



Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1

OASIS Standard, 2 September 2003

Document identifier:

oasis-sstc-saml-core-1.1

Location:

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

Editors:

Eve Maler, Sun Microsystems (eve.maler@sun.com)
Prateek Mishra, Netegrity, Inc. (pmishra@netegrity.com)
Rob Philpott, RSA Security (rphilpott@rsasecurity.com)

Contributors:

Stephen Farrell, Baltimore Technologies
Irving Reid, Baltimore Technologies
Hal Lockhart, BEA Systems
David Orchard, BEA Systems
Krishna Sankar, Cisco Systems
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Bob Blakley, IBM
Marlena Erdos, IBM
Scott Cantor, individual
RL "Bob" Morgan, individual
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Charles Knouse, Oblix
Simon Godik, Overkeer
Darren Platt, formerly of RSA Security
Jahan Moreh, Sigaba
Jeff Hodges, Sun Microsystems
Phillip Hallam-Baker, VeriSign (former editor)

Abstract:

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

39 **Status:**

40 This is an OASIS Standard document produced by the Security Services Technical Committee. It
41 was approved by the OASIS membership on 2 September 2003.

42 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
43 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them to the [security-services-](mailto:security-services-comment@lists.oasis-open.org)
44 [comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org) list (to post, you must subscribe; to subscribe, send a message to
45 security-services-comment-request@lists.oasis-open.org with "subscribe" in the body) or use
46 other OASIS-supported means of submitting comments. The committee will publish vetted errata
47 on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

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

Table of Contents

53	1	Introduction.....	6
54	1.1	Notation.....	6
55	1.2	Schema Organization and Namespaces.....	6
56	1.2.1	String and URI Values.....	7
57	1.2.2	Time Values.....	7
58	1.2.3	ID and ID Reference Values.....	7
59	1.2.4	Comparing SAML Values.....	7
60	1.3	SAML Concepts (Non-Normative).....	8
61	1.3.1	Overview.....	8
62	1.3.2	SAML and URI-Based Identifiers.....	9
63	1.3.3	SAML and Extensibility.....	10
64	2	SAML Assertions.....	11
65	2.1	Schema Header and Namespace Declarations.....	11
66	2.2	Simple Types.....	11
67	2.2.1	Simple Type DecisionType.....	12
68	2.3	Assertions.....	12
69	2.3.1	Element <AssertionIDReference>.....	12
70	2.3.2	Element <Assertion>.....	12
71	2.3.2.1	Element <Conditions>.....	14
72	2.3.2.1.1	Attributes NotBefore and NotOnOrAfter.....	15
73	2.3.2.1.2	Element <Condition>.....	15
74	2.3.2.1.3	Elements <AudienceRestrictionCondition> and <Audience>.....	15
75	2.3.2.1.4	Element <DoNotCacheCondition>.....	16
76	2.3.2.2	Element <Advice>.....	16
77	2.4	Statements.....	17
78	2.4.1	Element <Statement>.....	17
79	2.4.2	Element <SubjectStatement>.....	17
80	2.4.2.1	Element <Subject>.....	17
81	2.4.2.2	Element <NameIdentifier>.....	18
82	2.4.2.3	Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>.....	18
83	2.4.3	Element <AuthenticationStatement>.....	19
84	2.4.3.1	Element <SubjectLocality>.....	20
85	2.4.3.2	Element <AuthorityBinding>.....	20
86	2.4.4	Element <AttributeStatement>.....	21
87	2.4.4.1	Elements <AttributeDesignator> and <Attribute>.....	21
88	2.4.4.1.1	Element <AttributeValue>.....	22
89	2.4.5	Element <AuthorizationDecisionStatement>.....	22
90	2.4.5.1	Element <Action>.....	23
91	2.4.5.2	Element <Evidence>.....	24
92	3	SAML Protocol.....	25
93	3.1	Schema Header and Namespace Declarations.....	25
94	3.2	Requests.....	25

95	3.2.1 Complex Type RequestAbstractType	26
96	3.2.1.1 Element <RespondWith>	26
97	3.2.2 Element <Request>	27
98	3.2.2.1 Requests for Assertions by Reference	28
99	3.2.2.2 Element <AssertionArtifact>	28
100	3.3 Queries	28
101	3.3.1 Element <Query>	28
102	3.3.2 Element <SubjectQuery>	28
103	3.3.3 Element <AuthenticationQuery>	28
104	3.3.4 Element <AttributeQuery>	29
105	3.3.5 Element <AuthorizationDecisionQuery>	30
106	3.4 Responses	31
107	3.4.1 Complex Type ResponseAbstractType	31
108	3.4.2 Element <Response>	32
109	3.4.3 Element <Status>	33
110	3.4.3.1 Element <StatusCode>	33
111	3.4.3.2 Element <StatusMessage>	34
112	3.4.3.3 Element <StatusDetail>	35
113	3.4.4 Responses to Queries	35
114	4 SAML Versioning	36
115	4.1 SAML Specification Set Version	36
116	4.1.1 Schema Version	36
117	4.1.2 SAML Assertion Version	36
118	4.1.3 SAML Protocol Version	37
119	4.1.3.1 Request Version	37
120	4.1.4 Response Version	37
121	4.1.5 Permissible Version Combinations	37
122	4.2 SAML Namespace Version	38
123	4.2.1 Schema Evolution	38
124	5 SAML and XML Signature Syntax and Processing	39
125	5.1 Signing Assertions	39
126	5.2 Request/Response Signing	40
127	5.3 Signature Inheritance	40
128	5.4 XML Signature Profile	40
129	5.4.1 Signing Formats and Algorithms	40
130	5.4.2 References	40
131	5.4.3 Canonicalization Method	40
132	5.4.4 Transforms	41
133	5.4.5 KeyInfo	41
134	5.4.6 Binding Between Statements in a Multi-Statement Assertion	41
135	5.4.7 Interoperability with SAML V1.0	41
136	5.4.8 Example	41
137	6 SAML Extensions	44
138	6.1 Assertion Schema Extension	44
139	6.2 Protocol Schema Extension	44

140	6.3 Use of Type Derivation and Substitution Groups	45
141	7 SAML-Defined Identifiers	46
142	7.1 Authentication Method Identifiers	46
143	7.1.1 Password.....	46
144	7.1.2 Kerberos	46
145	7.1.3 Secure Remote Password (SRP).....	46
146	7.1.4 Hardware Token.....	47
147	7.1.5 SSL/TLS Certificate Based Client Authentication:	47
148	7.1.6 X.509 Public Key	47
149	7.1.7 PGP Public Key	47
150	7.1.8 SPKI Public Key	47
151	7.1.9 XKMS Public Key	47
152	7.1.10 XML Digital Signature.....	47
153	7.1.11 Unspecified.....	47
154	7.2 Action Namespace Identifiers.....	47
155	7.2.1 Read/Write/Execute/Delete/Control	48
156	7.2.2 Read/Write/Execute/Delete/Control with Negation	48
157	7.2.3 Get/Head/Put/Post	48
158	7.2.4 UNIX File Permissions	48
159	7.3 NameIdentifier Format Identifiers	49
160	7.3.1 Unspecified.....	49
161	7.3.2 Email Address	49
162	7.3.3 X.509 Subject Name	49
163	7.3.4 Windows Domain Qualified Name.....	49
164	8 References	50
165	Appendix A. Acknowledgments	52
166	Appendix B. Notices.....	53
167		

1 Introduction

This specification defines the syntax and semantics for XML-encoded Security Assertion Markup Language (SAML) assertions, protocol requests, and protocol responses. These constructs are typically embedded in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML specification for bindings and profiles [**SAMLBind**] provides frameworks for this embedding and transport. Files containing just the SAML assertion schema [**SAML-XSD**] and protocol schema [**SAML-P-XSD**] are available.

