



Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)

Committee Specification 01, 31 May 2002

Document identifier:

cs-sstc-core-01 ([PDF](#), [Word](#))

Location:

<http://www.oasis-open.org/committees/security/docs/>

Editors:

Phillip Hallam-Baker, VeriSign (pbaker@verisign.com)
Eve Maler, Sun Microsystems (eve.maler@sun.com)

Contributors:

Stephen Farrell, Baltimore Technologies
Irving Reid, Baltimore Technologies
David Orchard, BEA Systems
Krishna Sankar, Cisco Systems
Simon Godik, Crosslogix
Hal Lockhart, Entegriety
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Prateek Mishra, Netegrity
Charles Knouse, Oblix
Scott Cantor, Ohio State University
Darren Platt, formerly with RSA Security
Jeff Hodges, Sun Microsystems
Bob Blakley Tivoli
Marlena Erdos, Tivoli
RL "Bob" Morgan, University of Washington

Abstract:

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

37 **Status:**

38 This is a stable Committee Specification that is undergoing a vote of the OASIS membership in
39 pursuit of OASIS Standard status.

40 If you are on the security-services@lists.oasis-open.org list for committee members, send
41 comments there. If you are not on that list, subscribe to the [security-services-
comment@lists.oasis-open.org](mailto:security-services-
42 comment@lists.oasis-open.org) list and send comments there. To subscribe, send an email
43 message to security-services-comment-request@lists.oasis-open.org with the word "subscribe"
44 as the body of the message.

45 The errata document for this specification is located at [http://www.oasis-
open.org/committees/security/docs/](http://www.oasis-
46 open.org/committees/security/docs/). Its document identifier is draft-sstc-cs-errata-*nn*.

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

51 Copyright © 2001, 2002 The Organization for the Advancement of Structured Information Standards
52 [OASIS]

53 Table of Contents

54	1	Introduction.....	6
55	1.1	Notation.....	6
56	1.2	Schema Organization and Namespaces.....	6
57	1.2.1	String and URI Values.....	7
58	1.2.2	Time Values.....	7
59	1.2.3	Comparing SAML Values.....	7
60	1.3	SAML Concepts (Non-Normative).....	7
61	1.3.1	Overview.....	7
62	1.3.2	SAML and URI-Based Identifiers.....	9
63	1.3.3	SAML and Extensibility.....	9
64	2	SAML Assertions.....	10
65	2.1	Schema Header and Namespace Declarations.....	10
66	2.2	Simple Types.....	10
67	2.2.1	Simple Types IDType and IDReferenceType.....	10
68	2.2.2	Simple Type DecisionType.....	11
69	2.3	Assertions.....	11
70	2.3.1	Element <AssertionIDReference>.....	11
71	2.3.2	Element <Assertion>.....	11
72	2.3.2.1	Element <Conditions>.....	13
73	2.3.2.1.1	Attributes NotBefore and NotOnOrAfter.....	14
74	2.3.2.1.2	Element <Condition>.....	14
75	2.3.2.1.3	Elements <AudienceRestrictionCondition> and <Audience>.....	14
76	2.3.2.2	Element <Advice>.....	15
77	2.4	Statements.....	15
78	2.4.1	Element <Statement>.....	15
79	2.4.2	Element <SubjectStatement>.....	15
80	2.4.2.1	Element <Subject>.....	16
81	2.4.2.2	Element <NameIdentifier>.....	16
82	2.4.2.3	Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>.....	17
83	2.4.3	Element <AuthenticationStatement>.....	18
84	2.4.3.1	Element <SubjectLocality>.....	18
85	2.4.3.2	Element <AuthorityBinding>.....	19
86	2.4.4	Element <AuthorizationDecisionStatement>.....	19
87	2.4.4.1	Element <Action>.....	21
88	2.4.4.2	Element <Evidence>.....	21
89	2.4.5	Element <AttributeStatement>.....	21
90	2.4.5.1	Elements <AttributeDesignator> and <Attribute>.....	22
91	2.4.5.1.1	Element <AttributeValue>.....	23
92	3	SAML Protocol.....	24
93	3.1	Schema Header and Namespace Declarations.....	24
94	3.2	Requests.....	24
95	3.2.1	Complex Type RequestAbstractType.....	24

96	3.2.1.1 Element <RespondWith>	25
97	3.2.2 Element <Request>	25
98	3.2.3 Element <AssertionArtifact>	26
99	3.3 Queries	26
100	3.3.1 Element <Query>	26
101	3.3.2 Element <SubjectQuery>	27
102	3.3.3 Element <AuthenticationQuery>	27
103	3.3.4 Element <AttributeQuery>	28
104	3.3.5 Element <AuthorizationDecisionQuery>	28
105	3.4 Responses	29
106	3.4.1 Complex Type ResponseAbstractType	29
107	3.4.2 Element <Response>	30
108	3.4.3 Element <Status>	31
109	3.4.3.1 Element <StatusCode>	31
110	3.4.3.2 Element <StatusMessage>	32
111	3.4.3.3 Element <StatusDetail>	33
112	3.4.4 Responses to <AuthenticationQuery> and <AttributeQuery>	33
113	4 SAML Versioning	34
114	4.1 Assertion Version	34
115	4.2 Request Version	34
116	4.3 Response Version	34
117	5 SAML and XML Signature Syntax and Processing	36
118	5.1 Signing Assertions	36
119	5.2 Request/Response Signing	36
120	5.3 Signature Inheritance	37
121	5.4 XML Signature Profile	37
122	5.4.1 Signing Formats	37
123	5.4.2 Canonicalization Method	37
124	5.4.3 Transforms	37
125	5.4.4 KeyInfo	37
126	5.4.5 Binding Between Statements in a Multi-Statement Assertion	37
127	6 SAML Extensions	38
128	6.1 Assertion Schema Extension	38
129	6.2 Protocol Schema Extension	38
130	6.3 Use of Type Derivation and Substitution Groups	39
131	7 SAML-Defined Identifiers	40
132	7.1 Authentication Method Identifiers	40
133	7.1.1 Password	40
134	7.1.2 Kerberos	40
135	7.1.3 Secure Remote Password (SRP)	40
136	7.1.4 Hardware Token	40
137	7.1.5 SSL/TLS Certificate Based Client Authentication:	41
138	7.1.6 X.509 Public Key	41
139	7.1.7 PGP Public Key	41
140	7.1.8 SPKI Public Key	41

141	7.1.9 XKMS Public Key	41
142	7.1.10 XML Digital Signature.....	41
143	7.1.11 Unspecified.....	41
144	7.2 Action Namespace Identifiers.....	41
145	7.2.1 Read/Write/Execute/Delete/Control	41
146	7.2.2 Read/Write/Execute/Delete/Control with Negation	42
147	7.2.3 Get/Head/Put/Post	42
148	7.2.4 UNIX File Permissions	42
149	8 References	44
150	Appendix A. Acknowledgments	46
151	Appendix B. Notices.....	47
152		

153 1 Introduction

154 This specification defines the syntax and semantics for XML-encoded SAML assertions, protocol
155 requests, and protocol responses. These constructs are typically embedded in other structures for
156 transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML specification for
157 bindings and profiles **[SAMLBind]** provides frameworks for this embedding and transport. Files
158 containing just the SAML assertion schema **[SAML-XSD]** and protocol schema **[SAML-XP]** are
159 available.

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

161 1.1 Notation

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

164 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
165 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
166 described in IETF RFC 2119 **[RFC 2119]**:

167 *"they MUST only be used where it is actually required for interoperation or to limit*
168 *behavior which has potential for causing harm (e.g., limiting retransmissions)"*

169 These keywords are thus capitalized when used to unambiguously specify requirements over protocol
170 and application features and behavior that affect the interoperability and security of implementations.
171 When these words are not capitalized, they are meant in their natural-language sense.

172 `Listings of SAML schemas appear like this.`

173 `Example code listings appear like this.`

175 In cases of disagreement between the SAML schema files **[SAML-XSD]** **[SAML-XP]** and this
176 specification, the schema files take precedence.

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

- 180 • The prefix `saml`: stands for the SAML assertion namespace.
- 181 • The prefix `samlp`: stands for the SAML request-response protocol namespace.
- 182 • The prefix `ds`: stands for the W3C XML Signature namespace.
- 183 • The prefix `xsd`: stands for the W3C XML Schema namespace in example listings. In schema
184 listings, this is the default namespace and no prefix is shown.

185 This specification uses the following typographical conventions in text: `<SAMLElement>`,
186 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

187 1.2 Schema Organization and Namespaces

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

190 `urn:oasis:names:tc:SAML:1.0:assertion`

191 The SAML request-response protocol structures are defined in a schema **[SAML-XP]** associated with
192 the following XML namespace:

193 `urn:oasis:names:tc:SAML:1.0:protocol`

194 **Note:** The SAML namespace names may change when SAML 1.0 becomes an OASIS
195 Standard.

196 The assertion schema is imported into the protocol schema. Also imported into both schemas is the
197 schema for XML Signature [**XMLSig-XSD**], which is associated with the following XML namespace:

198 `http://www.w3.org/2000/09/xmlsig#`

199 **1.2.1 String and URI Values**

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

205 **1.2.2 Time Values**

206 All SAML time values have the type **dateTime**, which is built in to the W3C XML Schema Datatypes
207 specification [**Schema2**] and **MUST** be expressed in UTC form.

208 SAML Requestors and Responders **SHOULD NOT** rely on other applications supporting time resolution
209 finer than milliseconds. Implementations **MUST NOT** generate time instants that specify leap seconds.

210 **1.2.3 Comparing SAML Values**

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

217 If an implementation is comparing values that are represented using different character encodings, the
218 implementation **MUST** use a comparison method that returns the same result as converting both values
219 to the Unicode character encoding (<http://www.unicode.org>), Normalization Form C [**UNICODE-C**] and
220 then performing an exact binary comparison. This requirement is intended to conform to the W3C
221 Character Model for the World Wide Web [**W3C-CharMod**], and in particular the rules for Unicode-
222 normalized Text.

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

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

232 **1.3 SAML Concepts (Non-Normative)**

233 This section is informative only and is superseded by any contradicting information in the normative text
234 in Section 2 and following. A glossary of SAML terms and concepts [**SAMLGloss**] is available.

