URI normalized form wherever possible as follows:

- 1294 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1295 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1296 Inconsistent URI reference interpretation can also result from differences between the URI reference  
1297 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
1298 employed to specify an access control policy language. The following security conditions SHOULD be  
1299 satisfied by the system which employs SAML assertions:

- 1300 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
1301 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
1302 part of the resource URI reference.
- 1303 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
1304 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
1305 gain access to a denied resource by creating such an equivalence.

1306 The <AuthzDecisionStatement> element is of type **AuthzDecisionStatementType**, which extends  
1307 **StatementAbstractType** with the addition of the following elements and attributes:

1308 Resource [Required]

1309 A URI reference identifying the resource to which access authorization is sought. This attribute MAY  
1310 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the  
1311 current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1312 Decision [Required]

1313 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
1314 the **DecisionType** simple type.

1315 <Action> [One or more]

1316 The set of actions authorized to be performed on the specified resource.

1317 <Evidence> [Optional]

1318 A set of assertions that the SAML authority relied on in making the decision.

1319 The following schema fragment defines the <AuthzDecisionStatement> element and its  
1320 **AuthzDecisionStatementType** complex type:

```
1321 <element name="AuthzDecisionStatement"  
1322 type="saml:AuthzDecisionStatementType"/>  
1323 <complexType name="AuthzDecisionStatementType">  
1324   <complexContent>  
1325     <extension base="saml:StatementAbstractType">  
1326       <sequence>  
1327         <element ref="saml:Action" maxOccurs="unbounded"/>  
1328         <element ref="saml:Evidence" minOccurs="0"/>  
1329       </sequence>  
1330       <attribute name="Resource" type="anyURI" use="required"/>  
1331       <attribute name="Decision" type="saml:DecisionType" use="required"/>
```



```
1332     </extension>
1333     </complexContent>
1334 </complexType>
```

### 1335 2.7.4.1 Simple Type **DecisionType**

1336 The **DecisionType** simple type defines the possible values to be reported as the status of an  
1337 authorization decision statement.

1338 Permit

1339 The specified action is permitted.

1340 Deny

1341 The specified action is denied.

1342 Indeterminate

1343 The SAML authority cannot determine whether the specified action is permitted or denied.

1344 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to  
1345 provide an affirmative statement but where it is not able to issue a decision. Additional information as to  
1346 the reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>`  
1347 elements in the enclosing `<Response>`.

1348 The following schema fragment defines the **DecisionType** simple type:

```
1349 <simpleType name="DecisionType">
1350   <restriction base="string">
1351     <enumeration value="Permit"/>
1352     <enumeration value="Deny"/>
1353     <enumeration value="Indeterminate"/>
1354   </restriction>
1355 </simpleType>
```

### 1356 2.7.4.2 Element **<Action>**

1357 The `<Action>` element specifies an action on the specified resource for which permission is sought. Its  
1358 string-data content provides the label for an action sought to be performed on the specified resource, and  
1359 it has the following attribute:

1360 Namespace [Optional]

1361 A URI reference representing the namespace in which the name of the specified action is to be  
1362 interpreted. If this element is absent, the namespace  
1363 `urn:oasis:names:tc:SAML:1.0:action:rwdc-negation` specified in Section 8.1.2 is in  
1364 effect.

1365 The following schema fragment defines the `<Action>` element and its **ActionType** complex type:

```
1366 <element name="Action" type="saml:ActionType"/>
1367 <complexType name="ActionType">
1368   <simpleContent>
1369     <extension base="string">
1370       <attribute name="Namespace" type="anyURI" use="required"/>
1371     </extension>
1372   </simpleContent>
1373 </complexType>
```

### 1374 2.7.4.3 Element <Evidence>

1375 The <Evidence> element contains one or more assertions or assertion references that the SAML  
1376 authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains  
1377 a mixture of one or more of the following elements:

1378 <AssertionIDRef> [Any number]

1379 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1380 <AssertionURIRef> [Any number]

1381 Specifies an assertion by means of a URI reference.

1382 <Assertion> [Any number]

1383 Specifies an assertion by value.

1384 <EncryptedAssertion> [Any number]

1385 Specifies an encrypted assertion by value.

1386 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
1387 and the SAML authority making the authorization decision. For example, in the case that the SAML relying  
1388 party presented an assertion to the SAML authority in a request, the SAML authority MAY use that  
1389 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's  
1390 assertion as valid either to the relying party or any other third party.

1391 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
1392 <element name="Evidence" type="saml:EvidenceType"/>  
1393 <complexType name="EvidenceType">  
1394   <choice maxOccurs="unbounded">  
1395     <element ref="saml:AssertionIDRef"/>  
1396     <element ref="saml:AssertionURIRef"/>  
1397     <element ref="saml:Assertion"/>  
1398     <element ref="saml:EncryptedAssertion"/>  
1399   </choice>  
1400 </complexType>
```

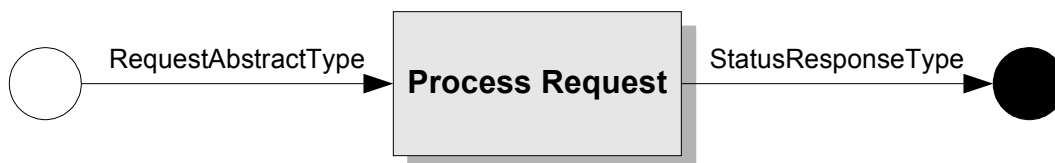
## 3 SAML Protocols

1401

1402 SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML  
1403 bindings specification [SAMLBind] describes specific means of transporting protocol messages using  
1404 existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a  
1405 number of applications of the protocols defined in this section together with additional processing rules,  
1406 restrictions, and requirements that facilitate interoperability.

1407 Specific SAML request and response messages derive from common types. The requester sends an  
1408 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an  
1409 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1410



1412

Figure 1: SAML Request-Response Protocol

1413 In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the  
1414 responder having received a corresponding request.

1415 The protocols defined by SAML achieve the following actions:

- 1416 • Returning one or more requested assertions. This can occur in response to either a direct request  
1417 for specific assertions or a query for assertions that meet particular criteria.
- 1418 • Performing authentication on request and returning the corresponding assertion
- 1419 • Registering a name identifier or terminating a name registration on request
- 1420 • Retrieving a protocol message that has been requested by means of an artifact
- 1421 • Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on  
1422 request
- 1423 • Providing a name identifier mapping on request

1424 Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not  
1425 shown with the conventional namespace prefix `samlp:.` For clarity, text descriptions of elements and  
1426 types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:.`

### 3.1 Schema Header and Namespace Declarations

1427

1428 The following schema fragment defines the XML namespaces and other header information for the  
1429 protocol schema:

1430

```
1431 <schema  
1432   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1433   xmlns="http://www.w3.org/2001/XMLSchema"  
1434   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1435   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1436   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1437   elementFormDefault="unqualified"  
1438   attributeFormDefault="unqualified"  
   blockDefault="substitution"/>
```

```

1439     version="2.0">
1440     <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1441       schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
1442     <import namespace="http://www.w3.org/2000/09/xmlsig#"
1443       schemaLocation="http://www.w3.org/TR/2002/REC-xmlsig-core-
1444 20020212/xmlsig-core-schema.xsd"/>
1445     <annotation>
1446       <documentation>
1447         Document identifier: draft-sstc-saml-schema-protocol-2.0
1448         Location: http://www.oasis-
1449 open.org/committees/documents.php?wg_abbrev=security
1450         Revision history:
1451         V1.0 (November, 2002):
1452           Initial Standard Schema.
1453         V1.1 (September, 2003):
1454           Updates within the same V1.0 namespace.
1455         V2.0 CD-03 (December, 2004):
1456           New protocol schema based in a SAML V2.0 namespace.
1457       </documentation>
1458     </annotation>
1459     ...
1460 </schema>

```

## 1461 3.2 Requests and Responses

1462 The following sections define the SAML constructs and basic requirements that underlie all of the request  
 1463 and response messages used in SAML protocols.

### 1464 3.2.1 Complex Type RequestAbstractType

1465 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
 1466 This type defines common attributes and elements that are associated with all SAML requests:

1467 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**  
 1468 for V2.0 of SAML.

#### 1469 ID [Required]

1470 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section  
 1471 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo  
 1472 attribute in the corresponding response MUST match.

#### 1473 Version [Required]

1474 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".  
 1475 SAML versioning is discussed in Section 4.

#### 1476 IssueInstant [Required]

1477 The time instant of issue of the request. The time value is encoded in UTC, as described in Section  
 1478 1.3.3.

#### 1479 Destination [Optional]

1480 A URI reference indicating the address to which this request has been sent. This is useful to prevent  
 1481 malicious forwarding of requests to unintended recipients, a protection that is required by some  
 1482 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the  
 1483 location at which the message was received. If it does not, the request MUST be discarded. Some  
 1484 protocol bindings may require the use of this attribute (see [SAMLBind]).

1485 `Consent` [Optional]

1486 Indicates whether or not (and under what conditions) consent has been obtained from a principal in  
1487 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value  
1488 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the  
1489 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in  
1490 effect.

1491 `<saml:Issuer>` [Optional]

1492 Identifies the entity that generated the request message. (For more information on this element, see  
1493 Section 2.2.5.)

1494 `<ds:Signature>` [Optional]

1495 An XML Signature that authenticates the requester and provides message integrity, as described  
1496 below and in Section 5.

1497 `<Extensions>` [Optional]

1498 This extension point contains optional protocol message extension elements that are agreed on  
1499 between the communicating parties. No extension schema is required in order to make use of this  
1500 extension point, and even if one is provided, the lax validation setting does not impose a requirement  
1501 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-  
1502 SAML-defined namespace.

1503 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to  
1504 authenticate itself, and message integrity may often be required. Authentication and message integrity  
1505 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request  
1506 MAY be signed, which provides both authentication of the requester and message integrity.

1507 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML  
1508 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)  
1509 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the  
1510 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the  
1511 signature to determine the identity and appropriateness of the signer and may continue to process the  
1512 request or respond with an error (if the request is invalid for some other reason).

1513 If a `Consent` attribute is included and the value indicates that some form of principal consent has been  
1514 obtained, then the request SHOULD be signed.

1515 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if  
1516 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value  
1517 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a  
1518 suspected denial-of-service attack, not responding at all may be warranted.

1519 The following schema fragment defines the **RequestAbstractType** complex type:

```
1520 <complexType name="RequestAbstractType" abstract="true">
1521   <sequence>
1522     <element ref="saml:Issuer" minOccurs="0"/>
1523     <element ref="ds:Signature" minOccurs="0"/>
1524     <element ref="samlp:Extensions" minOccurs="0"/>
1525   </sequence>
1526   <attribute name="ID" type="ID" use="required"/>
1527   <attribute name="Version" type="string" use="required"/>
1528   <attribute name="IssueInstant" type="dateTime" use="required"/>
1529   <attribute name="Destination" type="anyURI" use="optional"/>
1530   <attribute name="Consent" type="anyURI" use="optional"/>
1531 </complexType>
1532 <element name="Extensions" type="samlp:ExtensionsType"/>
1533 <complexType name="ExtensionsType">
1534   <sequence>
1535     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
```

1536  
1537

```
</sequence>  
</complexType>
```

### 1538 3.2.2 Complex Type StatusResponseType

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

#### 1541 ID [Required]

1542 An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in  
1543 Section 1.3.4 for identifier uniqueness.

#### 1544 InResponseTo [Optional]

1545 A reference to the identifier of the request to which the response corresponds, if any. If the response  
1546 is not generated in response to a request, or if the ID attribute value of a request cannot be  
1547 determined (for example, the request is malformed), then this attribute MUST NOT be present.  
1548 Otherwise, it MUST be present and its value MUST match the value of the corresponding request's  
1549 ID attribute.

#### 1550 Version [Required]

1551 The version of this response. The identifier for the version of SAML defined in this specification is  
1552 "2.0". SAML versioning is discussed in Section 4.

#### 1553 IssueInstant [Required]

1554 The time instant of issue of the response. The time value is encoded in UTC, as described in Section  
1555 1.3.3.

#### 1556 Destination [Optional]

1557 A URI reference indicating the address to which this response has been sent. This is useful to prevent  
1558 malicious forwarding of responses to unintended recipients, a protection that is required by some  
1559 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the  
1560 location at which the message was received. If it does not, the response MUST be discarded. Some  
1561 protocol bindings may require the use of this attribute (see [SAMLBind]).

#### 1562 Consent [Optional]

1563 Indicates whether or not (and under what conditions) consent has been obtained from a principal in  
1564 the sending of this response. See Section 8.4 for some URI references that MAY be used as the value  
1565 of the **Consent** attribute and their associated descriptions. If no **Consent** value is provided, the  
1566 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in  
1567 effect.

#### 1568 <saml:Issuer> [Optional]

1569 Identifies the entity that generated the response message. (For more information on this element, see  
1570 Section 2.2.5.)

#### 1571 <ds:Signature> [Optional]

1572 An XML Signature that authenticates the responder and provides message integrity, as described  
1573 below and in Section 5.

#### 1574 <Extensions> [Optional]

1575 This extension point contains optional protocol message extension elements that are agreed on  
1576 between the communicating parties. . No extension schema is required in order to make use of this  
1577 extension point, and even if one is provided, the lax validation setting does not impose a requirement  
1578 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-  
1579 SAML-defined namespace.

1580 <Status> [Required]

1581 A code representing the status of the corresponding request.

1582 Depending on the requirements of particular protocols or profiles, a SAML responder may often need to  
1583 authenticate itself, and message integrity may often be required. Authentication and message integrity  
1584 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML  
1585 response MAY be signed, which provides both authentication of the responder and message integrity.

1586 If such a signature is used, then the <ds:Signature> element MUST be present, and the SAML  
1587 requester receiving the response MUST verify that the signature is valid (that is, that the message has not  
1588 been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on  
1589 the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD  
1590 evaluate the signature to determine the identity and appropriateness of the signer and may continue to  
1591 process the response as it deems appropriate.

1592 If a Consent attribute is included and the value indicates that some form of principal consent has been  
1593 obtained, then the response SHOULD be signed.

1594 The following schema fragment defines the **StatusResponseType** complex type:

```
1595 <complexType name="StatusResponseType">  
1596   <sequence>  
1597     <element ref="saml:Issuer" minOccurs="0"/>  
1598     <element ref="ds:Signature" minOccurs="0"/>  
1599     <element ref="samlp:Extensions" minOccurs="0"/>  
1600     <element ref="samlp:Status"/>  
1601   </sequence>  
1602   <attribute name="ID" type="ID" use="required"/>  
1603   <attribute name="InResponseTo" type="NCName" use="optional"/>  
1604   <attribute name="Version" type="string" use="required"/>  
1605   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1606   <attribute name="Destination" type="anyURI" use="optional"/>  
1607   <attribute name="Consent" type="anyURI" use="optional"/>  
1608 </complexType>
```

### 1609 3.2.2.1 Element <Status>

1610 The <Status> element contains the following elements:

1611 <StatusCode> [Required]

1612 A code representing the status of the activity carried out in response to the corresponding request.

1613 <StatusMessage> [Optional]

1614 A message which MAY be returned to an operator.

1615 <StatusDetail> [Optional]

1616 Additional information concerning the status of the request.

1617 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1618 <element name="Status" type="samlp:StatusType"/>  
1619 <complexType name="StatusType">  
1620   <sequence>  
1621     <element ref="samlp:StatusCode"/>  
1622     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1623     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1624   </sequence>  
1625 </complexType>
```

### 1626 **3.2.2.2 Element <StatusCode>**

1627 The <StatusCode> element specifies a code or a set of nested codes representing the status of the  
1628 corresponding request. The <StatusCode> element has the following element and attribute:

1629 Value [Required]

1630 The status code value. This attribute contains a URI reference. The value of the topmost  
1631 <StatusCode> element MUST be from the top-level list provided in this section.

1632 <StatusCode> [Optional]

1633 A subordinate status code that provides more specific information on an error condition. Note that  
1634 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for  
1635 additional information by intentionally presenting erroneous requests.

1636 The permissible top-level <StatusCode> values are as follows:

1637 urn:oasis:names:tc:SAML:2.0:status:Success

1638 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or  
1639 <StatusDetail> elements.

1640 urn:oasis:names:tc:SAML:2.0:status:Requester

1641 The request could not be performed due to an error on the part of the requester.

1642 urn:oasis:names:tc:SAML:2.0:status:Responder

1643 The request could not be performed due to an error on the part of the SAML responder or SAML  
1644 authority.

1645 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1646 The SAML responder could not process the request because the version of the request message was  
1647 incorrect.

1648 The following second-level status codes are referenced at various places in this specification. Additional  
1649 second-level status codes MAY be defined in future versions of the SAML specification. System entities  
1650 are free to define more specific status codes by defining appropriate URI references.

1651 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1652 The responding provider was unable to successfully authenticate the principal.

1653 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1654 Unexpected or invalid content was encountered within a <saml:Attribute> or  
1655 <saml:AttributeValue> element.

1656 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1657 The responding provider cannot or will not support the requested name identifier policy.

1658 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1659 The specified authentication context requirements cannot be met by the responder.

1660 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1661 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an  
1662 <IDPList> can be resolved or that none of the supported identity providers are available.

1663 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1664 Indicates the responding provider cannot authenticate the principal passively, as has been requested.



1665 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP  
1666 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are  
1667 supported by the intermediary.

1668 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded  
1669 Indicates that a responding provider cannot authenticate the principal directly and is not permitted to  
1670 proxy the request further.

1671 urn:oasis:names:tc:SAML:2.0:status:RequestDenied  
1672 The SAML responder or SAML authority is able to process the request but has chosen not to respond.  
1673 This status code MAY be used when there is concern about the security context of the request  
1674 message or the sequence of request messages received from a particular requester.

1675 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported  
1676 The SAML responder or SAML authority does not support the request.

1677 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated  
1678 The SAML responder cannot process any requests with the protocol version specified in the request.

1679 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh  
1680 The SAML responder cannot process the request because the protocol version specified in the  
1681 request message is a major upgrade from the highest protocol version supported by the responder.

1682 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow  
1683 The SAML responder cannot process the request because the protocol version specified in the  
1684 request message is too low.

1685 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized  
1686 The resource value provided in the request message is invalid or unrecognized.

1687 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses  
1688 The response message would contain more elements than the SAML responder is able to return.

1689 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile  
1690 An entity that has no knowledge of a particular attribute profile has been presented with an attribute  
1691 drawn from that profile.

1692 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal  
1693 The responding provider does not recognize the principal specified or implied by the request.

1694 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding  
1695 The SAML responder cannot properly fulfill the request using the protocol binding specified in the  
1696 request.

1697 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
1698 type:

```
1699 <element name="StatusCode" type="samlp:StatusCodeType"/>  
1700 <complexType name="StatusCodeType">  
1701   <sequence>  
1702     <element ref="samlp:StatusCode" minOccurs="0"/>  
1703   </sequence>  
1704   <attribute name="Value" type="anyURI" use="required"/>  
1705 </complexType>
```

### 1706 3.2.2.3 Element <StatusMessage>

1707 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1708 The following schema fragment defines the <StatusMessage> element:

```
1709 <element name="StatusMessage" type="string"/>
```

### 1710 3.2.2.4 Element <StatusDetail>

1711 The <StatusDetail> element MAY be used to specify additional information concerning the status of  
1712 the request. The additional information consists of zero or more elements from any namespace, with no  
1713 requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

1714 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1715 complex type:

```
1716 <element name="StatusDetail" type="saml:StatusDetailType"/>  
1717 <complexType name="StatusDetailType">  
1718   <sequence>  
1719     <any namespace="##any" processContents="lax" minOccurs="0"  
1720     maxOccurs="unbounded"/>  
1721   </sequence>  
1722 </complexType>
```

## 1723 3.3 Assertion Query and Request Protocol

1724 This section defines messages and processing rules for requesting existing assertions by reference or  
1725 querying for assertions by subject and statement type.

### 1726 3.3.1 Element <AssertionIDRequest>

1727 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>  
1728 message element can be used to request that they be returned in a <Response> message. The  
1729 <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for  
1730 more information on this element.

1731 The following schema fragment defines the <AssertionIDRequest> element:

```
1732 <element name="AssertionIDRequest" type="saml:AssertionIDRequestType"/>  
1733 <complexType name="AssertionIDRequestType">  
1734   <complexContent>  
1735     <extension base="saml:RequestAbstractType">  
1736       <sequence>  
1737         <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>  
1738       </sequence>  
1739     </extension>  
1740   </complexContent>  
1741 </complexType>
```

### 1742 3.3.2 Queries

1743 The following sections define the SAML query request messages.

#### 1744 3.3.2.1 Element <SubjectQuery>

1745 The <SubjectQuery> message element is an extension point that allows new SAML queries to be  
1746 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and

1747 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the  
1748 <saml:Subject> element (defined in Section 2.4) to **RequestAbstractType**.

1749 The following schema fragment defines the <SubjectQuery> element and its  
1750 **SubjectQueryAbstractType** complex type:

```
1751 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>  
1752 <complexType name="SubjectQueryAbstractType" abstract="true">  
1753   <complexContent>  
1754     <extension base="samlp:RequestAbstractType">  
1755       <sequence>  
1756         <element ref="saml:Subject"/>  
1757       </sequence>  
1758     </extension>  
1759   </complexContent>  
1760 </complexType>
```

### 1761 3.3.2.2 Element <AuthnQuery>

1762 The <AuthnQuery> message element is used to make the query “What assertions containing  
1763 authentication statements are available for this subject?” A successful <Response> will contain one or  
1764 more assertions containing authentication statements.

1765 The <AuthnQuery> message MUST NOT be used as a request for a new authentication using  
1766 credentials provided in the request. <AuthnQuery> is a request for statements about authentication acts  
1767 that have occurred in a previous interaction between the indicated subject and the authentication authority.

1768 This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1769 the following element and attribute:

1770 SessionIndex [Optional]

1771 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1772 containing authentication statements do you have for this subject within the context of the supplied  
1773 session information?”

1774 <RequestedAuthnContext> [Optional]

1775 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1776 containing authentication statements do you have for this subject that satisfy the authentication  
1777 context requirements in this element?”

1778 In response to an authentication query, a SAML authority returns assertions with authentication  
1779 statements as follows:

- 1780 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the  
1781 assertions that may be returned.
- 1782 • If the SessionIndex attribute is present in the query, at least one <AuthnStatement> element in  
1783 the set of returned assertions MUST contain a SessionIndex attribute that matches the  
1784 SessionIndex attribute in the query. It is OPTIONAL for the complete set of all such matching  
1785 assertions to be returned in the response.
- 1786 • If the <RequestedAuthnContext> element is present in the query, at least one  
1787 <AuthnStatement> element in the set of returned assertions MUST contain an  
1788 <AuthnContext> element that satisfies the element in the query (see Section 3.3.2.3). It is  
1789 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1790 The following schema fragment defines the <AuthnQuery> element and its **AuthnQueryType** complex  
1791 type:

```
1792 <element name="AuthnQuery" type="samlp:AuthnQueryType"/>
```

```

1793 <complexType name="AuthnQueryType">
1794   <complexContent>
1795     <extension base="samlp:SubjectQueryAbstractType">
1796       <sequence>
1797         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1798       </sequence>
1799       <attribute name="SessionIndex" type="string" use="optional"/>
1800     </extension>
1801   </complexContent>
1802 </complexType>

```

### 1803 3.3.2.3 Element <RequestedAuthnContext>

1804 The <RequestedAuthnContext> element specifies the authentication context requirements of  
 1805 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**  
 1806 complex type defines the following elements and attributes:

1807 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1808 Specifies one or more URI references identifying authentication context classes or declarations.  
 1809 (These elements are defined in Section 2.7.2.2. For more information about authentication context  
 1810 classes, see [SAMLAuthnCxt].)

1811 Comparison [Optional]

1812 Specifies the comparison method used to evaluate the requested context classes or statements, one  
 1813 of "exact", "minimum", "maximum", or "better". The default is "exact".

1814 Either a set of class references or a set of declaration references can be used. The set of supplied  
 1815 references **MUST** be evaluated as an ordered set, where the first element is the most preferred  
 1816 authentication context class or declaration. If none of the specified classes or declarations can be satisfied  
 1817 in accordance with the rules below, then the responder **MUST** return a <Response> message with a  
 1818 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1819 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
 1820 statement **MUST** be the exact match of at least one of the authentication contexts specified.

1821 If Comparison is set to "minimum", then the resulting authentication context in the authentication  
 1822 statement **MUST** be at least as strong (as deemed by the responder) as one of the authentication  
 1823 contexts specified.

1824 If Comparison is set to "better", then the resulting authentication context in the authentication  
 1825 statement **MUST** be stronger (as deemed by the responder) than any one of the authentication contexts  
 1826 specified.

1827 If Comparison is set to "maximum", then the resulting authentication context in the authentication  
 1828 statement **MUST** be as strong as possible (as deemed by the responder) without exceeding the strength  
 1829 of at least one of the authentication contexts specified.