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

1.1 Notation

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

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

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

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

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

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

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

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

- The prefix `saml`: stands for the SAML assertion namespace.
- The prefix `samlp`: stands for the SAML request-response protocol namespace.
- The prefix `ds`: stands for the W3C XML Signature namespace [**XMLSig-XSD**].
- The prefix `xsd`: stands for the W3C XML Schema namespace [**Schema1**] in example listings. In schema listings, this is the default namespace and no prefix is shown.

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

1.2 Schema Organization and Namespaces

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

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

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

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

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

211 <http://www.w3.org/2000/09/xmldsig#>

212 See Section 4.2 for information on SAML namespace versioning.

213 1.2.1 String and URI Values

214 All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively, which
215 are built in to the W3C XML Schema Datatypes specification [**Schema2**]. All strings in SAML messages
216 MUST consist of at least one non-whitespace character (whitespace is defined in the XML
217 Recommendation [**XML**] §2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise
218 indicated in this specification, all URI reference values MUST consist of at least one non-whitespace
219 character, and are strongly RECOMMENDED to be absolute [**RFC 2396**].

220 1.2.2 Time Values

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

223 SAML system entities SHOULD NOT rely on other applications supporting time resolution finer than
224 milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

225 1.2.3 ID and ID Reference Values

226 The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses.
227 Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties:

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

232 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the
233 implementation. In the case that a pseudorandom technique is employed, the probability of two randomly
234 chosen identifiers being identical MUST be less than or equal to 2^{-128} and SHOULD be less than or equal
235 to 2^{-160} . This requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in
236 length. The encoding must conform to the rules defining the **xsd:ID** datatype.

237 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that
238 **xsd:IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML reference
239 identifier might actually be defined in a document separate from that in which the identifier reference is
240 used. XML [**XML**] requires that names of type **xsd:IDREF** must match the value of an ID attribute on
241 some element in the same XML document.

242 1.2.4 Comparing SAML Values

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

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

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

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

263 **1.3 SAML Concepts (Non-Normative)**

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

266 **1.3.1 Overview**

267 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security
268 information. This security information is expressed in the form of assertions about subjects, where a
269 subject is an entity (either human or computer) that has an identity in some security domain. A typical
270 example of a subject is a person, identified by his or her email address in a particular Internet DNS
271 domain.

272 Assertions can convey information about authentication acts that were previously performed by subjects,
273 attributes of subjects, and authorization decisions about whether subjects are allowed to access certain
274 resources. A single assertion might contain several different internal statements about authentication,
275 authorization, and attributes.

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

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

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

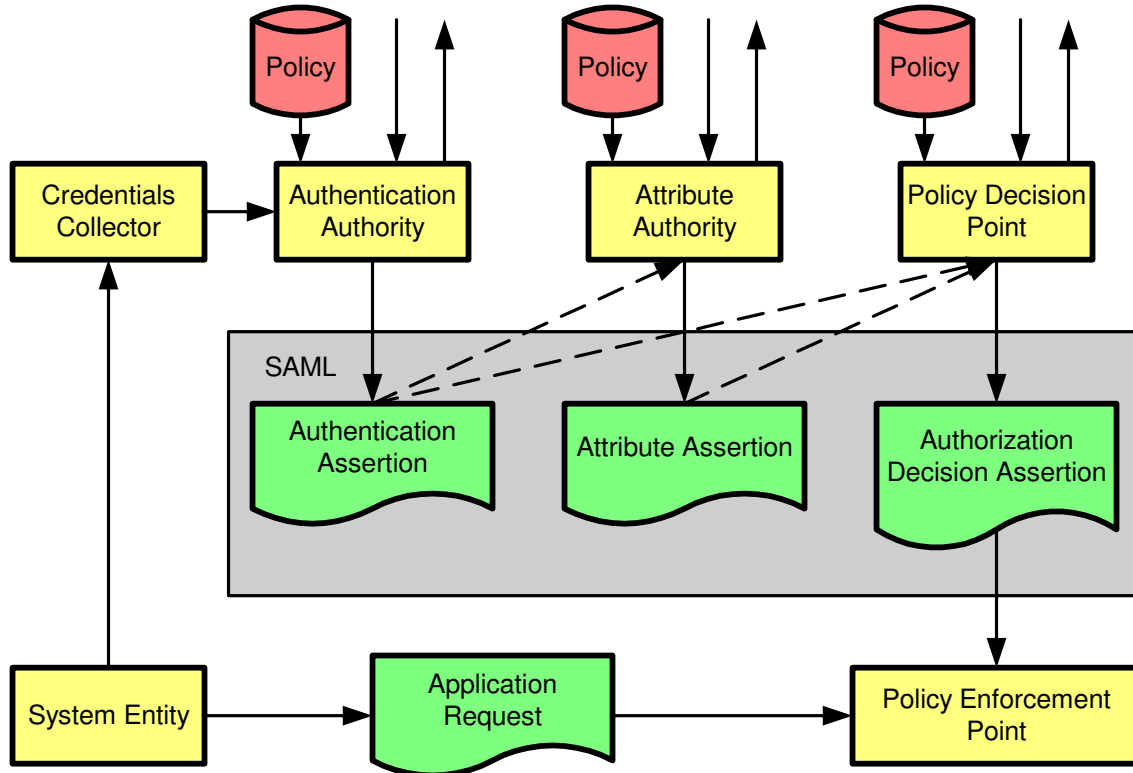


Figure 1 The SAML Domain Model

286
287

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

292 The assertion and protocol data formats are defined in this specification. The bindings and profiles are
293 defined in a separate specification [SAMLBind]. A conformance program for SAML is defined in the
294 conformance specification [SAMLConform]. Security issues are discussed in a separate security and
295 privacy considerations specification [SAMLSecure].

296 1.3.2 SAML and URI-Based Identifiers

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

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

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

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

303 These identifiers are defined as Uniform Resource Identifier (URI) references, but they are not
304 necessarily able to be resolved to some Web resource. At times, SAML authorities need to use identifier
305 strings of their own design, for example to define additional kinds of authentication methods not covered
306 by SAML-defined identifiers. In the case where a form is used that is compatible with interpretation as a
307 URI reference, it is not required to be resolvable to some Web resource. However, using URI references
308 – particularly URLs based on the `http:` scheme or URNs based on the `urn:` scheme – is likely to
309 mitigate problems with clashing identifiers to some extent.

310 The Read/Write/Execute/Delete/Control identifier above is an example of a namespace (not in the sense
311 of an XML namespace). SAML uses this namespace mechanism to manage the universe of possible
312 types of actions and possible names of attributes.
313 See Section 7 for a list of SAML-defined identifiers.

314 **1.3.3 SAML and Extensibility**

315 The XML formats for SAML assertions and protocol messages have been designed to be extensible.
316 Section 6 describes SAML's design for extensibility in more detail.
317 However, it is possible that the use of extensions will harm interoperability and therefore the use of
318 extensions should be carefully considered.