235 **1.3.1 Overview**

236 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security
237 information. This security information is expressed in the form of assertions about subjects, where a

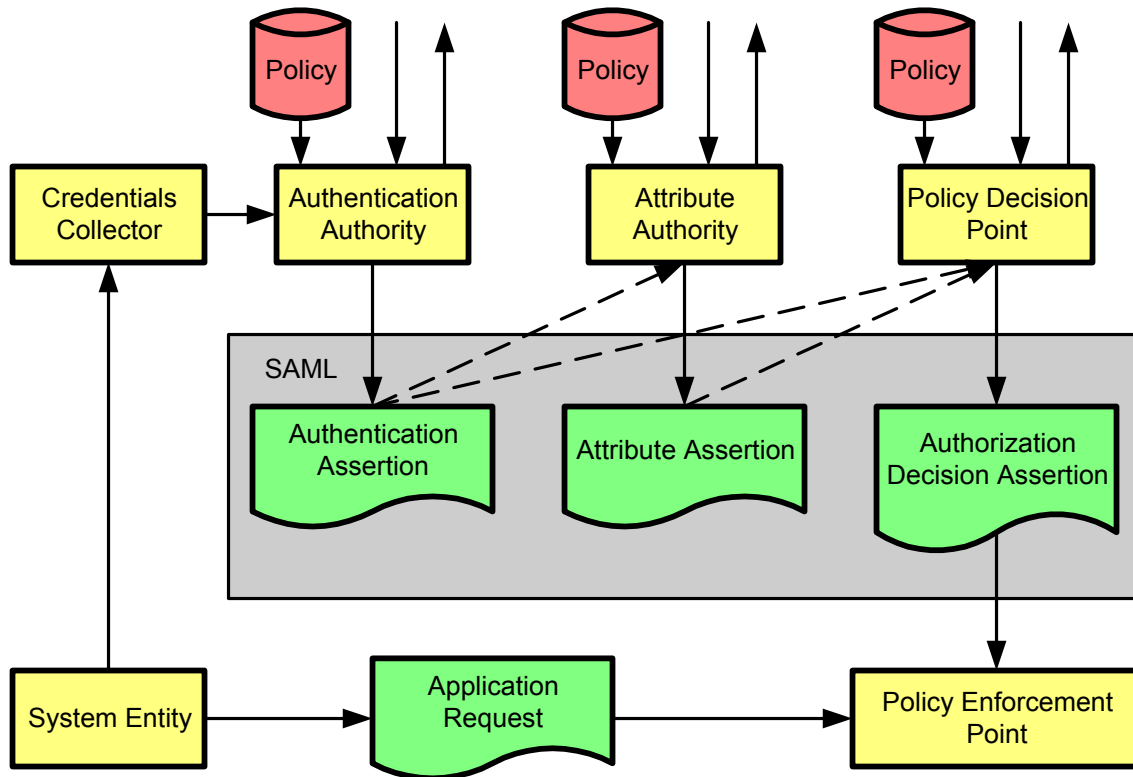
238 subject is an entity (either human or computer) that has an identity in some security domain. A typical
239 example of a subject is a person, identified by his or her email address in a particular Internet DNS
240 domain.

241 Assertions can convey information about authentication acts performed by subjects, attributes of subjects,
242 and authorization decisions about whether subjects are allowed to access certain resources. Assertions
243 are represented as XML constructs and have a nested structure, whereby a single assertion might
244 contain several different internal statements about authentication, authorization, and attributes. Note that
245 assertions containing authentication statements merely describe acts of authentication that happened
246 previously.

247 Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities, and
248 policy decision points. SAML defines a protocol by which clients can request assertions from SAML
249 authorities and get a response from them. This protocol, consisting of XML-based request and response
250 message formats, can be bound to many different underlying communications and transport protocols;
251 SAML currently defines one binding, to SOAP over HTTP.

252 SAML authorities can use various sources of information, such as external policy stores and assertions
253 that were received as input in requests, in creating their responses. Thus, while clients always consume
254 assertions, SAML authorities can be both producers and consumers of assertions.

255 The following model is conceptual only; for example, it does not account for real-world information flow or
256 the possibility of combining of authorities into a single system.



257
258

Figure 1 The SAML Domain Model

259 One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in one
260 domain and use resources in other domains without re-authenticating. However, SAML can be used in
261 various configurations to support additional scenarios as well. Several profiles of SAML are currently
262 being defined that support different styles of SSO and the securing of SOAP payloads.

263 The assertion and protocol data formats are defined in this specification. The bindings and profiles are
264 defined in a separate specification **[SAMLBind]**. A conformance program for SAML is defined in the

265 conformance specification [**SAMLConform**]. Security issues are discussed in a separate security and
266 privacy considerations specification [**SAMLSecure**].

267 **1.3.2 SAML and URI-Based Identifiers**

268 SAML defines some identifiers to manage references to well-known concepts and sets of values. For
269 example, the SAML-defined identifier for the password authentication method is as follows:

```
270 urn:oasis:names:tc:SAML:1.0:am:password
```

271 For another example, the SAML-defined identifier for the set of possible actions on a resource consisting
272 of Read/Write/Execute/Delete/Control is as follows:

```
273 urn:oasis:names:tc:SAML:1.0:action:rwedc
```

274 These identifiers are defined as Uniform Resource Identifiers (URIs), but they are not necessarily able to
275 be resolved to some Web resource. At times SAML authorities need to use identifier strings of their own
276 design, for example, for assertion IDs or additional kinds of authentication methods not covered by
277 SAML-defined identifiers. In these cases, using a URI form is not required; if it is used, it is not required to
278 be resolvable to some Web resource. However, using URIs – particularly URLs based on the `http:`
279 scheme – is likely to mitigate problems with clashing identifiers to some extent.

280 The Read/Write/Execute/Delete/Control identifier above is an example of a namespace (not in the sense
281 of an XML namespace). SAML uses this namespace mechanism to manage the universe of possible
282 types of actions and possible names of attributes.

283 See Section 7 for a list of SAML-defined identifiers.

284 **1.3.3 SAML and Extensibility**

285 The XML formats for SAML assertions and protocol messages have been designed to be extensible.

286 However, it is possible that the use of extensions will harm interoperability and therefore the use of
287 extensions SHOULD be carefully considered.

288

2 SAML Assertions

289 An assertion is a package of information that supplies one or more statements made by an issuer. SAML
290 allows issuers to make three different kinds of assertion statement:

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

296 Assertions have a nested structure. A series of inner elements representing authentication statements,
297 authorization decision statements, and attribute statements contain the specifics, while an outer generic
298 assertion element provides information that is common to all of the statements.

299 2.1 Schema Header and Namespace Declarations

300 The following schema fragment defines the XML namespaces and other header information for the
301 assertion schema:

```
302 <schema  
303   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
304   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
305   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
306   xmlns="http://www.w3.org/2001/XMLSchema"  
307   elementFormDefault="unqualified">  
308   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
309     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
310     schema.xsd"/>  
311   <annotation>  
312     <documentation>  
313       Document identifier: cs-sstc-schema-assertion-01  
314       Location: http://www.oasis-open.org/committees/security/docs/  
315     </documentation>  
316   </annotation>  
317   ...  
318 </schema>
```

319 2.2 Simple Types

320 The following sections define the SAML assertion-related simple types.

321 2.2.1 Simple Types IDType and IDReferenceType

322 The **IDType** simple type is used to declare identifiers to assertions, requests, and responses. The
323 **IDReferenceType** is used to reference identifiers of type **IDType**.

324 Values declared to be of type **IDType** MUST satisfy the following properties:

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

329 The mechanism by which the SAML Requestor or Responder ensures that the identifier is unique is left to
330 the implementation. In the case that a pseudorandom technique is employed, the probability of two
331 randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less than 2^{-160} . This

332 requirement MAY be met by applying Base64 encoding to a randomly chosen value 128 or 160 bits in
333 length.

334 It is OPTIONAL for an identifier based on **IDType** to be resolvable in principle to some resource. In the
335 case that the identifier is resolvable in principle (for example, the identifier is in the form of a URI
336 reference), it is OPTIONAL for the identifier to be dereferenceable.

337 The following schema fragment defines the **IDType** and **IDReferenceType** simple types:

```
338 <simpleType name="IDType">  
339   <restriction base="string"/>  
340 </simpleType>  
341 <simpleType name="IDReferenceType">  
342   <restriction base="string"/>  
343 </simpleType>
```

344 2.2.2 Simple Type DecisionType

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

347 Permit

348 The specified action is permitted.

349 Deny

350 The specified action is denied.

351 Indeterminate

352 The issuer cannot determine whether the specified action is permitted or denied.

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

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

```
357 <simpleType name="DecisionType">  
358   <restriction base="string">  
359     <enumeration value="Permit"/>  
360     <enumeration value="Deny"/>  
361     <enumeration value="Indeterminate"/>  
362   </restriction>  
363 </simpleType>
```

364 2.3 Assertions

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

366 2.3.1 Element `<AssertionIDReference>`

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

368 The following schema fragment defines the `<AssertionIDReference>` element:

```
369 <element name="AssertionIDReference" type="saml:IDReferenceType"/>
```

370 2.3.2 Element `<Assertion>`

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

373 MajorVersion [Required]
 374 The major version of this assertion. The identifier for the version of SAML defined in this specification
 375 is 1. Processing of this attribute is specified in Section 3.4.4.

376 MinorVersion [Required]
 377 The minor version of this assertion. The identifier for the version of SAML defined in this specification
 378 is 0. Processing of this attribute is specified in Section 3.4.4.

379 AssertionID [Required]
 380 The identifier for this assertion. It is of type **IDType**, and MUST follow the requirements specified by
 381 that type for identifier uniqueness.

382 Issuer [Required]
 383 The issuer of the assertion. The name of the issuer is provided as a string. The issuer name
 384 SHOULD be unambiguous to the intended relying parties. SAML authorities may use an identifier
 385 such as a URI reference that is designed to be unambiguous regardless of context.

386 IssueInstant [Required]
 387 The time instant of issue in UTC as described in Section 1.2.2.

388 <Conditions> [Optional]
 389 Conditions that MUST be taken into account in assessing the validity of the assertion.

390 <Advice> [Optional]
 391 Additional information related to the assertion that assists processing in certain situations but which
 392 MAY be ignored by applications that do not support its use.