1830 The following schema fragment defines the <RequestedAuthnContext> element and its  
 1831 **RequestedAuthnContextType** complex type:

```

1832 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1833 <complexType name="RequestedAuthnContextType">
1834   <choice>
1835     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1836     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1837   </choice>
1838   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1839   use="optional"/>
1840 </complexType>
1841 <simpleType name="AuthnContextComparisonType">

```

```

1842     <restriction base="string">
1843         <enumeration value="exact"/>
1844         <enumeration value="minimum"/>
1845         <enumeration value="maximum"/>
1846         <enumeration value="better"/>
1847     </restriction>
1848 </simpleType>

```

### 1849 3.3.2.4 Element <AttributeQuery>

1850 The <AttributeQuery> element is used to make the query "Return the requested attributes for this  
1851 subject." A successful response will be in the form of assertions containing attribute statements, to the  
1852 extent allowed by policy. This element is of type **AttributeQueryType**, which extends  
1853 **SubjectQueryAbstractType** with the addition of the following element:

1854 <saml:Attribute> [Any Number]

1855 Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no  
1856 attributes are specified, it indicates that all attributes allowed by policy are requested. If a given  
1857 <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if  
1858 that attribute is returned in the response, it MUST NOT contain any values that are not equal to the  
1859 values specified in the query. In the absence of equality rules specified by particular profiles or  
1860 attributes, equality is defined as an identical XML representation of the value. (For more information  
1861 on <saml:Attribute>, see Section 2.7.3.1.)

1862 A single query MUST NOT contain two <saml:Attribute> elements with the same Name and  
1863 NameFormat values (that is, a given attribute MUST be named only once in a query).

1864 In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- 1865 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the  
1866 assertions that may be returned.
- 1867 • If any <Attribute> elements are present in the query, they constrain/filter the attributes and  
1868 optionally the values returned, as noted above.
- 1869 • The attributes and values returned MAY also be constrained by application-specific policy  
1870 considerations.

1871 The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile  
1872 and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to  
1873 indicate problems with the interpretation of attribute or value information in a query.

1874 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**  
1875 complex type:

```

1876 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1877 <complexType name="AttributeQueryType">
1878     <complexContent>
1879         <extension base="samlp:SubjectQueryAbstractType">
1880             <sequence>
1881                 <element ref="saml:Attribute" minOccurs="0"
1882 maxOccurs="unbounded"/>
1883             </sequence>
1884         </extension>
1885     </complexContent>
1886 </complexType>

```

### 1887 3.3.2.5 Element <AuthzDecisionQuery>

1888 The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource  
1889 be allowed for this subject, given this evidence?” A successful response will be in the form of assertions  
1890 containing authorization decision statements.

1891 **Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no  
1892 future enhancements planned. Users who require additional functionality may want to  
1893 consider the eXtensible Access Control Markup Language [XACML], which offers  
1894 enhanced authorization decision features.

1895 This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the  
1896 addition of the following elements and attribute:

1897 Resource [Required]

1898 A URI reference indicating the resource for which authorization is requested.

1899 <saml:Action> [One or More]

1900 The actions for which authorization is requested. (For more information on this element, see Section  
1901 2.7.4.2.)

1902 <saml:Evidence> [Optional]

1903 A set of assertions that the SAML authority MAY rely on in making its authorization decision. (For  
1904 more information on this element, see Section 2.7.4.3.)

1905 In response to an authorization decision query, a SAML authority returns assertions with authorization  
1906 decision statements as follows:

- 1907 • Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the  
1908 assertions that may be returned.

1909 The following schema fragment defines the <AuthzDecisionQuery> element and its  
1910 **AuthzDecisionQueryType** complex type:

```
1911 <element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>  
1912 <complexType name="AuthzDecisionQueryType">  
1913   <complexContent>  
1914     <extension base="samlp:SubjectQueryAbstractType">  
1915       <sequence>  
1916         <element ref="saml:Action" maxOccurs="unbounded"/>  
1917         <element ref="saml:Evidence" minOccurs="0"/>  
1918       </sequence>  
1919       <attribute name="Resource" type="anyURI" use="required"/>  
1920     </extension>  
1921   </complexContent>  
1922 </complexType>
```

### 1923 3.3.3 Element <Response>

1924 The <Response> message element is used when a response consists of a list of zero or more assertions  
1925 that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType**  
1926 and adds the following elements:

1927 <saml:Assertion> or <saml:EncryptedAssertion> [Any Number]

1928 Specifies an assertion by value, or optionally an encrypted assertion by value. (See Section 2.3.3 for  
1929 more information on these elements.)

1930 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1931 <element name="Response" type="samlp:ResponseType"/>
1932 <complexType name="ResponseType">
1933   <complexContent>
1934     <extension base="samlp:StatusResponseType">
1935       <choice minOccurs="0" maxOccurs="unbounded">
1936         <element ref="saml:Assertion"/>
1937         <element ref="saml:EncryptedAssertion"/>
1938       </choice>
1939     </extension>
1940   </complexContent>
1941 </complexType>

```

### 1942 3.3.4 Processing Rules

1943 In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST**  
 1944 contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the  
 1945 query.

1946 A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both  
 1947 apply:

- 1948 • If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST**  
 1949 include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2.  
 1950 In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical"  
 1951 means that the identifier element's content and attribute values **MUST** be the same. An encrypted  
 1952 identifier will be identical to the original according to this definition, once decrypted.
- 1953 • If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at  
 1954 least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner  
 1955 described by at least one `<saml:SubjectConfirmation>` element in S2.

1956 As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular  
 1957 Format value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting  
 1958 S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with  
 1959 different Format values and element content, even if the two identifiers are considered to refer to the  
 1960 same principal.

1961 If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
 1962 expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an  
 1963 `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value  
 1964 `urn:oasis:names:tc:SAML:2.0:status:Success`.

1965 All other processing rules associated with the underlying request and response messages **MUST** be  
 1966 observed.

## 1967 3.4 Authentication Request Protocol

1968 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing  
 1969 authentication statements to establish a security context at one or more relying parties, it can use the  
 1970 authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and  
 1971 request that it return a `<Response>` message containing one or more such assertions. Such assertions  
 1972 **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one  
 1973 authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

1974 Apart from this requirement, the specific contents of the returned assertions depend on the profile or  
 1975 context of use. Also, the exact means by which the principal or agent authenticates to the identity provider  
 1976 is not specified, though the means of authentication might impact the content of the response. Other  
 1977 issues related to the validation of authentication credentials by the identity provider or any communication



1978 between the identity provider and any other entities involved in the authentication process are also out of  
1979 scope of this protocol.

1980 The descriptions and processing rules in the following sections reference the following actors, many of  
1981 whom might be the same entity in a particular profile of use:

1982 Request Issuer

1983 The entity who creates the authentication request and to whom the response is to be returned.

1984 Presenter

1985 The entity who presents the request to the identity provider and either authenticates itself during  
1986 the transmission of the message, or relies on an existing security context to establish its identity. If  
1987 not the request issuer, the presenter acts as an intermediary between the request issuer and the  
1988 responding identity provider.

1989 Requested Subject

1990 The entity about whom one or more assertions are being requested.

1991 Confirming Entity

1992 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`  
1993 elements of the resulting assertion(s).

1994 Relying Party

1995 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1996 the profile or context of use, generally to establish a security context.

1997 Identity Provider

1998 The entity to whom the presenter gives the request and from whom the presenter receives the  
1999 response.

### 2000 **3.4.1 Element `<AuthnRequest>`**

2001 To request that an identity provider issue an assertion with an authentication statement, a presenter  
2002 authenticates to that identity provider (or relies on an existing security context) and sends it an  
2003 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to  
2004 satisfy its purpose. Among these properties may be information that relates to the content of the assertion  
2005 and/or information that relates to how the resulting `<Response>` message should be delivered to the  
2006 request issuer. The process of authentication of the presenter may take place before, during, or after the  
2007 initial delivery of the `<AuthnRequest>` message.

2008 The request issuer might not be the same as the presenter of the request if, for example, the request  
2009 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the  
2010 requested subject so that the relying party can decide whether to provide a service.

2011 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected  
2012 by the protocol binding used to deliver the message.

2013 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
2014 adds the following elements and attributes, all of which are optional in general, but may be required by  
2015 specific profiles:

2016 `<saml:Subject>` [Optional]

2017 Specifies the requested subject of the resulting assertion(s). This may include one or more  
2018 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions  
2019 can be confirmed. (For more information on this element, see Section 2.4.)



2020 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
2021 requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the presenter  
2022 is presumed to be the only confirming entity required and the method is implied by the profile of use  
2023 and/or the policies of the identity provider.

2024 `<NameIDPolicy>` [Optional]  
2025 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,  
2026 then any type of identifier supported by the identity provider for the requested subject can be used,  
2027 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2028 `<saml:Conditions>` [Optional]  
2029 Specifies the SAML conditions the request issuer expects to limit the validity and/or use of the  
2030 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary. The  
2031 information in this element is used as input to the process of constructing the assertion, rather than as  
2032 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2033 `<RequestedAuthnContext>` [Optional]  
2034 Specifies the requirements, if any, that the request issuer places on the authentication context that  
2035 applies to the responding provider's authentication of the presenter. See Section 3.3.2.3 for  
2036 processing rules regarding this element.

2037 `<Scoping>` [Optional]  
2038 Specifies a set of identity providers trusted by the request issuer to authenticate the presenter, as well  
2039 as limitations and context related to proxying of the `<AuthnRequest>` message to subsequent  
2040 identity providers by the responder.

2041 `ForceAuthn` [Optional]  
2042 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
2043 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
2044 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the  
2045 presenter unless the constraints of `IsPassive` can be met.

2046 `IsPassive` [Optional]  
2047 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of the  
2048 user interface from the request issuer and interact with the presenter in a noticeable fashion. If a value  
2049 is not provided, the default is "false".

2050 `ProtocolBinding` [Optional]  
2051 A URI reference that identifies a SAML protocol binding to be used when returning the `<Response>`  
2052 message. See [SAMLBind] for more information about protocol bindings and URI references defined  
2053 for them.

2054 `AssertionConsumerServiceIndex` [Optional]  
2055 Indirectly identifies the location to which the `<Response>` message should be returned to the request  
2056 issuer. It applies only to profiles in which the request issuer is different from the presenter, such as the  
2057 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map  
2058 the index value in the attribute to a location associated with the request issuer. [SAMLMeta] provides  
2059 one possible mechanism. If omitted, then the identity provider MUST return the `<Response>`  
2060 message to the default location associated with the request issuer for the profile of use. If the index  
2061 specified is invalid, then the identity provider MAY return an error `<Response>` or it MAY use the  
2062 default location. This attribute is mutually exclusive with the `AssertionConsumerServiceURL`  
2063 attribute.

2064 `AssertionConsumerServiceURL` [Optional]  
2065 Specifies by value the location to which the `<Response>` message MUST be returned to the request

2066 issuer. The responder MUST ensure by some means that the value specified is in fact associated with  
2067 the request issuer. [SAMLMeta] provides one possible mechanism; signing the enclosing  
2068 <AuthnRequest> message is another. This attribute is mutually exclusive with the  
2069 AssertionConsumerServiceIndex attribute.

2070 AttributeConsumingServiceIndex [Optional]

2071 Indirectly identifies information associated with the request issuer describing the SAML attributes the  
2072 request issuer desires or requires to be supplied by the identity provider in the <Response>  
2073 message. The identity provider MUST have a trusted means to map the index value in the attribute to  
2074 information associated with the request issuer. [SAMLMeta] provides one possible mechanism. The  
2075 identity provider MAY use this information to populate one or more <saml:AttributeStatement>  
2076 elements in the assertion(s) it returns.

2077 ProviderName [Optional]

2078 Specifies the human-readable name of the request issuer for use by the presenter's user agent or the  
2079 identity provider.

2080 See Section 3.4.1.4 for general processing rules regarding this message.

2081 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**  
2082 complex type:

```
2083 <element name="AuthnRequest" type="saml:AuthnRequestType"/>
2084 <complexType name="AuthnRequestType">
2085   <complexContent>
2086     <extension base="saml:RequestAbstractType">
2087       <sequence>
2088         <element ref="saml:Subject" minOccurs="0"/>
2089         <element ref="saml:NameIDPolicy" minOccurs="0"/>
2090         <element ref="saml:Conditions" minOccurs="0"/>
2091         <element ref="saml:RequestedAuthnContext" minOccurs="0"/>
2092         <element ref="saml:Scoping" minOccurs="0"/>
2093       </sequence>
2094       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2095       <attribute name="IsPassive" type="boolean" use="optional"/>
2096       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2097       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2098 use="optional"/>
2099       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2100 use="optional"/>
2101       <attribute name="AttributeConsumingServiceIndex"
2102 type="unsignedShort" use="optional"/>
2103       <attribute name="ProviderName" type="string" use="optional"/>
2104     </extension>
2105   </complexContent>
2106 </complexType>
```

### 2107 3.4.1.1 Element <NameIDPolicy>

2108 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
2109 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

2110 Format [Required]

2111 Specifies the URI reference corresponding to a name identifier format defined in this or another  
2112 specification (see Section 8.3 for examples). The additional value of  
2113 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use  
2114 within this attribute to indicate a request that the resulting identifier be encrypted.

2115 SPNameQualifier [Optional]

2116 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of

2117 a service provider other than the request issuer, or in the namespace of an affiliation group of service  
2118 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`  
2119 `format:persistent` in section 8.3.7.

2120 **AllowCreate** [Optional]

2121 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the  
2122 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the  
2123 request issuer constrains the identity provider to only issue an assertion to it if an acceptable identifier  
2124 for the principal has already been established. Note that this does not prevent the identity provider  
2125 from creating such identifiers outside the context of this specific request (for example, in advance for  
2126 a large number of principals).

2127 When this element is used, if the content is not understood by or acceptable to the identity provider, then  
2128 a `<Response>` message element **MUST** be returned with an error `<Status>`, and **MAY** contain a  
2129 second-level `<StatusCode>` of  
2130 `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2131 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates  
2132 that the resulting assertion(s) **MUST** contain `<EncryptedID>` elements instead of plaintext. The  
2133 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the  
2134 requested subject.

2135 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider **MAY** return an  
2136 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly  
2137 specific to the service provider) require that an encrypted identifier be used.

2138 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**  
2139 complex type:

```
2140 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>  
2141 <complexType name="NameIDPolicyType">  
2142   <attribute name="Format" type="anyURI" use="required"/>  
2143   <attribute name="SPNameQualifier" type="string" use="optional"/>  
2144   <attribute name="AllowCreate" type="boolean" use="optional"/>  
2145 </complexType>
```

### 2146 **3.4.1.2 Element <Scoping>**

2147 The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the  
2148 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to  
2149 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
2150 elements and attribute:

2151 **ProxyCount** [Optional]

2152 Specifies the number of proxying indirections permissible between the identity provider that receives  
2153 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of  
2154 zero permits no proxying, while omitting this attribute expresses no such restriction.

2155 `<IDPList>` [Optional]

2156 An advisory list of identity providers and associated information that the request issuer deems  
2157 acceptable to respond to the request.

2158 `<RequesterID>` [Zero or More]

2159 Identifies the set of requesting entities on whose behalf the request issuer is acting. Used to  
2160 communicate the chain of request issuers when proxying occurs, as described in Section 3.4.1.5. See  
2161 Section 8.3.6 for a description of entity identifiers.

2162 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
2163 <Response> message with an error <Status> and a second-level <StatusCode> of  
2164 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or  
2165 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support  
2166 any of the specified identity providers.

2167 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
2168 <element name="Scoping" type="samlp:ScopingType"/>
2169 <complexType name="ScopingType">
2170   <sequence>
2171     <element ref="samlp:IDPList" minOccurs="0"/>
2172     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2173   </sequence>
2174   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2175 </complexType>
2176 <element name="RequesterID" type="anyURI"/>
```

### 2177 3.4.1.3 Element <IDPList>

2178 The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the  
2179 presenter. Its **IDPListType** complex type defines the following elements:

2180 <IDPEntry> [One or More]

2181 Information about a single identity provider.

2182 <GetComplete> [Optional]

2183 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to  
2184 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML  
2185 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>  
2186 element.

2187 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
2188 <element name="IDPList" type="samlp:IDPListType"/>
2189 <complexType name="IDPListType">
2190   <sequence>
2191     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2192     <element ref="samlp:GetComplete" minOccurs="0"/>
2193   </sequence>
2194 </complexType>
2195 <element name="GetComplete" type="anyURI"/>
```

#### 2196 3.4.1.3.1 Element <IDPEntry>

2197 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to authenticate  
2198 the presenter. Its **IDPEntryType** complex type defines the following attributes:

2199 ProviderID [Required]

2200 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2201 Name [Optional]

2202 A human-readable name for the identity provider.

2203 Loc [Optional]

2204 A URI reference representing the location of a profile-specific endpoint supporting the authentication  
2205 request protocol. The binding to be used must be understood from the profile of use.

2206 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```

2207 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2208 <complexType name="IDPEntryType">
2209   <attribute name="ProviderID" type="anyURI" use="required"/>
2210   <attribute name="Name" type="string" use="optional"/>
2211   <attribute name="Loc" type="anyURI" use="optional"/>
2212 </complexType>

```

### 2213 3.4.1.4 Processing Rules

2214 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
 2215 therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
 2216 kinds of input and output are required or prohibited. The following processing rules apply as invariant  
 2217 behavior across any profile of this protocol exchange. All other processing rules associated with the  
 2218 underlying request and response messages MUST also be observed.

2219 The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing  
 2220 one or more assertions that meet the specifications defined by the request, or with a <Response>  
 2221 message containing a <Status> describing the error that occurred. The responder MAY conduct  
 2222 additional message exchanges with the presenter as needed to initiate or complete the authentication  
 2223 process, subject to the nature of the protocol binding and the authentication mechanism. As described in  
 2224 the next section, this includes proxying the request by directing the presenter to another identity provider  
 2225 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to  
 2226 authenticate the presenter to the original responder, in effect using SAML as the authentication  
 2227 mechanism.

2228 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if  
 2229 prevented from providing an assertion by policies in effect at the identity provider (for example the  
 2230 intended subject has prohibited the identity provider from providing assertions to the relying party), then it  
 2231 MUST return a <Response> with an error <Status>, and MAY return a second-level <StatusCode> of  
 2232 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or  
 2233 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2234 If the <saml:Subject> element in the request is present, then the resulting assertions'  
 2235 <saml:Subject> MUST **strongly match** the request <saml:Subject>, as described in Section 3.3.4,  
 2236 except that the identifier MAY be in a different format if specified by <NameIDPolicy>. In such a case,  
 2237 the identifier's physical content MAY be different, but it MUST refer to the same principal.

2238 All of the content defined specifically within <AuthnRequest> is optional, although some may be required  
 2239 by certain profiles. In the absence of any specific content at all, the following behavior is assumed:

- 2240 • The assertion(s) returned MUST contain a <saml:Subject> element that represents the  
 2241 presenter. The identifier type and format are determined by the identity provider. At least one  
 2242 statement in at least one assertion MUST be a <saml:AuthnStatement> that describes the  
 2243 authentication performed by the responder or authentication service associated with it.
- 2244 • The request presenter should, to the extent possible, be the only entity able to satisfy the  
 2245 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation  
 2246 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2247 • The resulting assertion(s) MUST contain a <saml:AudienceRestriction> element  
 2248 referencing the request issuer as an acceptable relying party. Other audiences MAY be included as  
 2249 deemed appropriate by the identity provider.

### 2250 3.4.1.5 Proxying

2251 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or  
 2252 cannot directly authenticate the presenter, but believes that the presenter has already authenticated to  
 2253 another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new

2254 <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in  
2255 whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying  
2256 identity provider.

2257 Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the  
2258 enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the  
2259 proxying provider can issue an assertion of its own in response to the original <AuthnRequest>,  
2260 completing the overall message exchange. Both the proxying and authenticating identity providers MAY  
2261 include constraints on proxying activity in the messages and assertions they issue, as described in  
2262 previous sections and below.

2263 The request issuer can influence proxy behavior by including a <Scoping> element where the provider  
2264 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be  
2265 proxied by including an ordered <IDPList> of preferred providers.

2266 An identity provider can control secondary use of its assertions by proxying identity providers using a  
2267 <ProxyRestriction> element in the assertions it issues.

#### 2268 **3.4.1.5.1 Proxying Processing Rules**

2269 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is  
2270 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY  
2271 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

2272 An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. The identity  
2273 provider MUST return an error <Status> containing a second-level <StatusCode> value of  
2274 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded, unless it can directly  
2275 authenticate the presenter.

2276 If it chooses to proxy to a SAML identity provider, when creating the new <AuthnRequest>, the proxying  
2277 identity provider MUST include equivalent or stricter forms of all the information included in the original  
2278 request (such as authentication context policy). Note, however, that the proxying provider is free to specify  
2279 whatever <NameIDPolicy> it wishes to maximize the chances of a successful response.

2280 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST  
2281 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for  
2282 example) will be honored by the authenticating provider.

2283 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less  
2284 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new  
2285 request SHOULD contain a <ProxyCount> attribute.

2286 If an <IDPList> was specified in the original request, the new request MUST also contain an  
2287 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the  
2288 <IDPList>, but MUST NOT remove any from the list.

2289 The authentication request and response are processed in normal fashion, in accordance with the rules  
2290 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity  
2291 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2292 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
2293 relevant information from the original assertion or non-SAML equivalent.
- 2294 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original  
2295 request issuer's preferences, as defined by its <NameIDPolicy> element.
- 2296 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>  
2297 element containing a <saml:AuthenticatingAuthority> element referencing the identity



2298 provider to which the proxying identity provider referred the presenter. If the original assertion  
2299 contains `<saml:AuthnContext>` information that includes one or more  
2300 `<saml:AuthenticatingAuthority>` elements, those elements SHOULD be included in the  
2301 new assertion, with the new element placed after them.

2302 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
2303 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
2304 consistent over time across different requests. The value MUST not conflict with values used or  
2305 generated by other SAML providers.

2306 • Any other `<saml:AuthnContext>` information MAY be copied, translated, or omitted in  
2307 accordance with the policies of the proxying identity provider, provided that the original  
2308 requirements dictated by the request issuer are met.

2309 If, in the future, the identity provider is asked to authenticate the same presenter for a second request  
2310 issuer, and this request is equally or less strict than the original request (as determined by the proxying  
2311 identity provider), the identity provider MAY skip the creation of a new `<AuthnRequest>` to the  
2312 authenticating identity provider and immediately issue another assertion (assuming the original assertion  
2313 or non-SAML equivalent it received is still valid).

## 2314 3.5 Artifact Resolution Protocol

2315 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be  
2316 transported in a SAML binding by reference instead of by value. Both requests and responses can be  
2317 obtained by reference using this specialized protocol. A message sender, instead of binding a message to  
2318 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a  
2319 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver  
2320 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
2321 protocol to resolve the artifact into the original protocol message.

2322 The most common use for this mechanism is with bindings that cannot easily carry a message because of  
2323 size constraints, or to enable a message to be communicated via a secure channel between the SAML  
2324 requester and responder, avoiding the need for a signature.

2325 Depending on the characteristics of the underlying message being passed by reference, the artifact  
2326 resolution protocol MAY require protections such as mutual authentication, integrity protection,  
2327 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST  
2328 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used  
2329 by any party.

2330 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly  
2331 as if the message so obtained had been sent originally in place of the artifact.

### 2332 3.5.1 Element `<ArtifactResolve>`

2333 The `<ArtifactResolve>` message is used to request that a SAML protocol message be returned in an  
2334 `<ArtifactResponse>` message by specifying an artifact that represents the SAML protocol message.  
2335 The original transmission of the artifact is governed by the specific protocol binding that is being used; see  
2336 [SAMLBind] for more information on the use of artifacts in bindings.

2337 The `<ArtifactResolve>` message SHOULD be signed or otherwise authenticated and integrity  
2338 protected by the protocol binding used to deliver the message.

2339 This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and  
2340 adds the following element:

2341 <Artifact> [Required]

2342 The artifact value that the requester received and now wishes to translate into the protocol message it  
2343 represents. See [SAMLBind] for specific artifact format information.

2344 The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType**  
2345 complex type:

```
2346 <element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
2347 <complexType name="ArtifactResolveType">
2348   <complexContent>
2349     <extension base="samlp:RequestAbstractType">
2350       <sequence>
2351         <element ref="samlp:Artifact"/>
2352       </sequence>
2353     </extension>
2354   </complexContent>
2355 </complexType>
2356 <element name="Artifact" type="string"/>
```

### 2357 3.5.2 Element <ArtifactResponse>

2358 The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse>  
2359 message element. This element is of complex type **ArtifactResponseType**, which extends  
2360 **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol  
2361 message being returned. This wrapped message element can be a request or a response.

2362 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity  
2363 protected by the protocol binding used to deliver the message.

2364 The following schema fragment defines the <ArtifactResponse> element and its  
2365 **ArtifactResponseType** complex type:

```
2366 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2367 <complexType name="ArtifactResponseType">
2368   <complexContent>
2369     <extension base="samlp:StatusResponseType">
2370       <sequence>
2371         <any namespace="##any" processContents="lax" minOccurs="0"/>
2372       </sequence>
2373     </extension>
2374   </complexContent>
2375 </complexType>
```

### 2376 3.5.3 Processing Rules

2377 If the responder recognizes the artifact as valid, then it responds with the associated protocol message in  
2378 an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>  
2379 element with no embedded message. In both cases, the <Status> element MUST include a  
2380 <StatusCode> element with the code value urn:oasis:names:tc:SAML:2.0:status:Success. A  
2381 response message with no embedded message inside it is termed an empty response in the remainder of  
2382 this section.

2383 The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent  
2384 request with the same artifact by any requester results in an empty response as described above.

2385 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles, MAY  
2386 be intended for consumption by any party that receives it and can respond appropriately. In most other  
2387 cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST  
2388 be associated with the intended recipient of the message that the artifact represents. If the artifact issuer



2389 receives an <ArtifactResolve> message from a requester that cannot authenticate itself as the  
2390 original intended recipient, then the artifact issuer MUST return an empty response.