319 2 SAML Assertions

320 An assertion is a package of information that supplies one or more statements made by a SAML
321 authority. This SAML specification defines three different kinds of assertion statement that can be created
322 by a SAML authority. As mentioned above and described in Section 6, extensions are permitted by the
323 SAML assertion schema, allowing user-defined extensions to assertions and SAML statements, as well
324 as allowing the definition of new kinds of assertion statement. The three kinds of statement defined in this
325 specification are:

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

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

333 2.1 Schema Header and Namespace Declarations

334 The following schema fragment defines the XML namespaces and other header information for the
335 assertion schema:

```
336 <schema  
337   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
338   xmlns="http://www.w3.org/2001/XMLSchema"  
339   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
340   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
341   elementFormDefault="unqualified"  
342   attributeFormDefault="unqualified"  
343   version="1.1">  
344   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
345     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
346     schema.xsd"/>  
347   <annotation>  
348     <documentation>  
349       Document identifier: oasis-sstc-saml-schema-assertion-1.1  
350       Location: http://www.oasis-  
351       open.org/committees/documents.php?wg_abbrev=security  
352       Revision history:  
353       V1.0 (November, 2002):  
354         Initial standard schema.  
355       V1.1 (September, 2003):  
356         * Note that V1.1 of this schema has the same XML  
357         namespace as V1.0.  
358         Rebased ID content directly on XML Schema types  
359         Added DoNotCacheCondition element and  
360         DoNotCacheConditionType  
361     </documentation>  
362   </annotation>  
363   ...  
364 </schema>
```

365 2.2 Simple Types

366 The following section(s) define the SAML assertion-related simple types.

367 2.2.1 Simple Type DecisionType

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

370 Permit

371 The specified action is permitted.

372 Deny

373 The specified action is denied.

374 Indeterminate

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

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

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

```
380 <simpleType name="DecisionType">  
381   <restriction base="string">  
382     <enumeration value="Permit"/>  
383     <enumeration value="Deny"/>  
384     <enumeration value="Indeterminate"/>  
385   </restriction>  
386 </simpleType>
```

387 2.3 Assertions

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

389 2.3.1 Element <AssertionIDReference>

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

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

```
392 <element name="AssertionIDReference" type="NCName"/>
```

393 2.3.2 Element <Assertion>

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

396 MajorVersion [Required]
 397 The major version of this assertion. The identifier for the version of SAML defined in this specification
 398 is 1. SAML versioning is discussed in Section 4.

399 MinorVersion [Required]
 400 The minor version of this assertion. The identifier for the version of SAML defined in this specification
 401 is 1. SAML versioning is discussed in Section 4.

402 AssertionID [Required]
 403 The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in
 404 Section 1.2.3 for identifier uniqueness.

405 Issuer [Required]
 406 The SAML authority that created the assertion. The name of the issuer is provided as a string. The
 407 issuer name SHOULD be unambiguous to the intended relying parties. SAML authorities may use an
 408 identifier such as a URI reference that is designed to be unambiguous regardless of context.

409 IssueInstant [Required]
 410 The time instant of issue in UTC, as described in Section 1.2.2.

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

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

416 <ds:Signature> [Optional]
 417 An XML Signature that authenticates the assertion, as described in Section 5.

418 One or more of the following statement elements:

419 <Statement>
 420 A statement defined in an extension schema.

421 <SubjectStatement>
 422 A subject statement defined in an extension schema.

423 <AuthenticationStatement>
 424 An authentication statement.

425 <AuthorizationDecisionStatement>
 426 An authorization decision statement.

427 <AttributeStatement>
 428 An attribute statement.

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

```

430 <element name="Assertion" type="saml:AssertionType"/>
431 <complexType name="AssertionType">
432   <sequence>
433     <element ref="saml:Conditions" minOccurs="0"/>
434     <element ref="saml:Advice" minOccurs="0"/>
435     <choice maxOccurs="unbounded">
436       <element ref="saml:Statement"/>
437       <element ref="saml:SubjectStatement"/>
438       <element ref="saml:AuthenticationStatement"/>
439       <element ref="saml:AuthorizationDecisionStatement"/>
440       <element ref="saml:AttributeStatement"/>
441     </choice>
442     <element ref="ds:Signature" minOccurs="0"/>
  
```

```

443     </sequence>
444     <attribute name="MajorVersion" type="integer" use="required"/>
445     <attribute name="MinorVersion" type="integer" use="required"/>
446     <attribute name="AssertionID" type="ID" use="required"/>
447     <attribute name="Issuer" type="string" use="required"/>
448     <attribute name="IssueInstant" type="dateTime" use="required"/>
449 </complexType>

```

450 2.3.2.1 Element <Conditions>

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

452 NotBefore [Optional]

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

455 NotOnOrAfter [Optional]

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

458 <Condition> [Any Number]

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

460 <AudienceRestrictionCondition> [Any Number]

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

462 <DoNotCacheCondition> [Any Number]

463 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future use.

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

```

466 <element name="Conditions" type="saml:ConditionsType"/>
467 <complexType name="ConditionsType">
468   <choice minOccurs="0" maxOccurs="unbounded">
469     <element ref="saml:AudienceRestrictionCondition"/>
470     <element ref="saml:DoNotCacheCondition"/>
471     <element ref="saml:Condition"/>
472   </choice>
473   <attribute name="NotBefore" type="dateTime" use="optional"/>
474   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
475 </complexType>

```

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

- 480 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
481 considered to be **Valid**.
- 482 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
483 assertion is **Invalid**.
- 484 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity
485 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 486 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the
487 assertion is considered to be **Valid**.

488 The <Conditions> element MAY be extended to contain additional conditions. If an element contained
489 within a <Conditions> element is encountered that is not understood, the status of the condition cannot

490 be evaluated and the validity status of the assertion MUST be deemed to be **Indeterminate** in
491 accordance with rule 3 above.
492 Note that an assertion that has validity status **Valid** may not be trustworthy for reasons such as not being
493 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

494 2.3.2.1.1 Attributes **NotBefore** and **NotOnOrAfter**

495 The **NotBefore** and **NotOnOrAfter** attributes specify time limits on the validity of the assertion.

496 The **NotBefore** attribute specifies the time instant at which the validity interval begins. The
497 **NotOnOrAfter** attribute specifies the time instant at which the validity interval has ended.

498 If the value for either **NotBefore** or **NotOnOrAfter** is omitted it is considered unspecified. If the
499 **NotBefore** attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the
500 assertion is valid at any time before the time instant specified by the **NotOnOrAfter** attribute. If the
501 **NotOnOrAfter** attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),
502 the assertion is valid from the time instant specified by the **NotBefore** attribute with no expiry. If neither
503 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is valid
504 at any time.

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

509 2.3.2.1.2 Element **<Condition>**

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

512 The following schema fragment defines the **<Condition>** element and its **ConditionAbstractType**
513 complex type:

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

516 2.3.2.1.3 Elements **<AudienceRestrictionCondition>** and **<Audience>**

517 The **<AudienceRestrictionCondition>** element specifies that the assertion is addressed to one or
518 more specific audiences identified by **<Audience>** elements. Although a SAML relying party that is
519 outside the audiences specified is capable of drawing conclusions from an assertion, the SAML authority
520 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the
521 following elements:

522 **<Audience>**

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

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

527 The SAML authority cannot prevent a party to whom the assertion is disclosed from taking action on the
528 basis of the information provided. However, the **<AudienceRestrictionCondition>** element allows
529 the SAML authority to state explicitly that no warranty is provided to such a party in a machine- and
530 human-readable form. While there can be no guarantee that a court would uphold such a warranty
531 exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably
532 improved.

533 The following schema fragment defines the **<AudienceRestrictionCondition>** element and its
534 **AudienceRestrictionConditionType** complex type:

```

535 <element name="AudienceRestrictionCondition"
536       type="saml:AudienceRestrictionConditionType"/>
537 <complexType name="AudienceRestrictionConditionType">
538   <complexContent>
539     <extension base="saml:ConditionAbstractType">
540       <sequence>
541         <element ref="saml:Audience" maxOccurs="unbounded"/>
542       </sequence>
543     </extension>
544   </complexContent>
545 </complexType>
546 <element name="Audience" type="anyURI"/>

```

547 2.3.2.1.4 Element <DoNotCacheCondition>

548 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be
549 retained for future use. A SAML authority SHOULD NOT include more than one
550 <DoNotCacheCondition> element within a <Conditions> element of an assertion. Note that no
551 Relying Party implementation is required to perform caching. However, any that do so MUST observe this
552 condition. If multiple <DoNotCacheCondition> elements appear within a <Conditions> element, a
553 Relying Party MUST treat the multiple elements as though a single <DoNotCacheCondition> element
554 was specified. For the purposes of determining the validity of the <Conditions> element, the
555 <DoNotCacheCondition> (see Section 2.3.2.1) is considered to always be valid.