393 <ds:Signature> [Optional]
 394 An XML Signature that authenticates the assertion, see Section 5.

395 One or more of the following statement elements:

396 <Statement>
 397 A statement defined in an extension schema.

398 <SubjectStatement>
 399 A subject statement defined in an extension schema.

400 <AuthenticationStatement>
 401 An authentication statement.

402 <AuthorizationDecisionStatement>
 403 An authorization decision statement.

404 <AttributeStatement>
 405 An attribute statement.

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

```

407 <element name="Assertion" type="saml:AssertionType"/>
408 <complexType name="AssertionType">
409   <sequence>
410     <element ref="saml:Conditions" minOccurs="0"/>
411     <element ref="saml:Advice" minOccurs="0"/>
412     <choice maxOccurs="unbounded">
413       <element ref="saml:Statement"/>
414       <element ref="saml:SubjectStatement"/>
415       <element ref="saml:AuthenticationStatement"/>
416       <element ref="saml:AuthorizationDecisionStatement"/>
417       <element ref="saml:AttributeStatement"/>
418     </choice>
419     <element ref="ds:Signature" minOccurs="0"/>
  
```

```

420     </sequence>
421     <attribute name="MajorVersion" type="integer" use="required"/>
422     <attribute name="MinorVersion" type="integer" use="required"/>
423     <attribute name="AssertionID" type="saml:IDType" use="required"/>
424     <attribute name="Issuer" type="string" use="required"/>
425     <attribute name="IssueInstant" type="dateTime" use="required"/>
426 </complexType>

```

427 2.3.2.1 Element <Conditions>

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

429 NotBefore [Optional]

430 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC as
 431 described in Section 1.2.2.

432 NotOnOrAfter [Optional]

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

435 <Condition> [Any Number]

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

437 <AudienceRestrictionCondition> [Any Number]

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

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

```

441 <element name="Conditions" type="saml:ConditionsType"/>
442 <complexType name="ConditionsType">
443   <choice minOccurs="0" maxOccurs="unbounded">
444     <element ref="saml:AudienceRestrictionCondition"/>
445     <element ref="saml:Condition"/>
446   </choice>
447   <attribute name="NotBefore" type="dateTime" use="optional"/>
448   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
449 </complexType>

```

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

- 454 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion
 455 is considered to be **Valid**.
- 456 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then
 457 the assertion is **Invalid**.
- 458 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the
 459 validity of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 460 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**,
 461 then the assertion is considered to be **Valid**.

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

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

468 **2.3.2.1.1 Attributes `NotBefore` and `NotOnOrAfter`**

469 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion.

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

472 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
473 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`), the
474 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the
475 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`),
476 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither
477 attribute is specified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is
478 valid at any time.

479 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is built
480 in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are specified in
481 Universal Coordinated Time (UTC) as described in Section 1.2.2. Implementations **MUST NOT** generate
482 time instants that specify leap seconds.

483 **2.3.2.1.2 Element `<Condition>`**

484 The `<Condition>` element serves as an extension point for new conditions. Its **ConditionAbstractType**
485 complex type is abstract; extension elements **MUST** use the `xsi:type` attribute to indicate the derived
486 type.

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

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

491 **2.3.2.1.3 Elements `<AudienceRestrictionCondition>` and `<Audience>`**

492 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to one or
493 more specific audiences identified by `<Audience>` elements. Although a party that is outside the
494 audiences specified is capable of drawing conclusions from an assertion, the issuer explicitly makes no
495 representation as to accuracy or trustworthiness to such a party. It contains the following elements:

496 `<Audience>`

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

499 The `AudienceRestrictionCondition` evaluates to `Valid` if and only if the relying party is a
500 member of one or more of the audiences specified.

501 The issuer of an assertion cannot prevent a party to whom it is disclosed from making a decision on the
502 basis of the information provided. However, the `<AudienceRestrictionCondition>` element allows
503 the issuer to state explicitly that no warranty is provided to such a party in a machine- and human-
504 readable form. While there can be no guarantee that a court would uphold such a warranty exclusion in
505 every circumstance, the probability of upholding the warranty exclusion is considerably improved.

506 The following schema fragment defines the `<AudienceRestrictionCondition>` element and its
507 **AudienceRestrictionConditionType** complex type:

```
508 <element name="AudienceRestrictionCondition"  
509 type="saml:AudienceRestrictionConditionType"/>  
510 <complexType name="AudienceRestrictionConditionType">
```

```

511     <complexContent>
512         <extension base="saml:ConditionAbstractType">
513             <sequence>
514                 <element ref="saml:Audience" maxOccurs="unbounded"/>
515             </sequence>
516         </extension>
517     </complexContent>
518 </complexType>
519 <element name="Audience" type="anyURI"/>

```

520 2.3.2.2 Element <Advice>

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

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

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

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

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

```

533 <element name="Advice" type="saml:AdviceType"/>
534 <complexType name="AdviceType">
535     <choice minOccurs="0" maxOccurs="unbounded">
536         <element ref="saml:AssertionIDReference"/>
537         <element ref="saml:Assertion"/>
538         <any namespace="##other" processContents="lax"/>
539     </choice>
540 </complexType>

```

541 2.4 Statements

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

543 2.4.1 Element <Statement>

544 The <Statement> element is an extension point that allows other assertion-based applications to reuse
545 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract; extension
546 elements MUST use the `xsi:type` attribute to indicate the derived type.

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

```

549 <element name="Statement" type="saml:StatementAbstractType"/>
550 <complexType name="StatementAbstractType" abstract="true"/>

```

551 2.4.2 Element <SubjectStatement>

552 The <SubjectStatement> element is an extension point that allows other assertion-based applications
553 to reuse the SAML assertion framework. It contains a <Subject> element that allows an issuer to
554 describe a subject. Its **SubjectStatementAbstractType** complex type, which extends

555 **StatementAbstractType**, is abstract; extension elements MUST use the `xsi:type` attribute to indicate
556 the derived type.

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

```
559 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
560 <complexType name="SubjectStatementAbstractType" abstract="true">  
561   <complexContent>  
562     <extension base="saml:StatementAbstractType">  
563       <sequence>  
564         <element ref="saml:Subject"/>  
565       </sequence>  
566     </extension>  
567   </complexContent>  
568 </complexType>
```

569 2.4.2.1 Element `<Subject>`

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

572 `<NameIdentifier>`

573 An identification of a subject by its name and security domain.

574 `<SubjectConfirmation>`

575 Information that allows the subject to be authenticated.

576 If the `<Subject>` element contains both a `<NameIdentifier>` and a `<SubjectConfirmation>`, the
577 issuer is asserting that if the relying party performs the specified `<SubjectConfirmation>`, it can be
578 confident that the entity presenting the assertion to the relying party is the entity that the issuer associates
579 with the `<NameIdentifier>`. A `<Subject>` element SHOULD NOT identify more than one principal.

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

```
581 <element name="Subject" type="saml:SubjectType"/>  
582 <complexType name="SubjectType">  
583   <choice>  
584     <sequence>  
585       <element ref="saml:NameIdentifier"/>  
586       <element ref="saml:SubjectConfirmation" minOccurs="0"/>  
587     </sequence>  
588     <element ref="saml:SubjectConfirmation"/>  
589   </choice>  
590 </complexType>
```

591 2.4.2.2 Element `<NameIdentifier>`

592 The `<NameIdentifier>` element specifies a subject by a combination of a name qualifier, a name and a
593 format. It has the following attributes:

594 `NameQualifier` [Optional]

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

597 `Format` [Optional]

598 The syntax used to describe the name of the subject.

599 The format value MUST be a URI reference. The following URI references are defined by this
600 specification, where only the fragment identifier portion is shown, assuming a base URI of the SAML
601 assertion namespace name.

602 #emailAddress

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

607 #X509SubjectName

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

612 #WindowsDomainQualifiedName

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

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

```
618 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
619 <complexType name="NameIdentifierType">  
620   <simpleContent>  
621     <extension base="string">  
622       <attribute name="NameQualifier" type="string" use="optional"/>  
623       <attribute name="Format" type="anyURI" use="optional"/>  
624     </extension>  
625   </simpleContent>  
626 </complexType>
```

627 The interpretation of the NameQualifier, and NameIdentifier's content in the case of a Format not
628 specified in this document, are left to individual implementations.

629 Regardless of format, issues of anonymity, pseudonymity, and the persistence of the identifier with
630 respect to the asserting and relying parties, are also implementation-specific.

631 **2.4.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 632 **<SubjectConfirmationData>**

633 The <SubjectConfirmation> element specifies a subject by supplying data that allows the subject to
634 be authenticated. It contains the following elements in order:

635 <ConfirmationMethod> [One or more]

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

640 <SubjectConfirmationData> [Optional]

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

642 <ds:KeyInfo> [Optional]

643 An XML Signature [XMLSig] element that specifies a cryptographic key held by the subject.

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

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

```

649     <sequence>
650         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
651         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
652         <element ref="ds:KeyInfo" minOccurs="0"/>
653     </sequence>
654 </complexType>
655 <element name="SubjectConfirmationData" type="anyType"/>
656 <element name="ConfirmationMethod" type="anyURI"/>

```

657 2.4.3 Element <AuthenticationStatement>

658 The <AuthenticationStatement> element supplies a statement by the issuer that its subject was
659 authenticated by a particular means at a particular time. It is of type **AuthenticationStatementType**,
660 which extends **SubjectStatementAbstractType** with the addition of the following element and attributes:

661 AuthenticationMethod [Required]

662 A URI reference that specifies the type of authentication that took place. URI references identifying
663 common authentication protocols are listed in Section 7.

664 AuthenticationInstant [Required]

665 Specifies the time at which the authentication took place. The time value is encoded in UTC as
666 described in Section 1.2.2.