2391 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such  
2392 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and  
2393 return it in an <ArtifactResolve> message to the issuer.

2394 Note that the <ArtifactResponse> message's InResponseTo attribute MUST contain the value of  
2395 the corresponding <ArtifactResolve> message's ID attribute, but the embedded protocol message  
2396 will contain its own message identifier, and in the case of an embedded response, may contain a different  
2397 InResponseTo value that corresponds to the original request message to which the embedded message  
2398 is responding.

2399 All other processing rules associated with the underlying request and response messages MUST be  
2400 observed.

## 2401 **3.6 Name Identifier Management Protocol**

2402 After establishing a name identifier for a principal, an identity provider wishing to change the value and/or  
2403 format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will  
2404 no longer be used to refer to the principal, informs service providers of the change by sending them a  
2405 <ManageNameIDRequest> message.

2406 A service provider also uses this message to register or change the SPProvidedID value to be included  
2407 when the underlying name identifier is used to communicate with it, or to terminate the use of a name  
2408 identifier between itself and the identity provider.

2409 Note that this protocol is typically not used with "transient" name identifiers, since their value is not  
2410 intended to be managed on a long term basis.

### 2411 **3.6.1 Element <ManageNameIDRequest>**

2412 A provider sends a <ManageNameIDRequest> message to inform the recipient of a changed name  
2413 identifier or to indicate the termination of the use of a name identifier.

2414 The <ManageNameIDRequest> message SHOULD be signed or otherwise authenticated and integrity  
2415 protected by the protocol binding used to deliver the message.

2416 This message has the complex type **ManageNameIDRequestType**, which extends  
2417 **RequestAbstractType** and adds the following elements:

2418 <saml:NameID> or <saml:EncryptedID> [Required]  
2419 The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the  
2420 principal as currently recognized by the identity and service providers prior to this request. (For more  
2421 information on these elements, see Section 2.2.)

2422 <NewID> or <NewEncryptedID> or <Terminate> [Required]  
2423 The new identifier value (in plaintext or encrypted form) to be used when communicating with the  
2424 requesting provider concerning this principal, or an indication that the use of the old identifier has  
2425 been terminated. In the former case, if the requester is the service provider, the new identifier MUST  
2426 appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the  
2427 identity provider, the new value will appear in subsequent <NameID> elements as the element's  
2428 content.

2429 The following schema fragment defines the <ManageNameIDRequest> element and its  
2430 **ManageNameIDRequestType** complex type:

```

2431 <element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
2432 <complexType name="ManageNameIDRequestType">
2433   <complexContent>
2434     <extension base="samlp:RequestAbstractType">
2435       <sequence>
2436         <choice>
2437           <element ref="saml:NameID"/>
2438           <element ref="saml:EncryptedID"/>
2439         </choice>
2440         <choice>
2441           <element ref="samlp:NewID"/>
2442           <element ref="samlp:NewEncryptedID"/>
2443           <element ref="samlp:Terminate"/>
2444         </choice>
2445       </sequence>
2446     </extension>
2447   </complexContent>
2448 </complexType>
2449 <element name="NewID" type="string"/>
2450 <element name="NewEncryptedID" type="saml:EncryptedElementType"/>
2451 <element name="Terminate" type="samlp:TerminateType"/>
2452 <complexType name="TerminateType"/>

```

### 2453 3.6.2 Element <ManageNameIDResponse>

2454 The recipient of a <ManageNameIDRequest> message MUST respond with a  
 2455 <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional  
 2456 content.

2457 The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity  
 2458 protected by the protocol binding used to deliver the message.

2459 The following schema fragment defines the <ManageNameIDResponse> element:

```

2460 <element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>

```

### 2461 3.6.3 Processing Rules

2462 If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize,  
 2463 the responding provider MUST respond with an error <Status> and MAY respond with a second-level  
 2464 <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2465 If the <Terminate> element is included in the request, the requesting provider is indicating that (in the  
 2466 case of a service provider) it will no longer accept assertions from the identity provider or (in the case of  
 2467 an identity provider) it will no longer issue assertions to the service provider about the principal. The  
 2468 receiving provider can perform any maintenance with the knowledge that the relationship represented by  
 2469 the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for  
 2470 whom a relationship has been terminated.

2471 If the service provider requests that its identifier for the principal be changed by including a <NewID> (or  
 2472 <NewEncryptedID>) element, the identity provider MUST include the element's content as the  
 2473 SPProvidedID when subsequently communicating to the service provider regarding this principal.

2474 If the identity provider requests that its identifier for the principal be changed by including a <NewID> (or  
 2475 <NewEncryptedID>) element, the service provider MUST use the element's content as the  
 2476 <saml:NameID> element content when subsequently communicating with the identity provider regarding  
 2477 this principal.

2478 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the  
 2479 <EncryptedID> and <NewEncryptedID> elements).

2480 In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute  
2481 MUST contain the most recent name identifier information established between the providers for the  
2482 principal.

2483 In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
2484 `format:persistent` or `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`, the  
2485 `NameQualifier` attribute MUST contain the unique identifier of the identity provider or be omitted. If the  
2486 identifier was established between the identity provider and an affiliation group of which the service  
2487 provider is a member, then the `SPNameQualifier` attribute MUST contain the unique identifier of the  
2488 affiliation group. Otherwise, it MUST contain the unique identifier of the service provider or be omitted.

2489 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
2490 systems at both the requester and the responder. Implementations might wish to allow each party to  
2491 accept either identifier for some period of time following the successful completion of a name identifier  
2492 change. Not doing so could result in the inability of the principal to access resources.

2493 All other processing rules associated with the underlying request and response messages MUST be  
2494 observed.

## 2495 **3.7 Single Logout Protocol**

2496 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
2497 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
2498 when a principal logs out at a session participant or when the principal logs out directly at the  
2499 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for  
2500 the logout event can be indicated through the `Reason` attribute.

2501  
2502 The principal may have established authenticated sessions with both the session authority and individual  
2503 session participants, based on assertions containing authentication statements supplied by the session  
2504 authority.

2505  
2506 When the principal invokes the single logout process at a session participant, the session participant  
2507 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion  
2508 containing the authentication statement related to that session at the session participant.

2509  
2510 When either the principal invokes a logout at the session authority, or a session participant sends a logout  
2511 request to the session authority specifying that principal, the session authority MUST send a  
2512 `<LogoutRequest>` message to each session participant to which it provided assertions containing  
2513 authentication statements under its current session with the principal, with the exception of the session  
2514 participant that sent the `<LogoutRequest>` message to the session authority.

### 2515 **3.7.1 Element `<LogoutRequest>`**

2516 A session participant or session authority sends a `<LogoutRequest>` message to indicate that a session  
2517 has been terminated.

2518 The `<LogoutRequest>` message SHOULD be signed or otherwise authenticated and integrity protected  
2519 by the protocol binding used to deliver the message.

2520 This message has the complex type `LogoutRequestType`, which extends `RequestAbstractType` and  
2521 adds the following elements and attributes:

2522 `NotOnOrAfter` [Optional]

2523 The time at which the request expires, after which the recipient may discard the message. The time  
2524 value is encoded in UTC, as described in Section 1.3.3.

2525 Reason [Optional]

2526 An indication of the reason for the logout, in the form of a URI reference.

2527 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2528 The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as  
2529 currently recognized by the identity and service providers prior to this request. (For more information  
2530 on this element, see Section 2.2.)

2531 <SessionIndex> [Optional]

2532 The identifier that indexes this session at the message recipient.

2533 The following schema fragment defines the <LogoutRequest> element and associated  
2534 **LogoutRequestType** complex type:

```
2535 <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2536 <complexType name="LogoutRequestType">
2537 <complexContent>
2538 <extension base="samlp:RequestAbstractType">
2539 <sequence>
2540 <choice>
2541 <element ref="saml:BaseID"/>
2542 <element ref="saml:NameID"/>
2543 <element ref="saml:EncryptedID"/>
2544 </choice>
2545 <element ref="samlp:SessionIndex" minOccurs="0"
2546 maxOccurs="unbounded"/>
2547 </sequence>
2548 <attribute name="Reason" type="string" use="optional"/>
2549 <attribute name="NotOnOrAfter" type="dateTime"
2550 use="optional"/>
2551 </extension>
2552 </complexContent>
2553 </complexType>
2554 <element name="SessionIndex" type="string"/>
```

### 2555 3.7.2 Element <LogoutResponse>

2556 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of  
2557 type **StatusResponseType**, with no additional content specified.

2558 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity  
2559 protected by the protocol binding used to deliver the message.

2560 The following schema fragment defines the <LogoutResponse> element:

```
2561 <element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

### 2562 3.7.3 Processing Rules

2563 The message sender MAY use the Reason attribute to indicate the reason for sending the  
2564 <LogoutRequest>. The following values are defined by this specification for use by all message  
2565 senders; other values MAY be agreed on between participants:

2566 urn:oasis:names:tc:SAML:2.0:logout:user

2567 Specifies that the message is being sent because the principal wishes to terminate the indicated  
2568 session.

2569 urn:oasis:names:tc:SAML:2.0:logout:admin

2570 Specifies that the message is being sent because an administrator wishes to terminate the indicated

2571 session for that principal.

2572 All other processing rules associated with the underlying request and response messages MUST be  
2573 observed.

2574 Additional processing rules are provided in the following sections.

### 2575 **3.7.3.1 Session Participant Rules**

2576 When a session participant receives a `<LogoutRequest>` message, the session participant MUST  
2577 authenticate the message. If the sender is the authority that provided an assertion containing an  
2578 authentication statement linked to the principal's current session, the session participant MUST invalidate  
2579 the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or  
2580 `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no  
2581 `<SessionIndex>` elements are supplied, then all sessions associated with the principal MUST be  
2582 invalidated.

2583  
2584 The session participant MUST apply the logout request message to any assertion that meets the following  
2585 conditions, even if the assertion arrives after the logout request:

- 2586 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or  
2587 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in section 3.3.4.
- 2588 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of  
2589 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no  
2590 `<SessionIndex>` elements.
- 2591 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself  
2592 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject  
2593 confirmation data).
- 2594 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on  
2595 the message).

2596 **Note:** This rule is intended to prevent a situation in which a session participant receives a  
2597 logout request targeted at a single, or multiple, assertion(s) (as identified by the  
2598 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid -  
2599 assertion(s) targeted by the logout request. It should honor the logout request until the  
2600 logout request itself may be discarded (the `NotOnOrAfter` value on the request has  
2601 been exceeded) or the assertion targeted by the logout request has been received and  
2602 has been handled appropriately.

### 2603 **3.7.3.2 Session Authority Rules**

2604 When a session authority receives a `<LogoutRequest>` message, the session authority MUST  
2605 authenticate the sender. If the sender is a session participant to which the session authority provided an  
2606 assertion containing an authentication statement for the current session, then the session authority  
2607 SHOULD do the following in the specified order:

- 2608 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session  
2609 authority proxied the principal's authentication, unless the second authority is the originator of the  
2610 `<LogoutRequest>`.
- 2611 • Send a `<LogoutRequest>` message to each session participant for which the session authority  
2612 provided assertions in the current session, *other than* the originator of a current  
2613 `<LogoutRequest>`.

2614 • Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`,  
2615 or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout  
2616 request message.

2617 If an error occurs during this further processing of the logout (for example, other session participants may  
2618 not all implement the particular single logout protocol binding used by the requesting session participant),  
2619 then the session authority MUST respond to the original requester with a `<LogoutResponse>` message,  
2620 indicating the status of the logout request. The value  
2621 `urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding` is provided for a second-level  
2622 `<StatusCode>`, indicating that the originating session participant should retry the `<LogoutRequest>`  
2623 using a different protocol binding.

2624 Note that this protocol does not permit "partial-success"; any error during the propagation of logout  
2625 requests by the session authority MUST result in the failure of the overall logout process, and the session  
2626 authority MUST return a `<LogoutResponse>` message containing an appropriate status code to the  
2627 originating session participant, if any. It need not communicate further with any session participants that  
2628 have already been successfully informed of the logout.

2629 Note that a session authority MAY initiate a logout for reasons other than having received a  
2630 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2631 • If some timeout period was agreed out-of-band with an individual session participant, the session  
2632 authority MAY send a `<LogoutRequest>` to that individual participant alone.
- 2633 • An agreed global timeout period has been exceeded.
- 2634 • The principal or some other trusted entity has requested logout of the principal directly at the session  
2635 authority.
- 2636 • The session authority has determined that the principal's credentials may have been compromised.

2637 When constructing a logout request message, the session authority MUST set the value of the  
2638 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message,  
2639 after which the logout request may be discarded by the recipient. This value SHOULD be set to a time  
2640 value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most  
2641 recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout  
2642 request).

2643 In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also  
2644 available for use by the session authority only:

2645 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2646 Specifies that the message is being sent because of the global session timeout interval period  
2647 being exceeded.

2648 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

2649 Specifies that the message is being sent because a timeout interval period agreed between a  
2650 participant and the session authority has been exceeded.

## 2651 **3.8 Name Identifier Mapping Protocol**

2652 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
2653 identifier for the same principal in a particular format or federation namespace, it can send a request to  
2654 the identity provider using this protocol.

2655 For example, a service provider that wishes to communicate with another service provider with whom it  
2656 does not share an identifier for the principal can use an identity provider that shares an identifier for the



2657 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,  
2658 with which it can communicate with the second service provider.

2659 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a  
2660 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

### 2661 **3.8.1 Element <NameIDMappingRequest>**

2662 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
2663 <NameIDMappingRequest> message. This message has the complex type  
2664 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2665 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2666 The identifier and associated descriptive data that specify the principal as currently recognized by the  
2667 requester and the responder. (For more information on this element, see Section 2.2.)

2668 <NameIDPolicy> [Required]

2669 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2670 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2671 binding used to deliver the message.

2672 The following schema fragment defines the <NameIDMappingRequest> element and its  
2673 **NameIDMappingRequestType** complex type:

```
2674 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2675 <complexType name="NameIDMappingRequestType">
2676   <complexContent>
2677     <extension base="samlp:RequestAbstractType">
2678       <sequence>
2679         <choice>
2680           <element ref="saml:BaseID"/>
2681           <element ref="saml:NameID"/>
2682           <element ref="saml:EncryptedID"/>
2683         </choice>
2684         <element ref="samlp:NameIDPolicy"/>
2685       </sequence>
2686     </extension>
2687   </complexContent>
2688 </complexType>
```

### 2689 **3.8.2 Element <NameIDMappingResponse>**

2690 The recipient of a <NameIDMappingRequest> message MUST respond with a  
2691 <NameIDMappingResponse> message. This message has the complex type  
2692 **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

2693 <saml:NameID> or <saml:EncryptedID> [Required]

2694 The identifier and associated attributes that specify the principal in the manner requested, usually in  
2695 encrypted form. (For more information on this element, see Section 2.2.)

2696 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2697 binding used to deliver the message.

2698 The following schema fragment defines the <NameIDMappingResponse> element and its  
2699 **NameIDMappingResponseType** complex type:

```
2700 <element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
2701 <complexType name="NameIDMappingResponseType">
```

```
2702     <complexContent>
2703         <extension base="samlp:StatusResponseType">
2704             <choice>
2705                 <element ref="saml:NameID"/>
2706                 <element ref="saml:EncryptedID"/>
2707             </choice>
2708         </extension>
2709     </complexContent>
2710 </complexType>
```

### 2711 **3.8.3 Processing Rules**

2712 If the responder does not recognize the principal identified in the request, it MAY respond with an error  
2713 <Status> containing a second-level <StatusCode> of  
2714 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2715 At the responder's discretion, the  
2716 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to  
2717 indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

2718 All other processing rules associated with the underlying request and response messages MUST be  
2719 observed.



---

## 2720 4 SAML Versioning

2721 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
2722 sections, along with processing rules for detecting and handling version differences. Also included are  
2723 guidelines on when and why specific version information is expected to change in future revisions of the  
2724 specification.

2725 When version information is expressed as both a Major and Minor version, it is expressed in the form  
2726 *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version number *Major<sub>A</sub>.Minor<sub>A</sub>* if and  
2727 only if:

2728  $Major_B > Major_A$  OR ( (  $Major_B = Major_A$  ) AND  $Minor_B > Minor_A$  )

### 2729 4.1 SAML Specification Set Version

2730 Each release of the SAML specification set will contain a major and minor version designation describing  
2731 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
2732 content and filenames of published materials, including the specification set documents and XML schema  
2733 documents. There are no normative processing rules surrounding specification set versioning, since it  
2734 merely encompasses the collective release of normative specification documents which themselves  
2735 contain processing rules.

2736 The overall size and scope of changes to the specification set documents will informally dictate whether a  
2737 set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
2738 compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain  
2739 valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major  
2740 revision.

#### 2741 4.1.1 Schema Version

2742 As a non-normative documentation mechanism, any XML schema documents published as part of the  
2743 specification set will contain a `version` attribute on the `<xsi:schema>` element whose value is in the  
2744 form *Major.Minor*, reflecting the specification set version in which it has been published. Validating  
2745 implementations MAY use the attribute as a means of distinguishing which version of a schema is being  
2746 used to validate messages, or to support multiple versions of the same logical schema.

#### 2747 4.1.2 SAML Assertion Version

2748 The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the  
2749 assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed  
2750 so as to document the syntax, semantics, and processing rules of the assertions of the same version.  
2751 That is, specification set version 1.0 describes assertion version 1.0, and so on.

2752 There is explicitly NO relationship between the assertion version and the target XML namespace specified  
2753 for the schema definitions for that assertion version.

2754 The following processing rules apply:

- 2755 • A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion  
2756 version number not supported by the authority.
- 2757 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
2758 supported by the relying party.
- 2759 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
2760 number is higher than the minor assertion version number supported by the relying party. However,

2761 all assertions that share a major assertion version number MUST share the same general  
2762 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For  
2763 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the  
2764 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the  
2765 likely effects of schema evolution.)

### 2766 4.1.3 SAML Protocol Version

2767 The various SAML protocols' request and response elements contain an attribute for expressing the major  
2768 and minor version of the request or response message using a string of the form *Major.Minor*. Each  
2769 version of the SAML specification set will be construed so as to document the syntax, semantics, and  
2770 processing rules of the protocol messages of the same version. That is, specification set version 1.0  
2771 describes request and response version V1.0, and so on.

2772 There is explicitly NO relationship between the protocol version and the target XML namespace specified  
2773 for the schema definitions for that protocol version.

2774 The version numbers used in SAML protocol request and response elements will match for any particular  
2775 revision of the SAML specification set.

#### 2776 4.1.3.1 Request Version

2777 The following processing rules apply to requests:

- 2778 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
2779 SAML requester and the SAML responder.
- 2780 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD  
2781 assume that the responder supports requests with the highest request version supported by the  
2782 requester.
- 2783 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version  
2784 number matching a response version number that the requester does not support.
- 2785 • A SAML responder MUST reject any request with a major request version number not supported by  
2786 the responder.
- 2787 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
2788 higher than the highest supported request version that it supports. However, all requests that share  
2789 a major request version number MUST share the same general processing rules and semantics,  
2790 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the  
2791 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill  
2792 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

#### 2793 4.1.3.2 Response Version

2794 The following processing rules apply to responses:

- 2795 • A SAML responder MUST NOT issue a response message with a response version number higher  
2796 than the request version number of the corresponding request message.
- 2797 • A SAML responder MUST NOT issue a response message with a major response version number  
2798 lower than the major request version number of the corresponding request message except to  
2799 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2800 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a  
2801 top-level `<StatusCode>` value of  
2802 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting  
2803 one of the following second-level values:

2804 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,  
2805 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or  
2806 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

### 2807 **4.1.3.3 Permissible Version Combinations**

2808 Assertions of a particular major version appear only in response messages of the same major version, as  
2809 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For  
2810 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1  
2811 response message, if the appropriate assertion schema is referenced during namespace importation. But  
2812 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major  
2813 versions.

## 2814 **4.2 SAML Namespace Version**

2815 XML schema documents published as part of the specification set contain one or more target  
2816 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct  
2817 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that  
2818 part of the specification.

2819 The namespace URI references defined by the specification set will generally contain version information  
2820 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond  
2821 to the major and minor version of the specification set in which the namespace is first introduced and  
2822 defined. This information is not typically consumed by an XML processor, which treats the namespace  
2823 opaquely, but is intended to communicate the relationship between the specification set and the  
2824 namespaces it defines. (This pattern is also followed by the SAML-defined URI-based identifiers that are  
2825 listed in Section 8.)

2826 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
2827 defined by a major revision of the specification set to remain valid and stable across minor revisions of the  
2828 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but  
2829 this is expected to be rare. In such cases, the older namespaces and their associated definitions should  
2830 be expected to remain valid until a major specification set revision.

### 2831 **4.2.1 Schema Evolution**

2832 In general, maintaining namespace stability while adding or changing the content of a schema are  
2833 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
2834 older implementations will react to any given change, making forward compatibility difficult to achieve.  
2835 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace  
2836 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),  
2837 implementations should expect forward-compatible schema changes in minor revisions, allowing new  
2838 messages to validate against older schemas.

2839 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
2840 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
2841 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such  
2842 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples  
2843 include new query, statement, or condition types.

2844

## 5 SAML and XML Signature Syntax and Processing

2845 SAML assertions and SAML protocol request and response messages may be signed, with the following  
2846 benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the  
2847 asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-  
2848 private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the  
2849 message originator supports message integrity, authentication of message origin to a destination, and, if  
2850 the signature is based on the originator's public-private key pair, non-repudiation of origin.

2851 A digital signature is not always required in SAML. For example, in some circumstances, signatures may  
2852 be "inherited," such as when an unsigned assertion gains protection from a signature on the containing  
2853 protocol response message. "Inherited" signatures should be used with care when the contained object  
2854 (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context  
2855 must be retained to allow validation, exposing the XML content and adding potentially unnecessary  
2856 overhead. As another example, the SAML relying party or SAML requester may have obtained an  
2857 assertion or protocol message from the SAML asserting party or SAML responder directly (with no  
2858 intermediaries) through a secure channel, with the asserting party or SAML responder having  
2859 authenticated to the relying party or SAML responder by some means other than a digital signature.

2860 Many different techniques are available for "direct" authentication and secure channel establishment  
2861 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, and so on. In  
2862 addition, the applicable security requirements depend on the communicating applications and the nature  
2863 of the assertion or message transported. It is RECOMMENDED that, in all other contexts, digital  
2864 signatures be used for assertions and request and response messages. Specifically:

- 2865 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting  
2866 party SHOULD be signed by the SAML asserting party.
- 2867 • A SAML protocol message arriving at a destination from an entity other than the originating sender  
2868 SHOULD be signed by the sender.

2869 Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that  
2870 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are  
2871 intended to be the primary SAML signature mechanism, but this specification attempts to ensure  
2872 compatibility with profiles that may require other mechanisms.

2873 Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be  
2874 enveloped.

### 2875 5.1 Signing Assertions

2876 All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as  
2877 described in Section 2.

### 2878 5.2 Request/Response Signing

2879 All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected  
2880 in the schema as described in Section 3.

### 2881 5.3 Signature Inheritance

2882 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
2883 or a request or response, which may be signed. When a SAML assertion does not contain a  
2884 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
2885 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,

2886 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
2887 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
2888 and signature options.

2889 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
2890 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY  
2891 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
2892 information from the surrounding context, but no such inheritance should be inferred unless specifically  
2893 identified by the profile.

## 2894 **5.4 XML Signature Profile**

2895 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
2896 and many choices. This section details constraints on these facilities so that SAML processors do not  
2897 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
2898 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**  
2899 attribute on `<Assertion>` and the various request and response elements. These attributes are  
2900 collectively referred to in this section as the identifier attributes.

2901 Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML  
2902 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to  
2903 SAML content are free to define other approaches.

### 2904 **5.4.1 Signing Formats and Algorithms**

2905 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
2906 detached.

2907 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
2908 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
2909 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2910 **5.4.2 References**

2911 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of  
2912 the assertion or protocol message being signed. The assertion's or protocol message's root element may  
2913 or may not be the root element of the actual XML document containing the signed assertion or protocol  
2914 message (e.g., it might be contained within a SOAP envelope).

2915 Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the **ID**  
2916 attribute value of the root element of the assertion or protocol message being signed. For example, if the  
2917 **ID** attribute value is "foo", then the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

### 2918 **5.4.3 Canonicalization Method**

2919 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,  
2920 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a  
2921 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over  
2922 SAML messages embedded in an XML context can be verified independent of that context.

### 2923 **5.4.4 Transforms**

2924 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
2925 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive

2926 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2927 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2928 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2929 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can  
2930 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2931 applying the transforms manually to the content and reverifying the result as consisting of the same SAML  
2932 message.

## 2933 5.4.5 KeyInfo

2934 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of  
2935 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be  
2936 absent.

## 2937 5.4.6 Example

2938 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2939 added for readability; the signatures are not valid and cannot be successfully verified.