556

```

557 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType" />
558 <complexType name="DoNotCacheConditionType">
559   <complexContent>
560     <extension base="saml:ConditionAbstractType"/>
561   </complexContent>
562 </complexType>

```

563 2.3.2.2 Element <Advice>

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

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

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

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

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

```

576 <element name="Advice" type="saml:AdviceType"/>
577 <complexType name="AdviceType">
578   <choice minOccurs="0" maxOccurs="unbounded">
579     <element ref="saml:AssertionIDReference"/>
580     <element ref="saml:Assertion"/>
581     <any namespace="##other" processContents="lax"/>
582   </choice>
583 </complexType>

```


584 2.4 Statements

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

586 2.4.1 Element <Statement>

587 The <Statement> element is an extension point that allows other assertion-based applications to reuse
588 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable
589 only as the base of a derived type.

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

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

594 2.4.2 Element <SubjectStatement>

595 The <SubjectStatement> element is an extension point that allows other assertion-based applications
596 to reuse the SAML assertion framework. It contains a <Subject> element that allows a SAML authority
597 to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends
598 **StatementAbstractType**, is abstract and is thus usable only as the base of a derived type.

599 The following schema fragment defines the <SubjectStatement> element and its
600 **SubjectStatementAbstractType** abstract type:

```
601 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
602 <complexType name="SubjectStatementAbstractType" abstract="true">  
603   <complexContent>  
604     <extension base="saml:StatementAbstractType">  
605       <sequence>  
606         <element ref="saml:Subject"/>  
607       </sequence>  
608     </extension>  
609   </complexContent>  
610 </complexType>
```

611 2.4.2.1 Element <Subject>

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

614 <NameIdentifier>

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

616 <SubjectConfirmation>

617 Information that allows the subject to be authenticated.

618 If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the
619 SAML authority is asserting that if the SAML relying party performs the specified

620 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying
621 party is the entity that the SAML authority associates with the <NameIdentifier>. A <Subject>
622 element SHOULD NOT identify more than one principal.

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

```
624 <element name="Subject" type="saml:SubjectType"/>  
625 <complexType name="SubjectType">  
626   <choice>  
627     <sequence>  
628       <element ref="saml:NameIdentifier"/>
```

629
630
631
632
633

```
        <element ref="saml:SubjectConfirmation" minOccurs="0"/>
    </sequence>
    <element ref="saml:SubjectConfirmation"/>
</choice>
</complexType>
```

634 **2.4.2.2 Element <NameIdentifier>**

635 The <NameIdentifier> element specifies a subject by a combination of a name qualifier, a name,
636 and a format. The name is provided as element content. The <NameIdentifier> element has the
637 following attributes:

638 NameQualifier [Optional]

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

641 Format [Optional]

642 A URI reference representing the format in which the <NameIdentifier> information is provided.
643 See Section 7.3 for some URI references that MAY be used as the value of the Format attribute. If
644 the Format attribute is not included, the identifier urn:oasis:names:tc:SAML:1.0:nameid-
645 format:unspecified (see Section 7.3.1) is in effect. Regardless of format, issues of anonymity,
646 pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties
647 are implementation-specific.

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

```
650 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
651 <complexType name="NameIdentifierType">
652   <simpleContent>
653     <extension base="string">
654       <attribute name="NameQualifier" type="string" use="optional"/>
655       <attribute name="Format" type="anyURI" use="optional"/>
656     </extension>
657   </simpleContent>
658 </complexType>
```

659 When a Format other than those specified in Section 7.3 is used, the NameQualifier attribute and the
660 <NameIdentifier> element's content are to be interpreted according to the specification of that format
661 as defined outside of this specification.

662 **2.4.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 663 **<SubjectConfirmationData>**

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

666 <ConfirmationMethod> [One or more]
667 A URI reference that identifies a protocol to be used to authenticate the subject. URI references
668 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the
669 SAML bindings and profiles specification [**SAMLBind**]. Additional methods may be added by defining
670 new profiles or by private agreement.

671 <SubjectConfirmationData> [Optional]
672 Additional authentication information to be used by a specific authentication protocol.

673 <ds:KeyInfo> [Optional]
674 An XML Signature [**XMLSig**] element that provides access to a cryptographic key held by the subject.

675 The following schema fragment defines the <SubjectConfirmation> element and its
676 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData> element and
677 the <ConfirmationMethod> element:

```
678 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
679 <complexType name="SubjectConfirmationType">  
680 <sequence>  
681 <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
682 <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
683 <element ref="ds:KeyInfo" minOccurs="0"/>  
684 </sequence>  
685 </complexType>  
686 <element name="SubjectConfirmationData" type="anyType"/>  
687 <element name="ConfirmationMethod" type="anyURI"/>
```

688 2.4.3 Element <AuthenticationStatement>

689 The <AuthenticationStatement> element describes a statement by the SAML authority asserting
690 that the statement's subject was authenticated by a particular means at a particular time. It is of type
691 **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the addition of the
692 following elements and attributes:

693 AuthenticationMethod [Required]
694 A URI reference that specifies the type of authentication that took place. URI references identifying
695 common authentication protocols are listed in Section 7.1.

696 AuthenticationInstant [Required]
697 Specifies the time at which the authentication took place. The time value is encoded in UTC as
698 described in Section 1.2.2.

699 <SubjectLocality> [Optional]
700 Specifies the DNS domain name and IP address for the system entity from which the subject was
701 apparently authenticated.

702 <AuthorityBinding> [Any Number]
703 Indicates that additional information about the subject of the statement may be available.

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

```
706 <element name="AuthenticationStatement"  
707 <complexType name="AuthenticationStatementType">  
708 <complexContent>  
709 <extension base="saml:SubjectStatementAbstractType">  
710 <sequence>  
711 <element ref="saml:SubjectLocality" minOccurs="0"/>  
712
```

```

713         <element ref="saml:AuthorityBinding"
714             minOccurs="0" maxOccurs="unbounded"/>
715     </sequence>
716     <attribute name="AuthenticationMethod" type="anyURI"
717 use="required"/>
718     <attribute name="AuthenticationInstant" type="dateTime"
719 use="required"/>
720 </extension>
721 </complexContent>
722 </complexType>

```

723 2.4.3.1 Element <SubjectLocality>

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

726 IPAddress [Optional]

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

728 DNSAddress [Optional]

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

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

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

```

734 <element name="SubjectLocality"
735     type="saml: SubjectLocalityType"/>
736 <complexType name="SubjectLocalityType">
737     <attribute name="IPAddress" type="string" use="optional"/>
738     <attribute name="DNSAddress" type="string" use="optional"/>
739 </complexType>

```

740 2.4.3.2 Element <AuthorityBinding>

741 The <AuthorityBinding> element MAY be used to indicate to a SAML relying party processing an
742 AuthenticationStatement that a SAML authority may be available to provide additional information about
743 the subject of the statement. A single SAML authority may advertise its presence over multiple protocol
744 bindings, at multiple locations, and as more than one kind of authority by sending multiple elements as
745 needed.

746 NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be
747 removed in the next major version of SAML.

748 The <AuthorityBinding> element has the following attributes:

749 AuthorityKind [Required]

750 The type of SAML protocol queries to which the authority described by this element will respond. The
751 value is specified as an XML Schema QName. The AuthorityKind value is either the QName of the
752 desired SAML protocol query element or, in the case of an extension schema, the QName of the
753 SAML **QueryAbstractType** complex type or some extension type that was derived from it. In the
754 case of an extension schema, the authority will respond to all query elements of the specified type.

755 For example, an attribute authority would be identified by

756 AuthorityKind="samlp:AttributeQuery", where there is a namespace declaration in the
757 scope of this attribute that binds the samlp: prefix to the SAML protocol namespace.

758 Location [Required]

759 A URI reference describing how to locate and communicate with the authority, the exact syntax of

760 which depends on the protocol binding in use. For example, a binding based on HTTP will be a web
761 URL, while a binding based on SMTP might use the `mailto:` scheme.