667 <SubjectLocality> [Optional]

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

670 <AuthorityBinding> [Any Number]

671 Indicates that additional information about the subject of the statement may be available.

672 The following schema fragment defines the <AuthenticationStatement> element and its
673 **AuthenticationStatementType** complex type:

```

674 <element name="AuthenticationStatement"
675         type="saml:AuthenticationStatementType"/>
676 <complexType name="AuthenticationStatementType">
677     <complexContent>
678         <extension base="saml:SubjectStatementAbstractType">
679             <sequence>
680                 <element ref="saml:SubjectLocality" minOccurs="0"/>
681                 <element ref="saml:AuthorityBinding"
682                     minOccurs="0" maxOccurs="unbounded"/>
683             </sequence>
684             <attribute name="AuthenticationMethod" type="anyURI"
685 use="required"/>
686             <attribute name="AuthenticationInstant" type="dateTime"
687 use="required"/>
688         </extension>
689     </complexContent>
690 </complexType>

```

691 2.4.3.1 Element <SubjectLocality>

692 The <SubjectLocality> element specifies the DNS domain name and IP address for the system
693 entity that was authenticated. It has the following attributes:

694 IPAddress [Optional]

695 The IP address of the system entity that was authenticated.

696 DNSAddress [Optional]

697 The DNS address of the system entity that was authenticated.

698 This element is entirely advisory, since both these fields are quite easily “spoofed,” but current practice
699 appears to require its inclusion.

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

```
702 <element name="SubjectLocality"  
703         type="saml: SubjectLocalityType"/>  
704 <complexType name="SubjectLocalityType">  
705   <attribute name="IPAddress" type="string" use="optional"/>  
706   <attribute name="DNSAddress" type="string" use="optional"/>  
707 </complexType>
```

708 2.4.3.2 Element <AuthorityBinding>

709 The <AuthorityBinding> element may be used to indicate to a relying party receiving an
710 AuthenticationStatement that a SAML authority may be available to provide additional information about
711 the subject of the statement. A single SAML authority may advertise its presence over multiple protocol
712 bindings, at multiple locations, and as more than one kind of authority by sending multiple elements as
713 needed.

714 AuthorityKind [Required]

715 The type of SAML Protocol queries to which the authority described by this element will respond. The
716 value is specified as an XML Schema QName. The acceptable values for AuthorityKind are the
717 namespace-qualified names of element types or elements derived from the SAML Protocol Query
718 element (see Section 3.3). For example, an attribute authority would be identified by
719 AuthorityKind="samlp:AttributeQuery". For extension schemas, where the actual type of
720 the <samlp:Query> would be identified by an xsi:type attribute, the value of AuthorityKind
721 MUST be the same as the value of the xsi:type attribute for the corresponding query.

722 Location [Required]

723 A URI reference describing how to locate and communicate with the authority, the exact syntax of
724 which depends on the protocol binding in use. For example, a binding based on HTTP will be a web
725 URL, while a binding based on SMTP might use the "mailto" scheme.

726 Binding [Required]

727 A URI reference identifying the SAML protocol binding to use in communicating with the authority. All
728 SAML protocol bindings will have an assigned URI reference.

729 The following schema fragment defines the <AuthorityBinding> element and its
730 **AuthorityBindingType** complex type:

```
731 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>  
732 <complexType name="AuthorityBindingType">  
733   <attribute name="AuthorityKind" type="QName" use="required"/>  
734   <attribute name="Location" type="anyURI" use="required"/>  
735   <attribute name="Binding" type="anyURI" use="required"/>  
736 </complexType>
```

737 2.4.4 Element <AuthorizationDecisionStatement>

738 The <AuthorizationDecisionStatement> element supplies a statement by the issuer that the
739 request for access by the specified subject to the specified resource has resulted in the specified decision
740 on the basis of some optionally specified evidence.

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

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

751 To avoid ambiguity resulting from variations in URI encoding SAML requestors and responders SHOULD
752 employ the URI normalized form wherever possible as follows:

- 753 • The assertion issuer SHOULD encode all resource URIs in normalized form.
- 754 • Relying parties SHOULD convert resource URIs to normalized form prior to processing.

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

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

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

768 Resource [Required]

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

772 Decision [Required]

773 The decision rendered by the issuer with respect to the specified resource. The value is of the
774 **DecisionType** simple type.

775 <Action> [One or more]

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

777 <Evidence> [Optional]

778 A set of assertions that the issuer relied on in making the decision.

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

```
781 <element name="AuthorizationDecisionStatement"  
782 type="saml:AuthorizationDecisionStatementType"/>  
783 <complexType name="AuthorizationDecisionStatementType">  
784 <complexContent>  
785 <extension base="saml:SubjectStatementAbstractType">  
786 <sequence>  
787 <element ref="saml:Action" maxOccurs="unbounded"/>  
788 <element ref="saml:Evidence" minOccurs="0"/>  
789 </sequence>  
790 <attribute name="Resource" type="anyURI" use="required"/>
```

791
792
793
794
795

```
                <attribute name="Decision" type="saml:DecisionType"
use="required"/>
            </extension>
        </complexContent>
    </complexType>
```

796 2.4.4.1 Element <Action>

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

799 Namespace [Optional]

800 A URI reference representing the namespace in which the name of the specified action is to be
801 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwdc-
802 negation specified in Section 7.2.2 is in effect.

803 *string data* [Required]

804 An action sought to be performed on the specified resource.

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

806
807
808
809
810
811
812
813

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

814 2.4.4.2 Element <Evidence>

815 The <Evidence> element contains an assertion that the issuer relied on in issuing the authorization
816 decision. It has the **EvidenceType** complex type. It contains one of the following elements:

817 <AssertionIDReference>

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

819 <Assertion>

820 Specifies an assertion by value.

821 The provision of an assertion as evidence MAY affect the reliance agreement between the requestor and
822 the Authorization Authority. For example, in the case that the requestor presented an assertion to the
823 Authorization Authority in a request, the Authorization Authority MAY use that assertion as evidence in
824 making its response without endorsing the assertion as valid either to the requestor or any third party.

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

826
827
828
829
830
831
832

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

833 2.4.5 Element <AttributeStatement>

834 The <AttributeStatement> element supplies a statement by the issuer that the specified subject is
835 associated with the specified attributes. It is of type **AttributeStatementType**, which extends
836 **SubjectStatementAbstractType** with the addition of the following element:

837 <Attribute> [One or More]

838 The <Attribute> element specifies an attribute of the subject.

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

```
841 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
842 <complexType name="AttributeStatementType">
843   <complexContent>
844     <extension base="saml:SubjectStatementAbstractType">
845       <sequence>
846         <element ref="saml:Attribute" maxOccurs="unbounded"/>
847       </sequence>
848     </extension>
849   </complexContent>
850 </complexType>
```

851 2.4.5.1 Elements <AttributeDesignator> and <Attribute>

852 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It
853 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute
854 values within a specific namespace be returned (see Section 3.3.4 for more information). The
855 <AttributeDesignator> element contains the following XML attributes:

856 AttributeNamespace [Required]

857 The namespace in which the AttributeName elements are interpreted.

858 AttributeName [Required]

859 The name of the attribute.

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

```
862 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
863 <complexType name="AttributeDesignatorType">
864   <attribute name="AttributeName" type="string" use="required"/>
865   <attribute name="AttributeNamespace" type="anyURI" use="required"/>
866 </complexType>
```

867 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the
868 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following
869 element:

870 <AttributeValue> [Any Number]

871 The value of the attribute.

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

```
873 <element name="Attribute" type="saml:AttributeType"/>
874 <complexType name="AttributeType">
875   <complexContent>
876     <extension base="saml:AttributeDesignatorType">
877       <sequence>
878         <element ref="saml:AttributeValue"
879 maxOccurs="unbounded"/>
880       </sequence>
881     </extension>
882   </complexContent>
883 </complexType>
```

884 **2.4.5.1.1 Element <AttributeValue>**

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

887 If the data content of an AttributeValue element is of a XML Schema simple type (e.g. interger, string, etc)
888 the data type *MAY* be declared explicitly by means of an `xsi:type` declaration in the
889 <AttributeValue> element. If the attribute value contains structured data the necessary data elements
890 may be defined in an extension schema introduced by means of the `xmlns=` mechanism.

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

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

3 SAML Protocol

893

894 SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings and
895 profiles specification for SAML [**SAMLBind**] describes specific means of transporting assertions using
896 existing widely deployed protocols.

897 SAML-aware requestors MAY in addition use the SAML request-response protocol defined by the
898 <Request> and <Response> elements. The requestor sends a <Request> element to a SAML
899 authority, and the authority generates a <Response> element, as shown in Figure 2.



900

901

Figure 2: SAML Request-Response Protocol

3.1 Schema Header and Namespace Declarations

902

903 The following schema fragment defines the XML namespaces and other header information for the
904 protocol schema:

```
905 <schema  
906   targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"  
907   xmlns="http://www.w3.org/2001/XMLSchema"  
908   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
909   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
910   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
911   elementFormDefault="unqualified">  
912   <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"  
913     schemaLocation="cs-sstc-schema-assertion-01.xsd"/>  
914   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
915     schemaLocation="xmldsig-core-schema.xsd"/>  
916   <annotation>  
917     <documentation>  
918       Document identifier: cs-sstc-schema-protocol-01  
919       Location: http://www.oasis-open.org/committees/security/docs/  
920     </documentation>  
921   </annotation>  
922   ...  
923 </schema>
```

3.2 Requests

924

925 The following sections define the SAML constructs that contain request information.

3.2.1 Complex Type RequestAbstractType

926

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

929 RequestID [Required]

930 An identifier for the request. It is of type **IDType**, and MUST follow the requirements specified by that
931 type for identifier uniqueness. The values of the RequestID attribute in a request and the
932 InResponseTo attribute in the corresponding response MUST match.

933 MajorVersion [Required]

934 The major version of this request. The identifier for the version of SAML defined in this specification is

935 1. Processing of this attribute is specified in Section 3.4.2.

936 `MinorVersion` [Required]

937 The minor version of this request. The identifier for the version of SAML defined in this specification is

938 0. Processing of this attribute is specified in Section 3.4.2.

939 `IssueInstant` [Required]

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

941 1.2.2.

942 `<RespondWith>` [Any Number]

943 Each `<RespondWith>` element specifies a type of response that is acceptable to the requester.

944 `<ds:Signature>` [Optional]

945 An XML Signature that authenticates the request, see Section 5.

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

```
947 <complexType name="RequestAbstractType" abstract="true">
948   <sequence>
949     <element ref="saml:RespondWith"
950       minOccurs="0" maxOccurs="unbounded"/>
951     <element ref="ds:Signature" minOccurs="0"/>
952   </sequence>
953   <attribute name="RequestID" type="saml:IDType" use="required"/>
954   <attribute name="MajorVersion" type="integer" use="required"/>
955   <attribute name="MinorVersion" type="integer" use="required"/>
956   <attribute name="IssueInstant" type="dateTime" use="required"/>
957 </complexType>
```

958 3.2.1.1 Element `<RespondWith>`

959 The `<RespondWith>` element specifies the type of Statement the requestor wants from the responder.

960 Multiple `<RespondWith>` elements MAY be included to indicate that the requestor will accept assertions

961 containing any of the specified types. If no `<RespondWith>` element is given, the responder may return

962 assertions containing statements of any type.