```
2940 <Response
2941   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
2942   ID="_c7055387-af61-4fce-8b98-e2927324b306"
2943   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
2944   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2945   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
2946   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2947     <ds:SignedInfo>
2948       <ds:CanonicalizationMethod
2949         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2950       <ds:SignatureMethod
2951         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2952       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
2953         <ds:Transforms>
2954           <ds:Transform
2955             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
2956 signature" />
2957           <ds:Transform
2958             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2959             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
2960               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2961           </ds:Transform>
2962         </ds:Transforms>
2963         <ds:DigestMethod
2964           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2965         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2966       </ds:Reference>
2967     </ds:SignedInfo>
2968     <ds:SignatureValue>
2969       x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwCRKrtwPS6vdVNCcY5rHaFPYwKf+5
2970       ELYcPzx+pXlh43SmwviCqXRjRtMANWbHLhWAptaKlywS7gFgsD0lqjyen3CP+m3D
2971       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
2972     </ds:SignatureValue>
2973     <ds:KeyInfo>
2974       <ds:X509Data>
2975         <ds:X509Certificate>
2976         MIICyjcCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
2977         MRlWEAyDVQQIEw1XaXNjb25zaW4xZDA0bGhWApTAklywS7gFgsD0lqjyen3CP+m3D
2978         F1VuaXZlcjNpdHkgb2YgV2lzyY29uc2luMSswKQYDVQQLExJEaXZpc2lubiBvZiBJ
2979         bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIiwvYDVQQDExxIRVBLSSBTZXJ2ZXIgc00Eg
2980         LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVoWYysx
2981         CzAJBgNVBAYTA1VTMRewDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMjQW5uIEFy
```



```

2982 Ym9yMQ4wDAYDVQKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
2983 dTEncUCGSqGSib3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldIuZWR1MIGfMA0G
2984 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
2985 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
2986 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
2987 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
2988 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2989 qgi7lFV6MDkHmTvtQtBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
2990 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==
2991 </ds:X509Certificate>
2992 </ds:X509Data>
2993 </ds:KeyInfo>
2994 </ds:Signature>
2995 <Status>
2996 <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
2997 </Status>
2998 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2999 IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3000 xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3001 <Issuer>https://www.opensaml.org/IDP</Issuer>
3002 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3003 <ds:SignedInfo>
3004 <ds:CanonicalizationMethod
3005 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
3006 <ds:SignatureMethod
3007 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
3008 <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3009 <ds:Transforms>
3010 <ds:Transform
3011 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3012 signature" />
3013 <ds:Transform
3014 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3015 <InclusiveNamespaces
3016 PrefixList="#default saml ds xs xsi"
3017 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
3018 </ds:Transform>
3019 </ds:Transforms>
3020 <ds:DigestMethod
3021 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
3022 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3023 </ds:Reference>
3024 </ds:SignedInfo>
3025 <ds:SignatureValue>
3026 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3027 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
3028 MwuL/cBUj20tBZOQMFN7jQ9YB7klIz3RqVL+wNmeWI4=
3029 </ds:SignatureValue>
3030 <ds:KeyInfo>
3031 <ds:X509Data>
3032 <ds:X509Certificate>
3033 MIICyCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwwgAKCzAJBgNVBAYTA1VT
3034 MRlWAEYDVQQIEwI1XaXNjb25zaW4xEDA0BgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3035 Fl1VuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
3036 bmZvcmlhdGlvbiBUZWNobm9sb2d25MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgc0Eg
3037 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkxNDA3Mjc1MVoWogYsX
3038 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbGJESMBAGA1UEBxMJQW5uIEFy
3039 Ym9yMQ4wDAYDVQKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3040 dTEncUCGSqGSib3DQEJARYYcm9vdEBzaGliMS5pbnRlcm5ldIuZWR1MIGfMA0G
3041 CSqGSib3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
3042 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBiaOAPSZB113R6+KYiE7x4XAWIrcP+
3043 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
3044 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3045 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3046 qgi7lFV6MDkHmTvtQtBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3047 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==

```

```
3048         </ds:X509Certificate>
3049         </ds:X509Data>
3050     </ds:KeyInfo>
3051 </ds:Signature>
3052 <Subject>
3053     <NameID
3054         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3055         scott@example.org
3056     </NameID>
3057     <SubjectConfirmation
3058         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3059 </Subject>
3060 <Conditions NotBefore="2003-04-17T00:46:02Z"
3061     NotOnOrAfter="2003-04-17T00:51:02Z">
3062     <AudienceRestriction>
3063         <Audience>http://www.opensaml.org/SP</Audience>
3064     </AudienceRestriction>
3065 </Conditions>
3066 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3067     <AuthnContext>
3068         <AuthnContextClassRef>
3069             urn:oasis:names:tc:SAML:2.0:ac:classes>Password
3070         </AuthnContextClassRef>
3071     </AuthnContext>
3072 </AuthnStatement>
3073 </Assertion>
3074 </Response>
```



3075

## 6 SAML and XML Encryption Syntax and Processing

3076 Encryption is used as the means to implement confidentiality. The most common motives for  
3077 confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for  
3078 competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness  
3079 of some other security mechanism. For example, a secret password or key may be encrypted.

3080 Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- 3081 • Communications confidentiality may be provided by mechanisms associated with a particular  
3082 binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS or  
3083 SOAP Message Security mechanisms for confidentiality.
- 3084 • A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>`  
3085 element within `<SubjectConfirmationData>`, which permits keys or other secrets to be  
3086 encrypted.
- 3087 • An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- 3088 • The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- 3089 • An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

### 6.1 General Considerations

3091 Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use  
3092 of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace  
3093 the cleartext information in the same location within the XML instance. The `<EncryptedData>` element's  
3094 `Type` attribute SHOULD be used and, if it is present, MUST have the value  
3095 `http://www.w3.org/2001/04/xmlenc#Element`.

3096 Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The  
3097 SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

### 6.2 Combining Signatures and Encryption

3099 Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and  
3100 encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in  
3101 the reverse order that signing and encryption were performed.

- 3102 • When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and  
3103 placed within the `<Assertion>` element before the element is encrypted.
- 3104 • When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be  
3105 performed first and then the signature calculated over the assertion or message containing the  
3106 encrypted element.

---

## 3107 7 SAML Extensibility

3108 SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas.  
3109 An example of an application that extends SAML assertions is the Liberty Protocols and Schema  
3110 Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions  
3111 and protocols.

3112 See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can  
3113 be combined with extensions to put the SAML framework to new uses.

### 3114 7.1 Schema Extension

3115 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML  
3116 elements can serve as the head element of a substitution group. However, SAML types are not defined as  
3117 *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that  
3118 extensions are typically defined only as types rather than elements, and are included in SAML instances  
3119 by means of an  `xsi:type`  attribute.

3120 The following sections discuss only elements and types that have been specifically designed to support  
3121 extensibility.

#### 3122 7.1.1 Assertion Schema Extension

3123 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
3124 statements it contains, if the extension mechanism is used for either part.

3125 The following elements are intended specifically for use as extension points in an extension schema; their  
3126 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 3127 • `<BaseID>` and **BaseIDAbstractType**
- 3128 • `<Condition>` and **ConditionAbstractType**
- 3129 • `<Statement>` and **StatementAbstractType**

3130 The following constructs that are directly usable as part of SAML are particularly interesting targets for  
3131 extension:

- 3132 • `<AuthnStatement>` and **AuthnStatementType**
- 3133 • `<AttributeStatement>` and **AttributeStatementType**
- 3134 • `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
- 3135 • `<AudienceRestriction>` and **AudienceRestrictionType**
- 3136 • `<ProxyRestriction>` and **ProxyRestrictionType**
- 3137 • `<OneTimeUse>` and **OneTimeUseType**

#### 3138 7.1.2 Protocol Schema Extension

3139 The following SAML protocol elements are intended specifically for use as extension points in an  
3140 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived  
3141 type:

- 3142 • `<Request>` and **RequestAbstractType**

3143 • <SubjectQuery> and **SubjectQueryAbstractType**

3144 The following constructs that are directly usable as part of SAML are particularly interesting targets for  
3145 extension:

3146 • <AuthnQuery> and **AuthnQueryType**

3147 • <AuthzDecisionQuery> and **AuthzDecisionQueryType**

3148 • <AttributeQuery> and **AttributeQueryType**

3149 • **StatusResponseType**

## 3150 7.2 Schema Wildcard Extension Points

3151 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes  
3152 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension  
3153 schema.

### 3154 7.2.1 Assertion Extension Points

3155 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

3156 • <SubjectConfirmationData>: Uses **xs:anyType**, which allows any sub-elements and  
3157 attributes.

3158 • <AuthnContextDecl>: Uses **xs:anyType**, which allows any sub-elements and attributes.

3159 • <AttributeValue>: Uses **xs:anyType**, which allows any sub-elements and attributes.

3160 • <Advice> and **AdviceType**: In addition to SAML-native elements, allows elements from other  
3161 namespaces with lax schema validation processing.

3162 The following constructs in the assertion schema allow arbitrary global attributes:

3163 • <Attribute> and **AttributeType**

### 3164 7.2.2 Protocol Extension Points

3165 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

3166 • <Extensions> and **ExtensionsType**: Allows elements from other namespaces with lax schema  
3167 validation processing.

3168 • <StatusDetail> and **StatusDetailType**: Allows elements from other namespaces with lax  
3169 schema validation processing.

3170 • <ArtifactResponse> and **ArtifactResponseType**: Allows elements from any namespaces with  
3171 lax schema validation processing. (It is specifically intended to carry a SAML request or response  
3172 message element, however.)

## 3173 7.3 Identifier Extension

3174 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier  
3175 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.  
3176 However, it is always possible to define additional URI-based identifiers for these purposes. It is  
3177 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should

3178 the meaning of a given URI used as such an identifier significantly change, or be used to mean two  
3179 different things.

---

## 3180 8 SAML-Defined Identifiers

3181 The following sections define URI-based identifiers for common resource access actions, subject name  
3182 identifier formats, and attribute name formats.

3183 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of  
3184 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
3185 have one of the following stems, according to the specification set version in which they were first  
3186 introduced:

```
3187 urn:oasis:names:tc:SAML:1.0:  
3188 urn:oasis:names:tc:SAML:1.1:  
3189 urn:oasis:names:tc:SAML:2.0:
```

### 3190 8.1 Action Namespace Identifiers

3191 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to  
3192 common sets of actions to perform on resources.

#### 3193 8.1.1 Read/Write/Execute/Delete/Control

3194 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

3195 Defined actions:

3196 `Read Write Execute Delete Control`

3197 These actions are interpreted as follows:

3198 `Read`

3199 `The subject may read the resource.`

3200 `Write`

3201 `The subject may modify the resource.`

3202 `Execute`

3203 `The subject may execute the resource.`

3204 `Delete`

3205 `The subject may delete the resource.`

3206 `Control`

3207 `The subject may specify the access control policy for the resource.`

#### 3208 8.1.2 Read/Write/Execute/Delete/Control with Negation

3209 **URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

3210 Defined actions:

3211 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

3212 The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions  
3213 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated  
3214 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is  
3215 affirmatively denied read permission.

3216 A SAML authority MUST NOT authorize both an action and its negated form.

### 3217 **8.1.3 Get/Head/Put/Post**

3218 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3219 Defined actions:

3220 GET HEAD PUT POST

3221 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
3222 the GET action on a resource is authorized to retrieve it.

3223 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST  
3224 actions to the write permission. The correspondence is not exact however since an HTTP GET operation  
3225 may cause data to be modified and a POST operation may cause modification to a resource other than  
3226 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

### 3227 **8.1.4 UNIX File Permissions**

3228 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3229 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3230 The action string is a four-digit numeric code:

3231 *extended user group world*

3232 Where the *extended* access permission has the value

3233 +2 if sgid is set

3234 +4 if suid is set

3235 The *user group* and *world* access permissions have the value

3236 +1 if execute permission is granted

3237 +2 if write permission is granted

3238 +4 if read permission is granted

3239 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read  
3240 and execute; and world read.

## 3241 **8.2 Attribute Name Format Identifiers**

3242 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex  
3243 type to refer to the classification of the attribute name for purposes of interpreting the name.

### 3244 **8.2.1 Unspecified**

3245 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3246 The interpretation of the attribute name is left to individual implementations.

## 3247 **8.2.2 URI Reference**

3248 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

3249 The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML  
3250 [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-  
3251 specific. See [SAMLProf] for attribute profiles that make use of this identifier.

## 3252 **8.2.3 Basic**

3253 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

3254 The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to  
3255 the primitive type **xs:Name** as defined in [Schema2] §3.3.6. See [SAMLProf] for attribute profiles that  
3256 make use of this identifier.

## 3257 **8.3 Name Identifier Format Identifiers**

3258 The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or  
3259 `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the  
3260 associated processing rules, if any.

3261 **Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for  
3262 SAML V2.0.

### 3263 **8.3.1 Unspecified**

3264 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3265 The interpretation of the content of the element is left to individual implementations.

### 3266 **8.3.2 Email Address**

3267 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3268 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as  
3269 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that  
3270 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in  
3271 parentheses) after it, and is not surrounded by "<" and ">".

### 3272 **8.3.3 X.509 Subject Name**

3273 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3274 Indicates that the content of the element is in the form specified for the contents of the  
3275 `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors  
3276 should note that the XML Signature specification specifies encoding rules for X.509 subject names that  
3277 differ from the rules given in IETF RFC 2253 [RFC 2253].

### 3278 **8.3.4 Windows Domain Qualified Name**

3279 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName



3280 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
3281 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator  
3282 MAY be omitted.

### 3283 **8.3.5 Kerberos Principal Name**

3284 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3285 Indicates that the content of the element is in the form of a Kerberos principal name using the format  
3286 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and  
3287 realm are described in [RFC 1510].

### 3288 **8.3.6 Entity Identifier**

3289 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3290 Indicates that the content of the element is the identifier of an entity that provides SAML-based services  
3291 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service  
3292 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to  
3293 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make  
3294 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be  
3295 used in other elements and attributes whose purpose is to identify a system entity in various protocol  
3296 exchanges.

3297 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is  
3298 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3299 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

### 3300 **8.3.7 Persistent Identifier**

3301 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3302 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to  
3303 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers  
3304 generated by identity providers MUST be constructed using pseudo-random values that have no  
3305 discernible correspondence with the subject's actual identifier (for example, username). The intent is to  
3306 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.  
3307 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3308 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity  
3309 provider that generated the identifier (see section 8.3.6). It MAY be omitted if the value can be derived  
3310 from the context of the message containing the element, such as the issuer of a protocol message or an  
3311 assertion containing the identifier in its subject. Note that a different system entity might later issue its own  
3312 protocol message or assertion containing the identifier; the `NameQualifier` does not change in this case,  
3313 but MUST continue to identify the entity that originally created the identifier (and MUST NOT be omitted in  
3314 such a case).

3315 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service  
3316 provider or affiliation of providers for whom the identifier was generated (see section 8.3.6). It MAY be  
3317 omitted if the element is contained in a message intended only for consumption directly by the service  
3318 provider, and the value would be the name of that service provider.

3319 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most  
3320 recently set by the service provider or affiliation, if any (see section 3.6). If no such identifier has been  
3321 established, then the attribute MUST be omitted.

3322 Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared  
3323 in clear text with providers other than the providers that have established the shared identifier.  
3324 Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and  
3325 protections. Deployments without such requirements are free to use other kinds of identifiers in their  
3326 SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

3327 Note also that while persistent identifiers are typically used to reflect an account linking relationship  
3328 between a pair of providers, a service provider is not obligated to recognize or make use of the long term  
3329 nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly  
3330 different and does not affect the behavior of the identity provider or any processing rules specific to  
3331 persistent identifiers in the protocols defined in this specification.

### 3332 **8.3.8 Transient Identifier**

3333 **URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

3334 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated  
3335 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in  
3336 accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of  
3337 256 characters.

3338 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier  
3339 represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in  
3340 accordance with the rules specified in Section 8.3.7.

## 3341 **8.4 Consent Identifiers**

3342 The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and  
3343 **StatusResponseType** complex types to communicate whether a principal gave consent, and under what  
3344 conditions, for the message.

### 3345 **8.4.1 Unspecified**

3346 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

3347 No claim as to principal consent is being made.

### 3348 **8.4.2 Obtained**

3349 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

3350 Indicates that a principal's consent has been obtained by the issuer of the message.

### 3351 **8.4.3 Prior**

3352 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

3353 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to  
3354 the action that initiated the message.

### 3355 **8.4.4 Implicit**

3356 **URI:** `urn:oasis:names:tc:SAML:2.0:consent:current-implicit`

3357 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the  
3358 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically  
3359 more proximal to the action in time and presentation than prior consent, such as part of a session of  
3360 activities.

#### 3361 **8.4.5 Explicit**

3362 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3363 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the  
3364 action that initiated the message.

#### 3365 **8.4.6 Unavailable**

3366 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3367 Indicates that the issuer of the message did not obtain consent.

#### 3368 **8.4.7 Inapplicable**

3369 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3370 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

3371

## 9 References

3372 The following works are cited in the body of this specification.

### 3373 9.1 Normative References

- 3374       **[Excl-C14N]**       J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web  
3375                               Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3376       **[Schema1]**        H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
3377                               Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.  
3378                               Note that this specification normatively references [Schema2], listed below.
- 3379       **[Schema2]**        P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
3380                               Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3381       **[XML]**            T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World  
3382                               Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3383       **[XMLEnc]**         D. Eastlake et al., XML Encryption Syntax and Processing,  
3384                               <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web  
3385                               Consortium. Note that this specification normatively references [XMLEnc-XSD],  
3386                               listed below.
- 3387       **[XMLEnc-XSD]**     XML Encryption Schema. World Wide Web Consortium.  
3388                               <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3389       **[XMLNS]**         T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January  
3390                               1999. <http://www.w3.org/TR/REC-xml-names>.
- 3391       **[XMLSig]**         D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web  
3392                               Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this  
3393                               specification normatively references [XMLSig-XSD], listed below.
- 3394       **[XMLSig-XSD]**     XML Signature Schema. World Wide Web Consortium.  
3395                               [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)  
3396                               [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

### 3397 9.2 Non-Normative References

- 3398       **[LibertyProt]**     J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty  
3399                               Alliance Project, January 2003,  
3400                               [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)  
3401                               [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3402       **[Needham78]**     R. Needham et al. *Using Encryption for Authentication in Large Networks of*  
3403                               *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December  
3404                               1978.
- 3405       **[PGP]**            Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*.  
3406                               IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3407       **[PKIX]**            R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*  
3408                               *Certificate and CRL Profile*. IETF RFC 2459, January 1999.  
3409                               <http://www.ietf.org/rfc/rfc2459.txt>.
- 3410       **[RFC 1510]**         J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF  
3411                               RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3412       **[RFC 2119]**         S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF  
3413                               RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

3414 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.  
3415 <http://www.ietf.org/rfc/rfc2246.txt>.

3416 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*  
3417 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.  
3418 <http://www.ietf.org/rfc/rfc2253.txt>.

3419 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF  
3420 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3421 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.  
3422 <http://www.ietf.org/rfc/rfc2630.txt>.

3423 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.  
3424 <http://www.ietf.org/rfc/rfc2822.txt>.

3425 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,  
3426 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3427 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF  
3428 RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3429 [RFC 3513] R. Hinden, S. Deering, *Internet Protocol Version 6 (IPv6) Addressing Architecture*.  
3430 IETF RFC 3513, April 2003. <http://www.ietf.org/rfc/rfc3513.txt>.

3431 [SAMLAuthnCxt] J. Kemp et al., *Authentication Context for the OASIS Security Assertion Markup*  
3432 *Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-  
3433 authn-context-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.

3434 [SAMLBind] S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*  
3435 *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-bindings-  
3436 2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.

3437 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion*  
3438 *Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID  
3439 sstc-saml-conformance-2.0-cd-03. [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3440 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3441 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*  
3442 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)  
3443 [open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3444 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language*  
3445 *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-glossary-  
3446 2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.

3447 [SAMLMeta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language*  
3448 *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-metadata-  
3449 2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.

3450 [SAMLXSD] S. Cantor et al., *SAML protocols schema*. OASIS SSTC, December 2004.  
3451 Document ID sstc-saml-schema-protocol-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3452 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3453 [SAMLProf] S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*  
3454 *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-profiles-  
3455 2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.

3456 [SAMLSecure] F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*  
3457 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004.  
3458 Document ID sstc-saml-sec-consider-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3459 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3460 [SAMLXSD] S. Cantor et al., *SAML assertions schema*. OASIS SSTC, December 2004.  
3461 Document ID sstc-saml-schema-assertion-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3462 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

- 3463 **[SAML-TechOvw]** J. Hughes et al. SAML Technical Overview. OASIS, July 2004. Document ID  
3464 oasis-sstc-saml-tech-overview-2.0. [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>3465 open.org/committees/security/).
- 3466 **[UNICODE-C]** M. Davis, M. J. Dürst. *Unicode Normalization Forms*. UNICODE Consortium,  
3467 March 2001. <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>.
- 3468 **[W3C-CHAR]** M. J. Dürst. *Requirements for String Identity Matching and String Indexing*. World  
3469 Wide Web Consortium, July 1998. <http://www.w3.org/TR/WD-charreq>.
- 3470 **[W3C-CharMod]** M. J. Dürst. *Character Model for the World Wide Web 1.0: Normalization*. World  
3471 Wide Web Consortium, February 2004. <http://www.w3.org/TR/charmod-norm/>.
- 3472 **[X.500]** ITU-T Recommendation X.501: Information Technology - Open Systems  
3473 Interconnection - The Directory: Models. 1993.
- 3474 **[XACML]** eXtensible Access Control Markup Language (XACML), product of the OASIS  
3475 XACML TC. See <http://www.oasis-open.org/committees/xacml>.
- 3476 **[XML-ID]** J. Marsh et al., *xml:id Version 1.0*, W3C, April 2004. [http://www.w3.org/TR/xml-  
id/](http://www.w3.org/TR/xml-<br/>3477 id/).

3478

---

## Appendix A. Acknowledgments

3479

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

3480

3481

- Conor Cahill, AOL

3482

- John Hughes, Atos Origin

3483

- Hal Lockhart, BEA Systems

3484

- Mike Beach, Boeing

3485

- Rebekah Metz, Booz Allen Hamilton

3486

- Rick Randall, Booz Allen Hamilton

3487

- Ronald Jacobson, Computer Associates

3488

- Paul Madsen, Entrust

3489

- Dana Kaufman, Forum Systems

3490

- Paula Austel, IBM

3491

- Michael McIntosh, IBM

3492

- Anthony Nadalin, IBM

3493

- Nick Ragouzis, Individual

3494

- Scott Cantor, Internet2

3495

- Bob Morgan, Internet2

3496

- Peter Davis, Neustar

3497

- Jeff Hodges, Neustar

3498

- Frederick Hirsch, Nokia

3499

- John Kemp, Nokia

3500

- Abbie Barbir, Nortel Networks

3501

- Scott Kiester, Novell

3502

- Cameron Morris, Novell

3503

- Charles Knouse, Oblix

3504

- Steve Anderson, OpenNetwork

3505

- Ari Kermaier, Oracle

3506

- Vamsi Motukuru, Oracle

3507

- Darren Platt, Ping Identity

3508

- Prateek Mishra, Principal Identity

3509

- Jim Lien, RSA Security

3510

- Rob Philpott, RSA Security

3511

- Dipak Chopra, SAP

3512

- Jahan Moreh, Sigaba

3513

- Bhavna Bhatnagar, Sun Microsystems

3514

- Eve Maler, Sun Microsystems

3515

- Ronald Monzillo, Sun Microsystems

3516

- Emily Xu, Sun Microsystems

3517

- Greg Whitehead, Trustgenix

3518



3519

3520 The editors also would like to acknowledge the following people for their contributions to previous versions  
3521 of the OASIS Security Assertions Markup Language Standard:

- 3522 • Stephen Farrell, Baltimore Technologies
- 3523 • David Orchard, BEA Systems
- 3524 • Krishna Sankar, Cisco Systems
- 3525 • Zahid Ahmed, CommerceOne
- 3526 • Carlisle Adams, Entrust
- 3527 • Tim Moses, Entrust
- 3528 • Nigel Edwards, Hewlett-Packard
- 3529 • Joe Pato, Hewlett-Packard
- 3530 • Bob Blakley, IBM
- 3531 • Marlena Erdos, IBM
- 3532 • Marc Chanliau, Netegrity
- 3533 • Chris McLaren, Netegrity
- 3534 • Lynne Rosenthal, NIST
- 3535 • Mark Skall, NIST
- 3536 • Simon Godik, Overxeer
- 3537 • Charles Norwood, SAIC
- 3538 • Evan Prodromou, Securant
- 3539 • Robert Griffin, RSA Security (former editor)
- 3540 • Sai Allarvarpu, Sun Microsystems
- 3541 • Chris Ferris, Sun Microsystems
- 3542 • Emily Xu, Sun Microsystems
- 3543 • Mike Myers, Traceroute Security
- 3544 • Phillip Hallam-Baker, VeriSign (former editor)
- 3545 • James Vanderbeek, Vodafone
- 3546 • Mark O'Neill, Vordel
- 3547 • Tony Palmer, Vordel

3548

3549 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
3550 input to the OASIS Security Assertions Markup Language specifications:

- 3551 • Thomas Gross, IBM
- 3552 • Birgit Pfitzmann, IBM

3553

## Appendix B. Notices

3554 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
3555 might be claimed to pertain to the implementation or use of the technology described in this document or  
3556 the extent to which any license under such rights might or might not be available; neither does it represent  
3557 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
3558 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
3559 available for publication and any assurances of licenses to be made available, or the result of an attempt  
3560 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
3561 users of this specification, can be obtained from the OASIS Executive Director.

3562 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
3563 other proprietary rights which may cover technology that may be required to implement this specification.  
3564 Please address the information to the OASIS Executive Director.

3565 **Copyright © OASIS Open 2004. All Rights Reserved.**

3566 This document and translations of it may be copied and furnished to others, and derivative works that  
3567 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
3568 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
3569 this paragraph are included on all such copies and derivative works. However, this document itself may  
3570 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
3571 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
3572 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
3573 into languages other than English.

3574 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
3575 or assigns.

3576 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3577 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
3578 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
3579 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.