762 Binding [Required]

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

765 The following schema fragment defines the `<AuthorityBinding>` element and its
766 **AuthorityBindingType** complex type:

```
767 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>  
768 <complexType name="AuthorityBindingType">  
769   <attribute name="AuthorityKind" type="QName" use="required"/>  
770   <attribute name="Location" type="anyURI" use="required"/>  
771   <attribute name="Binding" type="anyURI" use="required"/>  
772 </complexType>
```

773 2.4.4 Element `<AttributeStatement>`

774 The `<AttributeStatement>` element describes a statement by the SAML authority asserting that the
775 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,
776 which extends **SubjectStatementAbstractType** with the addition of the following element:

777 `<Attribute>` [One or More]

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

779 The following schema fragment defines the `<AttributeStatement>` element and its
780 **AttributeStatementType** complex type:

```
781 <element name="AttributeStatement" type="saml:AttributeStatementType"/>  
782 <complexType name="AttributeStatementType">  
783   <complexContent>  
784     <extension base="saml:SubjectStatementAbstractType">  
785       <sequence>  
786         <element ref="saml:Attribute" maxOccurs="unbounded"/>  
787       </sequence>  
788     </extension>  
789   </complexContent>  
790 </complexType>
```

791 2.4.4.1 Elements `<AttributeDesignator>` and `<Attribute>`

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

796 AttributeNamespace [Required]

797 The namespace in which the `AttributeName` elements are interpreted.

798 AttributeName [Required]

799 The name of the attribute.

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

```
802 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
803 <complexType name="AttributeDesignatorType">  
804   <attribute name="AttributeName" type="string" use="required"/>  
805   <attribute name="AttributeNamespace" type="anyURI" use="required"/>  
806 </complexType>
```

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

810 <AttributeValue> [Any Number]

811 The value of the attribute.

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

```
813 <element name="Attribute" type="saml:AttributeType"/>
814 <complexType name="AttributeType">
815   <complexContent>
816     <extension base="saml:AttributeDesignatorType">
817       <sequence>
818         <element ref="saml:AttributeValue"
819           maxOccurs="unbounded"/>
820       </sequence>
821     </extension>
822   </complexContent>
823 </complexType>
```

824 2.4.4.1.1 Element <AttributeValue>

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

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

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

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

833 2.4.5 Element <AuthorizationDecisionStatement>

834 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority
835 asserting that a request for access by the statement's subject to the specified resource has resulted in the
836 specified authorization decision on the basis of some optionally specified evidence.

837 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
838 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in
839 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
840 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
841 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

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

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

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

851 Inconsistent URI reference interpretation can also result from differences between the URI reference
852 syntax and the semantics of an underlying file system. Particular care is required if URI references are

- 853 employed to specify an access control policy language. The following security conditions should be
 854 satisfied by the system which employs SAML assertions:
- 855 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
 856 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
 857 part of the resource URI reference.
 - 858 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users to
 859 establish logical equivalences between file system entries. A requester SHOULD NOT be able to gain
 860 access to a denied resource by creating such an equivalence.

861 The `<AuthorizationDecisionStatement>` element is of type
 862 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the
 863 addition of the following elements (in order) and attributes:

864 **Resource** [Required]

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

868 **Decision** [Required]

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

871 **<Action>** [One or more]

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

873 **<Evidence>** [Optional]

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

875 The following schema fragment defines the `<AuthorizationDecisionStatement>` element and its
 876 **AuthorizationDecisionStatementType** complex type:

```

877 <element name="AuthorizationDecisionStatement"
878 type="saml:AuthorizationDecisionStatementType"/>
879 <complexType name="AuthorizationDecisionStatementType">
880   <complexContent>
881     <extension base="saml:SubjectStatementAbstractType">
882       <sequence>
883         <element ref="saml:Action" maxOccurs="unbounded"/>
884         <element ref="saml:Evidence" minOccurs="0"/>
885       </sequence>
886       <attribute name="Resource" type="anyURI" use="required"/>
887       <attribute name="Decision" type="saml:DecisionType"
888 use="required"/>
889     </extension>
890   </complexContent>
891 </complexType>
  
```

892 2.4.5.1 Element `<Action>`

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

895 Namespace [Optional]

896 A URI reference representing the namespace in which the name of the specified action is to be
897 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwe-dc-
898 negation specified in Section 7.2.2 is in effect.

899 *string data* [Required]

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

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

```
902 <element name="Action" type="saml:ActionType"/>  
903 <complexType name="ActionType">  
904   <simpleContent>  
905     <extension base="string">  
906       <attribute name="Namespace" type="anyURI"/>  
907     </extension>  
908   </simpleContent>  
909 </complexType>
```

910 2.4.5.2 Element <Evidence>

911 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on
912 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains one of the
913 following elements:

914 <AssertionIDReference>

915 Specifies an assertion by reference to the value of the assertion's *AssertionID* attribute.

916 <Assertion>

917 Specifies an assertion by value.

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

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

```
924 <element name="Evidence" type="saml:EvidenceType"/>  
925 <complexType name="EvidenceType">  
926   <choice maxOccurs="unbounded">  
927     <element ref="saml:AssertionIDReference"/>  
928     <element ref="saml:Assertion"/>  
929   </choice>  
930 </complexType>
```


931

3 SAML Protocol

932 SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings and
933 profiles specification for SAML [**SAMlBind**] describes specific means of transporting assertions using
934 existing widely deployed protocols.

935 SAML-aware requesters MAY in addition use the SAML request-response protocol defined by the
936 <Request> and <Response> elements. The requester sends a <Request> element to a SAML
937 responder, and the responder generates a <Response> element, as shown in Figure 2.



938

939

Figure 2: SAML Request-Response Protocol

940 3.1 Schema Header and Namespace Declarations

941 The following schema fragment defines the XML namespaces and other header information for the
942 protocol schema:

```
943 <schema  
944   targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"  
945   xmlns="http://www.w3.org/2001/XMLSchema"  
946   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
947   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
948   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
949   elementFormDefault="unqualified"  
950   attributeFormDefault="unqualified"  
951   version="1.1">  
952   <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"  
953     schemaLocation="oasis-sstc-saml-schema-assertion-1.1.xsd"/>  
954   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
955     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
956   schema.xsd "/>  
957   <annotation>  
958     <documentation>  
959       Document identifier: oasis-sstc-saml-schema-protocol-1.1  
960       Location: http://www.oasis-  
961   open.org/committees/documents.php?wg_abbrev=security  
962       Revision history:  
963       V1.0 (November, 2002):  
964         Initial standard schema.  
965       V1.1 (September, 2003):  
966         * Note that V1.1 of this schema has the same XML  
967         namespace as V1.0.  
968         Rebased ID content directly on XML Schema types  
969     </documentation>  
970   </annotation>  
971   ...  
972 </schema>
```

973 3.2 Requests

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

975 **3.2.1 Complex Type RequestAbstractType**

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

978 **RequestID** [Required]

979 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in
980 Section 1.2.3 for identifier uniqueness. The values of the **RequestID** attribute in a request and the
981 **InResponseTo** attribute in the corresponding response MUST match.

982 **MajorVersion** [Required]

983 The major version of this request. The identifier for the version of SAML defined in this specification is
984 1. SAML versioning is discussed in Section 4.

985 **MinorVersion** [Required]

986 The minor version of this request. The identifier for the version of SAML defined in this specification is
987 1. SAML versioning is discussed in Section 4.