963 If the requestor sends one or more `<RespondWith>` elements, the responder MUST NOT respond with

964 assertions containing statements of any type not specified in one of the `<RespondWith>` elements.

965 NOTE: Inability to find assertions that meet `<RespondWith>` criteria should be treated identical to any

966 other query for which no assertions are available. In both cases a status of success would normally be

967 returned in the Response message, but no assertions to be found therein.

968 `<RespondWith>` element values are XML QNames. The XML namespace and name specifically refer to

969 the namespace and element name of the Statement element, exactly as for the `saml:AuthorityKind`

970 attribute; see Section 2.4.3.2. For example, a requestor that wishes to receive assertions containing only

971 attribute statements must specify `<RespondWith>saml:AttributeStatement</RespondWith>`. To

972 specify extension types, the `<RespondWith>` element MUST contain exactly the extension element type

973 as specified in the `xsi:type` attribute on the corresponding element.

974 The following schema fragment defines the `<RespondWith>` element:

```
975 <element name="RespondWith" type="QName"/>
```

976 3.2.2 Element `<Request>`

977 The `<Request>` element specifies a SAML request. It provides either a query or a request for a specific

978 assertion identified by `<AssertionIDReference>` or `<AssertionArtifact>`. It has the complex

979 type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following

980 elements:

981 <Query>
 982 An extension point that allows extension schemas to define new types of query.
 983 <SubjectQuery>
 984 An extension point that allows extension schemas to define new types of query that specify a single
 985 SAML subject.
 986 <AuthenticationQuery>
 987 Makes a query for authentication information.
 988 <AttributeQuery>
 989 Makes a query for attribute information.
 990 <AuthorizationDecisionQuery>
 991 Makes a query for an authorization decision.
 992 <AssertionIDReference> [One or more]
 993 Requests assertions by reference to its assertion identifier.
 994 <AssertionArtifact> [One or more]
 995 Requests assertions by supplying an assertion artifact that represents it.
 996 The following schema fragment defines the <Request> element and its **RequestType** complex type:

```

997 <element name="Request" type="samlp:RequestType"/>
998 <complexType name="RequestType">
999   <complexContent>
1000     <extension base="samlp:RequestAbstractType">
1001       <choice>
1002         <element ref="samlp:Query"/>
1003         <element ref="samlp:SubjectQuery"/>
1004         <element ref="samlp:AuthenticationQuery"/>
1005         <element ref="samlp:AttributeQuery"/>
1006         <element ref="samlp:AuthorizationDecisionQuery"/>
1007         <element ref="saml:AssertionIDReference"
1008         maxOccurs="unbounded"/>
1009         <element ref="samlp:AssertionArtifact"
1010         maxOccurs="unbounded"/>
1011       </choice>
1012     </extension>
1013   </complexContent>
1014 </complexType>

```

1015 3.2.3 Element <AssertionArtifact>

1016 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
 1017 assertion.

1018 The following schema fragment defines the <AssertionArtifact> element:

```

1019 <element name="AssertionArtifact" type="string"/>

```

1020 3.3 Queries

1021 The following sections define the SAML constructs that contain query information.

1022 3.3.1 Element <Query>

1023 The <Query> element is an extension point that allows new SAML queries to be defined. Its
 1024 **QueryAbstractType** is abstract; extension elements MUST use the `xsi:type` attribute to indicate the
 1025 derived type. **QueryAbstractType** is the base type from which all SAML query elements are derived.

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

```
1027 <element name="Query" type="samlp:QueryAbstractType"/>
1028 <complexType name="QueryAbstractType" abstract="true"/>
```

1029 3.3.2 Element <SubjectQuery>

1030 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a single
1031 SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements MUST use
1032 the xsi:type attribute to indicate the derived type. **SubjectQueryAbstractType** adds the <Subject>
1033 element.

1034 The following schema fragment defines the <SubjectQuery> element and its
1035 **SubjectQueryAbstractType** complex type:

```
1036 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1037 <complexType name="SubjectQueryAbstractType" abstract="true">
1038   <complexContent>
1039     <extension base="samlp:QueryAbstractType">
1040       <sequence>
1041         <element ref="saml:Subject"/>
1042       </sequence>
1043     </extension>
1044   </complexContent>
1045 </complexType>
```

1046 3.3.3 Element <AuthenticationQuery>

1047 The <AuthenticationQuery> element is used to make the query “What assertions containing
1048 authentication statements are available for this subject?” A successful response will be in the form of
1049 assertions containing authentication statements.

1050 **Note:** The <AuthenticationQuery> MAY NOT be used as a request for a new authentication using
1051 credentials provided in the request. The <AuthenticationQuery> is a request for statements about
1052 authentication acts which have occurred in a previous interaction between the indicated principal and the
1053 Authentication Authority.

1054 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the
1055 addition of the following element:

1056 <AuthenticationMethod> [Optional]

1057 A filter for possible responses. If it is present, the query made is “What assertions containing
1058 authentication statements do you have for this subject with the supplied authentication method?”

1059 In response to an authentication query, a responder returns assertions with authentication statements as
1060 follows:

- 1061 • First, rules given in Section 3.4.4 for matching against the <Subject> element of the query identify
1062 the assertions that may be returned.
- 1063 • Further, if the <AuthenticationMethod> element is present in the query, at least one
1064 <AuthenticationMethod> element in the set of returned assertions MUST match. It is OPTIONAL
1065 for the complete set of all such matching assertions to be returned in the response.

1066 The following schema fragment defines the <AuthenticationQuery> type and its
1067 **AuthenticationQueryType** complex type:

```
1068 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1069 <complexType name="AuthenticationQueryType">
1070   <complexContent>
1071     <extension base="samlp:SubjectQueryAbstractType">
1072       <attribute name="AuthenticationMethod" type="anyURI"/>
1073     </extension>
```

```
1074     </complexContent>
1075 </complexType>
```

1076 3.3.4 Element <AttributeQuery>

1077 The <AttributeQuery> element is used to make the query "Return the requested attributes for this
1078 subject." A successful response will be in the form of assertions containing attribute statements. This
1079 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1080 the following element and attribute:

1081 Resource [Optional]

1082 The Resource attribute if present specifies that the attribute query is made in response to a specific
1083 authorization decision relating to the resource. The responder MAY use the resource attribute to
1084 establish the scope of the request. It is permitted for this attribute to have the value of the empty URI
1085 reference (""), and the meaning is defined to be "the start of the current document", as specified by
1086 **[RFC 2396] §4.2**.

1087 If the resource attribute is specified and the responder does not wish to support resource-specific
1088 attribute queries, or if the resource value provided is invalid or unrecognized, then it SHOULD
1089 respond with a top-level StatusCode value of Responder and a second-level code value of
1090 ResourceNotRecognized

1091 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1092 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no
1093 attributes are specified, it indicates that all attributes allowed by policy are requested.

1094 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1095 complex type:

```
1096 <element name="AttributeQuery" type="saml:AttributeQueryType"/>
1097 <complexType name="AttributeQueryType">
1098   <complexContent>
1099     <extension base="saml:SubjectQueryAbstractType">
1100       <sequence>
1101         <element ref="saml:AttributeDesignator"
1102           minOccurs="0" maxOccurs="unbounded"/>
1103       </sequence>
1104       <attribute name="Resource" type="anyURI reference"
1105         use="optional"/>
1106     </extension>
1107   </complexContent>
1108 </complexType>
```

1109 3.3.5 Element <AuthorizationDecisionQuery>

1110 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on
1111 this resource be allowed for this subject, given this evidence?" A successful response will be in the form
1112 of assertions containing authorization decision statements. This element is of type
1113 **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the
1114 following elements and attribute:

1115 Resource [Required]
1116 A URI reference indicating the resource for which authorization is requested.
1117 <Action> [One or More]
1118 The actions for which authorization is requested.
1119 <Evidence> [Optional]
1120 A set of assertions that the responder MAY rely on in making its response.
1121 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
1122 **AuthorizationDecisionQueryType** complex type:

```
1123 <element name="AuthorizationDecisionQuery"  
1124 type="samlp:AuthorizationDecisionQueryType"/>  
1125 <complexType name="AuthorizationDecisionQueryType">  
1126 <complexContent>  
1127 <extension base="samlp:SubjectQueryAbstractType">  
1128 <sequence>  
1129 <element ref="saml:Action" maxOccurs="unbounded"/>  
1130 <element ref="saml:Evidence" minOccurs="0"  
1131 maxOccurs="1"/>  
1132 </sequence>  
1133 <attribute name="Resource" type="anyURI" use="required"/>  
1134 </extension>  
1135 </complexContent>  
1136 </complexType>
```

1137 3.4 Responses

1138 The following sections define the SAML constructs that contain response information.

1139 3.4.1 Complex Type ResponseAbstractType

1140 All SAML responses are of types that are derived from the abstract **ResponseAbstractType** complex
1141 type. This type defines common attributes and elements that are associated with all SAML responses:

1142 **ResponseID** [Required]
 1143 An identifier for the response. It is of type **IDType**, and MUST follow the requirements specified by
 1144 that type for identifier uniqueness.

1145 **InResponseTo** [Optional]
 1146 A reference to the identifier of the request to which the response corresponds, if any. If the response
 1147 is not generated in response to a request, or if the RequestID of a request cannot be determined
 1148 (because the request is malformed), then this attribute MUST NOT be present. Otherwise, it MUST
 1149 be present and match the value of the corresponding RequestID attribute.

1150 **MajorVersion** [Required]
 1151 The major version of this response. The identifier for the version of SAML defined in this specification
 1152 is 1. Processing of this attribute is specified in Section 3.4.4.

1153 **MinorVersion** [Required]
 1154 The minor version of this response. The identifier for the version of SAML defined in this specification
 1155 is 0. Processing of this attribute is specified in Section 3.4.4.