988 **IssueInstant** [Required]

989 The time instant of issue of the request. The time value is encoded in UTC as described in Section
990 1.2.2.

991 **<RespondWith>** [Any Number]

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

993 **<ds:Signature>** [Optional]

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

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

```
996 <complexType name="RequestAbstractType" abstract="true">  
997   <sequence>  
998     <element ref="samlp:RespondWith"  
999       minOccurs="0" maxOccurs="unbounded"/>  
1000     <element ref="ds:Signature" minOccurs="0"/>  
1001   </sequence>  
1002   <attribute name="RequestID" type="ID" use="required"/>  
1003   <attribute name="MajorVersion" type="integer" use="required"/>  
1004   <attribute name="MinorVersion" type="integer" use="required"/>  
1005   <attribute name="IssueInstant" type="dateTime" use="required"/>  
1006 </complexType>
```

1007 **3.2.1.1 Element <RespondWith>**

1008 The **<RespondWith>** element specifies the type of statement the SAML relying party wants from the
1009 SAML authority. Multiple **<RespondWith>** elements MAY be included to indicate that the relying party
1010 will accept assertions containing any of the specified types. If no **<RespondWith>** element is given, the
1011 SAML authority MAY return assertions containing statements of any type.

1012 NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be
1013 removed in the next major version of SAML.

1014 If the **<Request>** element contains one or more **<RespondWith>** elements, the SAML authority MUST
1015 NOT respond with assertions containing statements of any type not specified in one of the
1016 **<RespondWith>** elements.

1017 Inability to find assertions that meet **<RespondWith>** criteria should be treated as identical to any other
1018 query for which no assertions are available. In both cases a status of success MUST be returned in the
1019 Response message, but no assertions will be included.

1020 The content of each <RespondWith> element is an XML QName. The <RespondWith> content is
1021 either the QName of the desired SAML statement element name or, in the case of an extension schema,
1022 it is the QName of the SAML **StatementAbstractType** complex type or some type that was derived from
1023 it. In the case of an extension schema, all statements of the specified type are requested.

1024 For example, a relying party that wishes to receive assertions containing only attribute statements would
1025 specify <RespondWith>saml:AttributeStatement</RespondWith>, where the prefix is bound to
1026 the SAML assertion namespace in a namespace declaration that is in the scope of this element.

1027 The following schema fragment defines the <RespondWith> element:

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

1029 3.2.2 Element <Request>

1030 The <Request> element specifies a SAML request. It provides either a query or a request for a specific
1031 assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has the complex
1032 type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following
1033 elements:

1034 <Query>

An extension point that allows extension schemas to define new types of query.

1036 <SubjectQuery>

An extension point that allows extension schemas to define new types of query that specify a single SAML subject.

1039 <AuthenticationQuery>

Makes a query for authentication information.

1041 <AttributeQuery>

Makes a query for attribute information.

1043 <AuthorizationDecisionQuery>

Makes a query for an authorization decision.

1045 <AssertionIDReference> [One or more]

Requests an assertion by reference to the value of its AssertionID attribute.

1047 <AssertionArtifact> [One or more]

Requests assertions by supplying an assertion artifact that represents it.

1049 The following schema fragment defines the <Request> element and its **RequestType** complex type:

```
<element name="Request" type="saml:RequestType"/>
<complexType name="RequestType">
  <complexContent>
    <extension base="saml:RequestAbstractType">
      <choice>
        <element ref="saml:Query"/>
        <element ref="saml:SubjectQuery"/>
        <element ref="saml:AuthenticationQuery"/>
        <element ref="saml:AttributeQuery"/>
        <element ref="saml:AuthorizationDecisionQuery"/>
        <element ref="saml:AssertionIDReference"
maxOccurs="unbounded"/>
        <element ref="saml:AssertionArtifact"
maxOccurs="unbounded"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

1068 3.2.2.1 Requests for Assertions by Reference

1069 In the context of a <Request> element, the <saml:AssertionIDReference> element is used to
1070 request an assertion by means of its ID. See Section 2.3.1 for more information on this element.

1071 3.2.2.2 Element <AssertionArtifact>

1072 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
1073 assertion being requested. Its use is governed by the specific profile of SAML that is being used; see the
1074 SAML specification for bindings and profiles [SAMLBind] for more information on the use of assertion
1075 artifacts in profiles.

1076 The following schema fragment defines the <AssertionArtifact> element:

```
1077 <element name="AssertionArtifact" type="string"/>
```

1078 3.3 Queries

1079 The following sections define the SAML constructs that contain query information.

1080 3.3.1 Element <Query>

1081 The <Query> element is an extension point that allows new SAML queries to be defined. Its
1082 **QueryAbstractType** is abstract and is thus usable only as the base of a derived type.
1083 **QueryAbstractType** is the base type from which all SAML query elements are derived.

1084 The following schema fragment defines the <Query> element and its **QueryAbstractType** complex type:

```
1085 <element name="Query" type="saml:QueryAbstractType"/>  
1086 <complexType name="QueryAbstractType" abstract="true"/>
```

1087 3.3.2 Element <SubjectQuery>

1088 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a single
1089 SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and is thus usable only as the
1090 base of a derived type. **SubjectQueryAbstractType** adds the <Subject> element.

1091 The following schema fragment defines the <SubjectQuery> element and its
1092 **SubjectQueryAbstractType** complex type:

```
1093 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>  
1094 <complexType name="SubjectQueryAbstractType" abstract="true">  
1095   <complexContent>  
1096     <extension base="saml:QueryAbstractType">  
1097       <sequence>  
1098         <element ref="saml:Subject"/>  
1099       </sequence>  
1100     </extension>  
1101   </complexContent>  
1102 </complexType>
```

1103 3.3.3 Element <AuthenticationQuery>

1104 The <AuthenticationQuery> element is used to make the query "What assertions containing
1105 authentication statements are available for this subject?" A successful response will be in the form of
1106 assertions containing authentication statements.

1107 The <AuthenticationQuery> element MUST NOT be used as a request for a new authentication
1108 using credentials provided in the request. <AuthenticationQuery> is a request for statements about

1109 authentication acts that have occurred in a previous interaction between the indicated subject and the
1110 Authentication Authority.

1111 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the
1112 addition of the following attribute:

1113 `AuthenticationMethod` [Optional]

1114 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1115 containing authentication statements do you have for this subject with the supplied authentication
1116 method?”

1117 In response to an authentication query, a SAML authority returns assertions with authentication
1118 statements as follows:

- 1119 • Rules given in Section 3.4.4 for matching against the `<Subject>` element of the query identify the
1120 assertions that may be returned.
- 1121 • If the `AuthenticationMethod` attribute is present in the query, at least one
1122 `<AuthenticationStatement>` element in the set of returned assertions **MUST** contain an
1123 `AuthenticationMethod` attribute that matches the `AuthenticationMethod` attribute in
1124 the query. It is **OPTIONAL** for the complete set of all such matching assertions to be returned in
1125 the response.
- 1126 • If any `<RespondWith>` elements are present and none of them contain
1127 “`saml:AuthenticationStatement`”, then the SAML authority returns no assertions with
1128 authentication statements. (See Section 3.2.1.1 for more information.)

1129 The following schema fragment defines the `<AuthenticationQuery>` element and its
1130 **AuthenticationQueryType** complex type:

```
1131 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
1132 <complexType name="AuthenticationQueryType">  
1133   <complexContent>  
1134     <extension base="samlp:SubjectQueryAbstractType">  
1135       <attribute name="AuthenticationMethod" type="anyURI"/>  
1136     </extension>  
1137   </complexContent>  
1138 </complexType>
```

1139 **3.3.4 Element `<AttributeQuery>`**

1140 The `<AttributeQuery>` element is used to make the query “Return the requested attributes for this
1141 subject.” A successful response will be in the form of assertions containing attribute statements. This
1142 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1143 the following element and attribute:

1144 Resource [Optional]

1145 If present, specifies that the attribute query is being made in order to evaluate a specific access
1146 request relating to the resource. The SAML authority MAY use the resource attribute to establish the
1147 scope of the request. It is permitted for this attribute to have the value of the empty URI reference (""),
1148 and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]
1149 §4.2.

1150 If the resource attribute is specified and the SAML authority does not wish to support resource-
1151 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the
1152 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a
1153 second-level <StatusCode> value of ResourceNotRecognized.

1154 <AttributeDesignator> [Any Number] (see Section 2.4.4.1)

1155 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no
1156 attributes are specified, it indicates that all attributes allowed by policy are requested.

1157 In response to an attribute query, a SAML authority returns assertions with attribute statements as
1158 follows:

- 1159 • Rules given in Section 3.4.4 for matching against the <Subject> element of the query identify the
1160 assertions that may be returned.
- 1161 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute
1162 values returned, as noted above.
- 1163 • The SAML authority MAY take the Resource attribute into account in further constraining the values
1164 returned, as noted above.
- 1165 • The attribute values returned MAY be constrained by application-specific policy considerations.
- 1166 • If any <RespondWith> elements are present and none of them contain
1167 "saml:AttributeStatement", then the SAML authority returns no assertions with attribute
1168 statements. (See Section 3.2.1.1 for more information.)

1169
1170 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1171 complex type:

```
1172 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1173 <complexType name="AttributeQueryType">  
1174   <complexContent>  
1175     <extension base="samlp:SubjectQueryAbstractType">  
1176       <sequence>  
1177         <element ref="saml:AttributeDesignator"  
1178           minOccurs="0" maxOccurs="unbounded"/>  
1179       </sequence>  
1180       <attribute name="Resource" type="anyURI" use="optional"/>  
1181     </extension>  
1182   </complexContent>  
1183 </complexType>
```

1184 3.3.5 Element <AuthorizationDecisionQuery>

1185 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on
1186 this resource be allowed for this subject, given this evidence?" A successful response will be in the form
1187 of assertions containing authorization decision statements. This element is of type
1188 **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the
1189 following elements and attribute:

1190 Resource [Required]
 1191 A URI reference indicating the resource for which authorization is requested.
 1192 <Action> [One or More]
 1193 The actions for which authorization is requested.
 1194 <Evidence> [Optional]
 1195 A set of assertions that the SAML authority MAY rely on in making its authorization decision.
 1196 In response to an authorization decision query, a SAML authority returns assertions with authorization
 1197 decision statements as follows:

- 1198 • Rules given in Section 3.4.4 for matching against the <Subject> element of the query identify the
 1199 assertions that may be returned.
- 1200 • If any <RespondWith> elements are present and none of them contain
 1201 "saml:AuthorizationDecisionStatement", then the SAML authority returns no assertions with
 1202 authorization decision statements. (See Section 3.2.1.1 for more information.)

1203 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
 1204 **AuthorizationDecisionQueryType** complex type:

```

1205 <element name="AuthorizationDecisionQuery"
1206 type="samlp:AuthorizationDecisionQueryType"/>
1207 <complexType name="AuthorizationDecisionQueryType">
1208   <complexContent>
1209     <extension base="samlp:SubjectQueryAbstractType">
1210       <sequence>
1211         <element ref="saml:Action" maxOccurs="unbounded"/>
1212         <element ref="saml:Evidence" minOccurs="0"/>
1213       </sequence>
1214       <attribute name="Resource" type="anyURI" use="required"/>
1215     </extension>
1216   </complexContent>
1217 </complexType>
  
```

1218 3.4 Responses

1219 The following sections define the SAML constructs that contain response information.

1220 3.4.1 Complex Type ResponseAbstractType

1221 All SAML responses are of types that are derived from the abstract **ResponseAbstractType** complex
 1222 type. This type defines common attributes and elements that are associated with all SAML responses:

- 1223 ResponseID [Required]
- 1224 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in
- 1225 Section 1.2.3 for identifier uniqueness.
- 1226 InResponseTo [Optional]
- 1227 A reference to the identifier of the request to which the response corresponds, if any. If the response
- 1228 is not generated in response to a request, or if the RequestID attribute value of a request cannot be
- 1229 determined (because the request is malformed), then this attribute MUST NOT be present.
- 1230 Otherwise, it MUST be present and its value MUST match the value of the corresponding
- 1231 RequestID attribute value.
- 1232 MajorVersion [Required]
- 1233 The major version of this response. The identifier for the version of SAML defined in this specification
- 1234 is 1. SAML versioning is discussed in Section 4.
- 1235 MinorVersion [Required]
- 1236 The minor version of this response. The identifier for the version of SAML defined in this specification
- 1237 is 1. SAML versioning is discussed in Section 4.
- 1238 IssueInstant [Required]
- 1239 The time instant of issue of the response. The time value is encoded in UTC as described in Section
- 1240 1.2.2.
- 1241 Recipient [Optional]
- 1242 The intended recipient of this response. This is useful to prevent malicious forwarding of responses to
- 1243 unintended recipients, a protection that is required by some use profiles. It is set by the generator of
- 1244 the response to a URI reference that identifies the intended recipient. If present, the actual recipient
- 1245 MUST check that the URI reference identifies the recipient or a resource managed by the recipient. If
- 1246 it does not, the response MUST be discarded.
- 1247 <ds:Signature> [Optional]
- 1248 An XML Signature that authenticates the response, as described in Section 5.
- 1249 The following schema fragment defines the **ResponseAbstractType** complex type:

```

1250 <complexType name="ResponseAbstractType" abstract="true">
1251   <sequence>
1252     <element ref = "ds:Signature" minOccurs="0"/>
1253   </sequence>
1254   <attribute name="ResponseID" type="ID" use="required"/>
1255   <attribute name="InResponseTo" type="NCName" use="optional"/>
1256   <attribute name="MajorVersion" type="integer" use="required"/>
1257   <attribute name="MinorVersion" type="integer" use="required"/>
1258   <attribute name="IssueInstant" type="dateTime" use="required"/>
1259   <attribute name="Recipient" type="anyURI" use="optional"/>
1260 </complexType>

```

1261 3.4.2 Element <Response>

1262 The <Response> element specifies the status of the corresponding SAML request and a list of zero or

1263 more assertions that answer the request. It has the complex type **ResponseType**, which extends

1264 **ResponseAbstractType** by adding the following elements in order:

- 1265 <Status> [Required]
- 1266 A code representing the status of the corresponding request.
- 1267 <Assertion> [Any Number]
- 1268 Specifies an assertion by value. (See Section 2.3.2 for more information.)
- 1269 The following schema fragment defines the <Response> element and its **ResponseType** complex type:


```

1270 <element name="Response" type="samlp:ResponseType"/>
1271 <complexType name="ResponseType">
1272   <complexContent>
1273     <extension base="samlp:ResponseAbstractType">
1274       <sequence>
1275         <element ref="samlp:Status"/>
1276         <element ref="saml:Assertion" minOccurs="0"
1277 maxOccurs="unbounded"/>
1278       </sequence>
1279     </extension>
1280   </complexContent>
1281 </complexType>

```

1282 3.4.3 Element <Status>

1283 The <Status> element contains the following elements:

1284 <StatusCode> [Required]

1285 A code representing the status of the corresponding request.

1286 <StatusMessage> [Optional]

1287 A message which MAY be returned to an operator.

1288 <StatusDetail> [Optional]

1289 Additional information concerning an error condition.

1290 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1291 <element name="Status" type="samlp:StatusType"/>
1292 <complexType name="StatusType">
1293   <sequence>
1294     <element ref="samlp:StatusCode"/>
1295     <element ref="samlp:StatusMessage" minOccurs="0"/>
1296     <element ref="samlp:StatusDetail" minOccurs="0"/>
1297   </sequence>
1298 </complexType>

```

1299 3.4.3.1 Element <StatusCode>

1300 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the
1301 corresponding request. The <StatusCode> element has the following element and attribute:

1302 Value [Required]

1303 The status code value. This attribute contains an XML Schema QName; a namespace prefix **MUST**
1304 be provided. The value of the topmost <StatusCode> element **MUST** be from the top-level list
1305 provided in this section.