1156 **IssueInstant** [Required]
 1157 The time instant of issue of the response. The time value is encoded in UTC as described in
 1158 Section 1.2.2.

1159 **Recipient** [Optional]
 1160 The intended recipient of this response. This is useful to prevent malicious forwarding of responses to
 1161 unintended recipients, a protection that is required by some use profiles. It is set by the generator of
 1162 the response to a URI reference that identifies the intended recipient. If present, the actual recipient
 1163 MUST check that the URI reference identifies the recipient or a resource managed by the recipient. If
 1164 it does not, the response MUST be discarded.

1165 **<ds:Signature>** [Optional]
 1166 An XML Signature that authenticates the response, see Section 5.

1167 The following schema fragment defines the **ResponseAbstractType** complex type:

```

1168 <complexType name="ResponseAbstractType" abstract="true">
1169   <sequence>
1170     <element ref = "ds:Signature" minOccurs="0"/>
1171   </sequence>
1172   <attribute name="ResponseID" type="saml:IDType" use="required"/>
1173   <attribute name="InResponseTo" type="saml:IDReferenceType" use="optional"/>
1174   <attribute name="MajorVersion" type="integer" use="required"/>
1175   <attribute name="MinorVersion" type="integer" use="required"/>
1176   <attribute name="IssueInstant" type="dateTime" use="required"/>
1177   <attribute name="Recipient" type="anyURI" use="optional"/>
1178 </complexType>
  
```

1179 **3.4.2 Element <Response>**

1180 The **<Response>** element specifies the status of the corresponding SAML request and a list of zero or
 1181 more assertions that answer the request. It has the complex type **ResponseType**, which extends
 1182 **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1183 **<Status>** [Required] (see Section 3.4.3)
 1184 A code representing the status of the corresponding request.

1185 **<Assertion>** [Any Number] (see Section 2.3.2)
 1186 Specifies an assertion by value.

1187 The following schema fragment defines the **<Response>** element and its **ResponseType** complex type:

```

1188 <element name="Response" type="samlp:ResponseType"/>
1189 <complexType name="ResponseType">
1190   <complexContent>
1191     <extension base="samlp:ResponseAbstractType">
1192       <sequence>
1193         <element ref="samlp:Status"/>
1194         <element ref="saml:Assertion" minOccurs="0"
1195 maxOccurs="unbounded"/>
1196       </sequence>
1197     </extension>
1198   </complexContent>
1199 </complexType>

```

3.4.3 Element <Status>

1200 The <Status> element:

1201 <StatusCode> [Required]

1202 A code representing the status of the corresponding request.

1203 <StatusMessage> [Optional]

1204 A message which MAY be returned to an operator.

1205 <StatusDetail> [Optional]

1206 Specifies additional information concerning an error condition.

1207 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1208 <element name="Status" type="samlp:StatusType"/>
1209 <complexType name="StatusType">
1210   <sequence>
1211     <element ref="samlp:StatusCode"/>
1212     <element ref="samlp:StatusMessage" minOccurs="0" maxOccurs="1"/>
1213     <element ref="samlp:StatusDetail" minOccurs="0"/>
1214   </sequence>
1215 </complexType>
1216

```

3.4.3.1 Element <StatusCode>

1217 The <StatusCode> element specifies one or more nested codes representing the status of the
1218 corresponding request. top-most code value MUST be one of the values defined below. Subsequent
1219 nested code values, if present, may provide more specific information concerning a particular error.

1220 Value [Required]

1221 The status code value as defined below.

1222 <StatusCode> [Optional]

1223 An optional subordinate status code value that provides more specific information on an error
1224 condition.

1225 The following top-level **StatusCode** Value QNames are defined. The responder MUST NOT include a
1226 code not listed below except by nesting it below one of the listed values.

1228 Success

1229 The request succeeded.

1230 VersionMismatch

1231 The receiver could not process the request because the version was incorrect.

1232 Requester

1233 The request could not be performed due to an error on the part of the requester.

1234 Responder

1235 The request could not be performed due to an error on the part of the responder.

1236 The following second-level status codes are referenced at various places in the specification. Additional

1237 subcodes MAY be defined in future versions of the SAML specification.

1238 RequestVersionTooHigh

1239 The protocol version specified in the request is a major upgrade from the highest protocol version

1240 supported by the responder.

1241 RequestVersionTooLow

1242 The responder cannot respond to the particular request using the SAML version specified in the

1243 request because it is too low.

1244 RequestVersionDeprecated

1245 The responder does not respond to any requests with the protocol version specified in the request.

1246 TooManyResponses

1247 The response would contain more elements than the responder will return.

1248 RequestDenied

1249 The responder is able to process the request but has chosen not to respond. MAY be used when the

1250 responder is concerned about the security context of the request or the sequence of requests

1251 received from a particular client.

1252 ResourceNotRecognized

1253 The responder does not wish to support resource-specific attribute queries, or the resource value

1254 provided is invalid or unrecognized.

1255 All status code values defined in this document are QNames associated with the SAML protocol

1256 namespace, urn:oasis:names:tc:SAML:1.0:protocol, and MUST be prefixed appropriately when they appear

1257 in SAML messages. SAML extensions and SAML Responders are free to define more specific status

1258 codes in other namespaces, but MAY NOT define additional codes in either the SAML assertion or

1259 protocol namespaces.

1260 The QNames defined as status codes SHOULD only be used in the StatusCode element's Value attribute

1261 and have the above semantics only in that context.

1262 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex

1263 type:

```

1264 <element name="StatusCode" type="samlp:StatusCodeType"/>
1265 <complexType name="StatusCodeType">
1266   <sequence>
1267     <element ref="samlp:StatusCode" minOccurs="0"/>
1268   </sequence>
1269   <attribute name="Value" type="QName" use="required"/>
1270 </complexType>

```

1271 **3.4.3.2 Element <StatusMessage>**

1272 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1273 The following schema fragment defines the <StatusMessage> element and its **StatusMessageType**
1274 complex type:

```
1275 <element name="StatusMessage" type="string"/>
```

1276 3.4.3.3 Element <StatusDetail>

1277 The <StatusDetail> element MAY be used to specify additional information concerning an error
1278 condition.

1279 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1280 complex type:

```
1281 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1282 <complexType name="StatusDetailType">  
1283   <sequence>  
1284     <any namespace="##any" processContents="lax" minOccurs="0"  
1285     maxOccurs="unbounded"/>  
1286   </sequence>  
1287 </complexType>
```

1288 3.4.4 Responses to <AuthenticationQuery> and <AttributeQuery>

1289 Responses to Authentication and Attribute queries are constructed by matching against the
1290 <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>
1291 elements. In response to these queries, every assertion returned by a SAML responder MUST contain at
1292 least one statement whose <saml:Subject> element **strongly matches** the <saml:Subject>
1293 element found in the query.

1294 A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1295 1 If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical
1296 <saml:NameIdentifier> element.
- 1297 2 If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an identical
1298 <saml:SubjectConfirmation> element.

1299 If the responder cannot provide an assertion with any statement(s) satisfying the constraints expressed
1300 by a query, the <saml:Response> element MUST NOT contain an <Assertion> element and MUST
1301 include a <saml:StatusCode> with value "Success". It MAY return a <saml:StatusMessage> with
1302 additional information.

1303 4 SAML Versioning

1304 SAML version information appears in the following elements:

- 1305 • <Assertion>
- 1306 • <Request>
- 1307 • <Response>

1308 The version numbering of the SAML assertion is independent of the version numbers of the SAML
1309 request-response protocol. The version information for each consists of a major version number and a
1310 minor version number, both of which are integers. In accordance with industry practice a version number
1311 SHOULD be presented to the user in the form *Major.Minor*. This document defines SAML Assertions 1.0
1312 and SAML Protocol 1.0.

1313 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

$$1314 \quad Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$$

1315 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that are
1316 the same as or higher than the corresponding version number in the SAML version that immediately
1317 preceded it. The specifications may be revised without a change to the SAML major or minor version
1318 number, as long as the specification changes raise no compatibility issues with SAML implementations.

1319 New versions of SAML SHALL assign new version numbers as follows:

- 1320 • **Minor upgrade:** $(Major_B = Major_A) \wedge (Minor_B > Minor_A)$
1321 If the major version number of versions *A* and *B* are the same and the minor version number of *B*
1322 is higher than that of *A*, the new SAML version MAY introduce changes to the SAML schema and
1323 semantics but any changes that are introduced in *B* SHALL be compatible with version *A*.
- 1324 • **Major upgrade:** $Major_B > Major_A$
1325 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY introduce
1326 changes to the SAML schema and semantics that are incompatible with *A*.

1327 4.1 Assertion Version

1328 A SAML authority MUST NOT issue any assertion whose version number is not supported.

1329 A SAML relying party MUST reject any assertion whose major version number is not supported.

1330 A SAML relying party MAY reject any assertion whose version number is higher than the highest
1331 supported version.

1332 4.2 Request Version

1333 A SAML authority SHOULD issue requests that specify the highest SAML version supported by both the
1334 sender and recipient.

1335 If the SAML authority does not know the capabilities of the recipient it should assume that it supports the
1336 highest SAML version supported by the sender.

1337 4.3 Response Version

1338 A SAML authority MUST NOT issue responses that specify a higher SAML version number than the
1339 corresponding request.

1340 A SAML authority MUST NOT issue a response that has a major version number that is lower than the
1341 major version number of the corresponding request except to report the error
1342 `RequestVersionTooHigh`.

- 1343 An error response resulting from incompatible protocol versions MUST result in reporting a top-level
1344 StatusCode value of VersionMismatch, and MAY result in reporting one of the following second-level
1345 values:
- 1346 RequestVersionTooHigh
- 1347 The protocol version specified in the request is a major upgrade from the highest protocol version
1348 supported by the responder.
- 1349 RequestVersionTooLow
- 1350 The responder cannot respond to the particular request using the SAML version specified in the
1351 request because it is too low.
- 1352 RequestVersionDeprecated
- 1353 The responder does not respond to any requests with the protocol version specified in the request.

5 SAML and XML Signature Syntax and Processing

1354

1355 SAML Assertions, Request and Response messages may be signed, with the following benefits:

- 1356
- 1357 • An Assertion signed by the asserting party (AP). This supports:
 - 1358 1. Message integrity.
 - 1359 2. Authentication of the asserting party to a relying party (RP).
 - 1360 3. If the signature is based on the asserting party's public-private key pair, then it also provides for non-repudiation of origin.
 - 1361 • A SAML request or a SAML response message signed by the message originator. This supports:
 - 1362 1. Message integrity.
 - 1363 2. Authentication of message origin to a destination.
 - 1364 3. If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.

1366 Note:

- 1367 • SAML documents may be the subject of signatures from different packaging contexts. **[XMLSig]**
1368 provides a framework for signing in XML and is the framework of choice. However, signing may
1369 also take place in the context of S/MIME or Java objects that contain SAML documents. One goal
1370 is to ensure compatibility with this type of "foreign" digital signing.
- 1371 • It is useful to characterize situations when a digital signature is NOT required in SAML.

1372 Assertions:

1373 The asserting party has provided the assertion to the relying party, authenticated by means other than
1374 digital signature and the channel is secure. In other words, the RP has obtained the assertion from the AP
1375 directly (no intermediaries) through a secure channel and the AP has authenticated to the RP.

1376 Request/Response messages:

1377 The originator has authenticated to the destination and the destination has obtained the assertion directly
1378 from the originator (no intermediaries) through secure channel(s).

1379 Many different techniques are available for "direct" authentication and secure channel between two
1380 parties. The list includes SSL, HMAC, password-based login etc. Also the security requirement depends
1381 on the communicating applications and the nature of the assertion transported.

1382 All other contexts require the use of digital signature for assertions and request and response messages.
1383 Specifically:

- 1384 1. An assertion obtained by a relying party from an entity other than the asserting party **MUST** be signed
1385 by the asserting party.
- 1386 2. A SAML message arriving at a destination from an entity other than the originating site **MUST** be
1387 signed by the origin site.

5.1 Signing Assertions

1389 All SAML assertions **MAY** be signed using the XML Signature. This is reflected in the assertion schema –
1390 Section 2.3.

5.2 Request/Response Signing

1392 All SAML requests and responses **MAY** be signed using the XML Signature. This is reflected in the
1393 schema – Sections 3.2 and 3.4.

1394 **5.3 Signature Inheritance**

1395 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
1396 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a
1397 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
1398 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,
1399 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
1400 interpretation should be equivalent to the case where the assertion itself was signed with the same key
1401 and signature options.

1402 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
1403 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may
1404 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
1405 information from the surrounding context, but no such inheritance should be inferred unless specifically
1406 identified by the profile.

1407 **5.4 XML Signature Profile**

1408 The XML Signature **[XMLSig]** specification calls out a general XML syntax for signing data with many
1409 flexibilities and choices. This section details the constraints on these facilities so that SAML processors
1410 do not have to deal with the full generality of XML Signature processing.

1411 **5.4.1 Signing Formats**

1412 XML Signature has three ways of representing signature in a document: enveloping, enveloped, and
1413 detached.

1414 SAML assertions and protocols **MUST** use the enveloped signatures for signing assertions and protocols.
1415 SAML processors should support use of RSA signing and verification for public key operations.

1416 **5.4.2 Canonicalization Method**

1417 XML Signature **REQUIRES** the Canonical XML (omits comments) ([http://www.w3.org/TR/2001/REC-xml-](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)
1418 [c14n-20010315](http://www.w3.org/TR/2001/REC-xml-c14n-20010315)). SAML implementations **SHOULD** use Canonical XML with no comments.

1419 **5.4.3 Transforms**

1420 **[XMLSig]** **REQUIRES** the enveloped signature transform [http://www.w3.org/2000/09/xmldsig#enveloped-](http://www.w3.org/2000/09/xmldsig#enveloped-signature)
1421 [signature](http://www.w3.org/2000/09/xmldsig#enveloped-signature).

1422 **5.4.4 KeyInfo**

1423 SAML does not restrict or impose any restrictions in this area. Therefore, following **[XMLSig]**,
1424 `<ds:KeyInfo>` may be absent.

1425 **5.4.5 Binding Between Statements in a Multi-Statement Assertion**

1426 Use of signing does not affect semantics of statements within assertions in any way, as stated in this
1427 document Sections 1 through 4.

1428 6 SAML Extensions

1429 The SAML schemas support extensibility. An example of an application that extends SAML assertions is
1430 the XTAML system for management of embedded trust roots [XTAML]. The following sections explain
1431 how to use the extensibility features in SAML to create extension schemas.

1432 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML elements
1433 MAY serve as the head element of a substitution group. Also, types are not defined as *final*, so that all
1434 SAML types MAY be extended and restricted. The following sections discuss only elements that have
1435 been specifically designed to support extensibility.

1436 6.1 Assertion Schema Extension

1437 The SAML assertion schema is designed to permit separate processing of the assertion package and the
1438 statements it contains, if the extension mechanism is used for either part.

1439 The following elements are intended specifically for use as extension points in an extension schema; their
1440 types are set to *abstract*, so that the use of an *xsi:type* attribute with these elements is REQUIRED:

- 1441 • <Assertion>
- 1442 • <Condition>
- 1443 • <Statement>
- 1444 • <SubjectStatement>

1445 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1446 • <AuthenticationStatement>
- 1447 • <AuthorizationDecisionStatement>
- 1448 • <AttributeStatement>
- 1449 • <AudienceRestrictionCondition>

1450 Finally, the following elements are defined to allow elements from arbitrary namespaces within them,
1451 which serves as a built-in extension point without requiring an extension schema:

- 1452 • <AttributeValue>
- 1453 • <Advice>

1454 6.2 Protocol Schema Extension

1455 The following elements are intended specifically for use as extension points in an extension schema; their
1456 types are set to *abstract*, so that the use of an *xsi:type* attribute with these elements is REQUIRED:

- 1457 • <Query>
- 1458 • <SubjectQuery>

1459 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1460 • <Request>
- 1461 • <AuthenticationQuery>
- 1462 • <AuthorizationDecisionQuery>
- 1463 • <AttributeQuery>
- 1464 • <Response>

1465 6.3 Use of Type Derivation and Substitution Groups

1466 W3C XML Schema [**Schema1**] provides two principal mechanisms for specifying an element of an
1467 extended type: type derivation and substitution groups.

1468 For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of the
1469 `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived
1470 from **StatementType**. The following example of a SAML assertion assumes that the extension schema
1471 (represented by the `new:` prefix) has defined this new type:

```
1472 <saml:Assertion ...>  
1473   <saml:Statement xsi:type="new:NewStatementType">  
1474     ...  
1475   </saml:Statement>  
1476 </saml:Assertion>
```

1477 Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a
1478 substitution group that has `<Statement>` as a head element. For the substituted element to be schema-
1479 valid, it needs to have a type that matches or is derived from the head element's type. The following is an
1480 example of an extension schema fragment that defines this new element:

```
1481 <xsd:element "NewStatement" type="new:NewStatementType"  
1482   substitutionGroup="saml:Statement"/>
```

1483 The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML
1484 `<Statement>` element can be used. The following is an example of a SAML assertion that uses the
1485 extension element:

```
1486 <saml:Assertion ...>  
1487   <new:NewStatement>  
1488     ...  
1489   </new:NewStatement>  
1490 </saml:Assertion>
```

1491 The choice of extension method has no effect on the semantics of the XML document but does have
1492 implications for interoperability.

1493 The advantages of type derivation are as follows:

- 1494 • A document can be more fully interpreted by a parser that does not have access to the extension
1495 schema because a “native” SAML element is available.
- 1496 • At the time of writing, some W3C XML Schema validators do not support substitution groups,
1497 whereas the `xsi:type` attribute is widely supported.

1498 The advantage of substitution groups is that a document can be explained without the need to explain the
1499 functioning of the `xsi:type` attribute.

1500 7 SAML-Defined Identifiers

1501 The following sections define URI-based identifiers for common authentication protocols and actions.

1502 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of
1503 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
1504 have the initial stem:

1505 `urn:oasis:names:tc:SAML:1.0:`

1506 7.1 Authentication Method Identifiers

1507 The <AuthenticationMethod> and <SubjectConfirmationMethod> elements perform different
1508 functions within the SAML architecture, although both can refer to the same underlying mechanisms.
1509 <AuthenticationMethod> is a part of an Authentication Statement, which describes an authentication
1510 act which occurred in the past. The <AuthenticationMethod> indicates how that authentication was
1511 done. Note that the authentication statement does not provide the means to perform that authentication,
1512 such as a password, key, or certificate.

1513 In contrast, <SubjectConfirmationMethod> is a part of the <SubjectConfirmation>, which is
1514 used to allow the Relying Party to confirm that the request or message came from the System Entity that
1515 corresponds to the Subject in the statement. The <SubjectConfirmationMethod> indicates the
1516 method that the Relying Party can use to do this in the future. This may or may not have any relationship
1517 to an authentication that was performed previously. Unlike the Authentication Method, the
1518 <SubjectConfirmationMethod> may be accompanied with some piece of information, such as a
1519 certificate or key, which will allow the Relying Party to perform the necessary check.

1520 Subject Confirmation Methods are defined in the SAML Profile or Profiles in which they are used
1521 [SAMLBind]. Additional methods may be added by defining new profiles or by private agreement.

1522 The following identifiers refer to SAML specified Authentication methods.

1523 7.1.1 Password

1524 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

1525 The authentication was performed by means of a password.

1526 7.1.2 Kerberos

1527 **URI:** urn:ietf:rfc:1510

1528 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the
1529 Needham-Schroeder symmetric key authentication mechanism [Needham78].

1530 7.1.3 Secure Remote Password (SRP)

1531 **URI:** urn:ietf:rfc:2945

1532 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC
1533 2945].

1534 7.1.4 Hardware Token

1535 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

1536 The authentication was performed by means of an unspecified hardware token.

1537 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

1538 **URI:** urn:ietf:rfc:2246

1539 The authentication was performed using either the SSL or TLS protocol with certificate based client
1540 authentication. TLS is described in **[RFC 2246]**.

1541 **7.1.6 X.509 Public Key**

1542 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

1543 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1544 an X.509 PKI **[X.500][PKIX]**. It may have been one of the mechanisms for which a more specific identifier
1545 has been defined below.

1546 **7.1.7 PGP Public Key**

1547 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

1548 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1549 a PGP web of trust **[PGP]**. It may have been one of the mechanisms for which a more specific identifier
1550 has been defined below.

1551 **7.1.8 SPKI Public Key**

1552 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

1553 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1554 a SPKI PKI **[SPKI]**. It may have been one of the mechanisms for which a more specific identifier has
1555 been defined below.

1556 **7.1.9 XKMS Public Key**

1557 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

1558 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1559 a XKMS trust service **[XKMS]**. It may have been one of the mechanisms for which a more specific
1560 identifier has been defined below.

1561 **7.1.10 XML Digital Signature**

1562 **URI:** urn:ietf:rfc:3075

1563 The authentication was performed by means of an XML digital signature
1564 **[RFC 3075]**.

1565 **7.1.11 Unspecified**

1566 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

1567 The authentication was performed by an unspecified means.

1568 **7.2 Action Namespace Identifiers**

1569 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see Section
1570 2.4.4.1) to refer to common sets of actions to perform on resources.

1571 **7.2.1 Read/Write/Execute/Delete/Control**

1572 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

1573 Defined actions:

1574 Read Write Execute Delete Control
1575 These actions are interpreted in the normal manner, i.e.:
1576 Read
1577 The subject may read the resource.
1578 Write
1579 The subject may modify the resource.
1580 Execute
1581 The subject may execute the resource.
1582 Delete
1583 The subject may delete the resource.
1584 Control
1585 The subject may specify the access control policy for the resource.

1586 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

1587 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

1588 Defined actions:

1589 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1590 The actions specified in Section 7.2.1 are interpreted in the same manner described there. Actions
1591 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
1592 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1593 affirmatively denied read permission.

1594 A SAML authority MUST NOT authorize both an action and its negated form.

1595 **7.2.3 Get/Head/Put/Post**

1596 **URI:** urn:oasis:names:tc:SAML:1.0: action:ghpp

1597 Defined actions:

1598 GET HEAD PUT POST

1599 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
1600 the GET action on a resource is authorized to retrieve it.

1601 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
1602 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
1603 operation may cause data to be modified and a POST operation may cause modification to a resource
1604 other than the one specified in the request. For this reason a separate Action URI reference specifier is
1605 provided.

1606 **7.2.4 UNIX File Permissions**

1607 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1608 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

1609 The action string is a four-digit numeric code:

1610 *extended user group world*

1611 Where the *extended* access permission has the value

1612 +2 if sgid is set

1613 +4 if suid is set

1614 The *user group* and *world* access permissions have the value

1615 +1 if execute permission is granted
1616 +2 if write permission is granted
1617 +4 if read permission is granted
1618 For example, 0754 denotes the UNIX file access permission: user read, write and execute, group read
1619 and execute and world read.

8 References

- 1620
- 1621 **[Kern-84]** B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March 1984)
1622 Prentice Hall Computer Books;
- 1623 **[Needham78]** R. Needham et al., *Using Encryption for Authentication in Large Networks of*
1624 *Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999, December
1625 1978.
- 1626 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann, *PGP Message Exchange Formats*,
1627 RFC 1991, August 1996.
- 1628 **[PKCS1]** B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also IETF
1629 RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 1630 **[PKCS7]** B. Kaliski., *PKCS #7: Cryptographic Message Syntax, Version 1.5*, RFC 2315,
1631 March 1998.
- 1632 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*
1633 *Certificate and CRL Profile*. RFC 2459, January 1999.
- 1634 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
1635 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 1636 **[RFC 2104]** H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,
1637 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 1638 **[RFC 2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1639 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 1640 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.
1641 <http://www.ietf.org/rfc/rfc2246.txt>
- 1642 **[RFC 2253]** M. Wahl et al., *Lightweight Directory Access Protocol (v3): UTF-8 String*
1643 *Representation of Distinguished Names*. December 1997.
1644 <http://www.ietf.org/rfc/rfc2253.txt>
- 1645 **[RFC 2396]** T. Berners-Lee et al., *Uniform Resource Identifiers (URI): Generic Syntax*
1646 <http://www.ietf.org/rfc/rfc2396.txt> IETF?
- 1647 **[RFC 2630]** R. Housley. *Cryptographic Message Syntax*. June 1999.
1648 <http://www.ietf.org/rfc/rfc2630.txt>
- 1649 **[RFC 2648]** R. Moats. *A URN Namespace for IETF Documents*. August 1999.
1650 <http://www.ietf.org/rfc/rfc2648.txt>
- 1651 **[RFC 2822]** P. Resnick. *Internet Message Format*. April 2001.
1652 <http://www.ietf.org/rfc/rfc2822.txt>
- 1653 **[RFC 2945]** T. Wu, *The SRP Authentication and Key Exchange System*, September 2000,
1654 <http://www.ietf.org/rfc/rfc2945.txt>
- 1655 **[RFC 3075]** D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. March
1656 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 1657 **[SAMLBind]** P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion Markup*
1658 *Language (SAML)*, <http://www.oasis-open.org/committees/security/>, OASIS, May
1659 2002.
- 1660 **[SAMLConform]** *Conformance Program Specification for the OASIS Security Assertion Markup*
1661 *Language (SAML)* <http://www.oasis-open.org/committees/security/>, OASIS, May
1662 2002.
- 1663 **[SAMLGloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language*
1664 *(SAML)*, <http://www.oasis-open.org/committees/security/>, OASIS, May 2002.
- 1665 **[SAMLXSD]** P. Hallam-Baker et al., *SAML protocol schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1666 [open.org/committees/security/](http://www.oasis-open.org/committees/security/), OASIS, May 2002.

1667	[SAMLSecure]	<i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML)</i> , http://www.oasis-open.org/committees/security/ , OASIS, May 2002.
1668		
1669		
1670	[SAML-XSD]	P. Hallam-Baker et al., <i>SAML assertion schema</i> , http://www.oasis-open.org/committees/security/ , OASIS, May 2002.
1671		
1672	[Schema1]	H. S. Thompson et al., <i>XML Schema Part 1: Structures</i> , http://www.w3.org/TR/xmlschema-1/ , World Wide Web Consortium Recommendation, May 2001.
1673		
1674		
1675	[Schema2]	P. V. Biron et al., <i>XML Schema Part 2: Datatypes</i> , http://www.w3.org/TR/xmlschema-2/ , World Wide Web Consortium Recommendation, May 2001.
1676		
1677		
1678	[SPKI]	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> , RFC 2693, September 1999.
1679		
1680	[UNICODE-C]	M. Davis, M. J. Dürst, http://www.unicode.org/unicode/reports/tr15/tr15-21.html , UNICODE Consortium
1681		
1682	[W3C-CHAR]	M. J. Dürst, <i>Requirements for String Identity Matching and String Indexing</i> http://www.w3.org/TR/WD-charreq , World Wide Web Consortium.
1683		
1684	[W3C-CharMod]	M. J. Dürst, <i>Unicode Normalization Forms</i> http://www.w3.org/TR/charmod/ , World Wide Web Consortium.
1685		
1686	[X.500]	ITU-T Recommendation X.501: <i>Information Technology - Open Systems Interconnection - The Directory: Models</i> , 1993.
1687		
1688	[XKMS]	W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp, XML Key Management Specification (XKMS), W3C Note 30 March 2001, http://www.w3.org/TR/xkms/ .
1689		
1690		
1691	[XML]	T. Bray et. al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> , http://www.w3.org/TR/REC-xml , World Wide Web Consortium.
1692		
1693	[XMLEnc]	<i>XML Encryption Specification</i> , In development.
1694	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , http://www.w3.org/TR/xmldsig-core/ , World Wide Web Consortium.
1695		
1696	[XMLSig-XSD]	XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd .
1697		
1698	[XTAML]	P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> , http://www.xmltrustcenter.org/ , VeriSign Inc. September 2001.
1699		

1700 **Appendix A. Acknowledgments**

1701 The editors would like to acknowledge the contributions of the OASIS SAML Technical Committee, whose
1702 voting members at the time of publication were:

- 1703 • Allen Rogers, Authentica
- 1704 • Irving Reid, Baltimore Technologies
- 1705 • Krishna Sankar, Cisco Systems
- 1706 • Ronald Jacobson, Computer Associates
- 1707 • Hal Lockhart, Entegriety
- 1708 • Carlisle Adams, Entrust Inc.
- 1709 • Robert Griffin, Entrust Inc.
- 1710 • Robert Zuccherato, Entrust Inc.
- 1711 • Don Flinn, Hitachi
- 1712 • Joe Pato, Hewlett-Packard (co-chair)
- 1713 • Jason Rouault, Hewlett-Packard
- 1714 • Marc Chanliau, Netegriety
- 1715 • Chris McLaren, Netegriety
- 1716 • Prateek Mishra, Netegriety
- 1717 • Charles Knouse, Oblix
- 1718 • Steve Anderson, OpenNetwork
- 1719 • Rob Philpott, RSA Security
- 1720 • Jahan Moreh, Sigaba
- 1721 • Bhavna Bhatnagar, Sun Microsystems
- 1722 • Jeff Hodges, Sun Microsystems (co-chair)
- 1723 • Eve Maler, Sun Microsystems (former chair)
- 1724 • Aravindan Ranganathan, Sun Microsystems
- 1725 • Emily Xu, Sun Microsystems
- 1726 • Bob Morgan, University of Washington and Internet2
- 1727 • Phillip Hallam-Baker, VeriSign

1728

Appendix B. Notices

1729 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1730 might be claimed to pertain to the implementation or use of the technology described in this document or
1731 the extent to which any license under such rights might or might not be available; neither does it
1732 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
1733 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
1734 made available for publication and any assurances of licenses to be made available, or the result of an
1735 attempt made to obtain a general license or permission for the use of such proprietary rights by
1736 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1737 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
1738 or other proprietary rights which may cover technology that may be required to implement this
1739 specification. Please address the information to the OASIS Executive Director.

1740 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS] 2001,
1741 2002. All Rights Reserved.

1742 This document and translations of it may be copied and furnished to others, and derivative works that
1743 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
1744 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
1745 and this paragraph are included on all such copies and derivative works. However, this document itself
1746 may not be modified in any way, such as by removing the copyright notice or references to OASIS,
1747 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
1748 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to
1749 translate it into languages other than English.

1750 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1751 or assigns.

1752 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1753 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1754 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1755 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.