1306 <StatusCode> [Optional]

1307 A subordinate status code that provides more specific information on an error condition.

1308 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The
1309 local parts of these QNames are as follows:

1310 Success

1311 The request succeeded.

1312 VersionMismatch

1313 The SAML responder could not process the request because the version of the request message was

1314 incorrect.

1315 Requester

1316 The request could not be performed due to an error on the part of the requester.

1317 Responder

1318 The request could not be performed due to an error on the part of the SAML responder or SAML

1319 authority.

1320 The following second-level status codes are referenced at various places in the specification. Additional

1321 second-level status codes MAY be defined in future versions of the SAML specification.

1322 RequestVersionTooHigh

1323 The SAML responder cannot process the request because the protocol version specified in the

1324 request message is a major upgrade from the highest protocol version supported by the responder.

1325 RequestVersionTooLow

1326 The SAML responder cannot process the request because the protocol version specified in the

1327 request message is too low.

1328 RequestVersionDeprecated

1329 The SAML responder can not process any requests with the protocol version specified in the request.

1330 TooManyResponses

1331 The response message would contain more elements than the SAML responder will return.

1332 RequestDenied

1333 The SAML responder or SAML authority is able to process the request but has chosen not to

1334 respond. This status code MAY be used when there is concern about the security context of the

1335 request message or the sequence of request messages received from a particular requester.

1336 ResourceNotRecognized

1337 The SAML authority does not wish to support resource-specific attribute queries, or the resource

1338 value provided in the request message is invalid or unrecognized.

1339 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT

1340 define additional codes in the SAML assertion or protocol namespace.

1341 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value

1342 attribute and have the above semantics only in that context.

1343 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex

1344 type:

```

1345 <element name="StatusCode" type="samlp:StatusCodeType"/>
1346 <complexType name="StatusCodeType">
1347   <sequence>
1348     <element ref="samlp:StatusCode" minOccurs="0"/>
1349   </sequence>
1350   <attribute name="Value" type="QName" use="required"/>
1351 </complexType>

```

1352 3.4.3.2 Element <StatusMessage>

1353 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1354 The following schema fragment defines the <StatusMessage> element and its **StatusMessageType**
1355 complex type:

```
1356 <element name="StatusMessage" type="string"/>
```

1357 3.4.3.3 Element <StatusDetail>

1358 The <StatusDetail> element MAY be used to specify additional information concerning an error
1359 condition.

1360 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1361 complex type:

```
1362 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1363 <complexType name="StatusDetailType">  
1364 <sequence>  
1365 <any namespace="##any" processContents="lax" minOccurs="0"  
1366 maxOccurs="unbounded"/>  
1367 </sequence>  
1368 </complexType>
```

1369 3.4.4 Responses to Queries

1370 In response to a query, every assertion returned by a SAML authority MUST contain at least one
1371 statement whose <saml:Subject> element **strongly matches** the <saml:Subject> element found in
1372 the query.

1373 A <saml:Subject> element S1 strongly matches S2 if and only if the following two conditions both
1374 apply:

- 1375 • If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical
1376 <saml:NameIdentifier> element.
- 1377 • If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an identical
1378 <saml:SubjectConfirmation> element.

1379 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
1380 expressed by a query, the <Response> element MUST NOT contain an <Assertion> element and
1381 MUST include a <StatusCode> element with value *Success*. It MAY return a <StatusMessage>
1382 element with additional information.

1383 4 SAML Versioning

1384 The SAML specification set is versioned in two independent ways. Each is discussed in the following
1385 sections, along with processing rules for detecting and handling version differences, when applicable.
1386 Also included are guidelines on when and why specific version information is expected to change in future
1387 revisions of the specification.

1388 When version information is expressed as both a Major and Minor version, it may be expressed
1389 discretely, or in the form *Major.Minor*. The version number *Major_B.Minor_B* is higher than the version
1390 number *Major_A.Minor_A* if and only if:

1391 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$

1392 4.1 SAML Specification Set Version

1393 Each release of the SAML specification set will contain a major and minor version designation describing
1394 its relationship to earlier and later versions of the specification set. The version will be expressed in the
1395 content and filenames of published materials, including the specification set document(s), and XML
1396 schema instance(s). There are no normative processing rules surrounding specification set versioning,
1397 since it merely encompasses the collective release of normative specification documents which
1398 themselves contain processing rules.

1399 The overall size and scope of changes to the specification set document(s) will informally dictate whether
1400 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards
1401 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics
1402 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major
1403 revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0, described
1404 in Section 5.4.7.

1405 4.1.1 Schema Version

1406 As a non-normative documentation mechanism, any XML schema instances published as part of the
1407 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the
1408 specification set version in which it has been published. Validating implementations MAY use the attribute
1409 as a means of distinguishing which version of a schema is being used to validate messages, or to support
1410 a multiplicity of versions of the same logical schema.

1411 4.1.2 SAML Assertion Version

1412 The SAML <Assertion> element contains attributes for expressing the major and minor version of the
1413 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to
1414 document the syntax, semantics, and processing rules of the assertions of the same version. That is,
1415 specification set version 1.0 describes assertion version 1.0, and so on.

1416 There is explicitly NO relationship between the assertion version and the SAML assertion XML
1417 namespace that contains the schema definitions for that assertion version.

1418 The following processing rules apply:

- 1419 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported by
1420 the authority.
- 1421 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
1422 supported by the relying party.
- 1423 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
1424 number is higher than the minor assertion version number supported by the relying party. However,
1425 all assertions that share a major assertion version number MUST share the same general processing

1426 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1
1427 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0
1428 assertion without ill effect.

1429 **4.1.3 SAML Protocol Version**

1430 The SAML protocol `<Request>` and `<Response>` elements contain attributes for expressing the major
1431 and minor version of the request or response message using a pair of integers. Each version of the SAML
1432 specification set will be construed so as to document the syntax, semantics, and processing rules of the
1433 protocol messages of the same version. That is, specification set version 1.0 describes request and
1434 response version V1.0, and so on.

1435 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace
1436 that contains the schema definitions for protocol messages for that protocol version.

1437 The version numbers used in SAML protocol `<Request>` and `<Response>` elements will be the same
1438 for any particular revision of the SAML specification set.

1439 **4.1.3.1 Request Version**

1440 The following processing rules apply to requests:

- 1441 • A SAML requester SHOULD issue requests with the highest request version supported by both the
1442 SAML requester and the SAML responder.
- 1443 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume
1444 that it supports requests with the highest request version supported by the requester.
- 1445 • A SAML requester MUST NOT issue a request message with a request version number matching a
1446 response version number that the requester does not support.
- 1447 • A SAML responder MUST reject any request with a major request version number not supported by
1448 the responder.
- 1449 • A SAML responder MAY process or MAY reject any request whose minor request version number is
1450 higher than the highest supported request version that it supports. However, all requests that share a
1451 major request version number MUST share the same general processing rules and semantics, and
1452 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax of
1453 a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

1454 **4.1.4 Response Version**

1455 The following processing rules apply to responses:

- 1456 • A SAML responder MUST NOT issue a response message with a response version number higher
1457 than the request version number of the corresponding request message.
- 1458 • A SAML responder MUST NOT issue a response message with a major response version number
1459 lower than the major request version number of the corresponding request message except to report
1460 the error `RequestVersionTooHigh`.

1461 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-
1462 level `<StatusCode>` value of `VersionMismatch`, and MAY result in reporting one of the following
1463 second-level values: `RequestVersionTooHigh`, `RequestVersionTooLow`, or
1464 `RequestVersionDeprecated`.

1465 **4.1.5 Permissible Version Combinations**

1466 In general, assertions of a particular major version may appear in response messages of the same major
1467 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol
1468 schema. Future versions of this specification are expected to explicitly describe the permitted
1469 combinations across major versions.

1470 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to
1471 appear in a V1.1 response message.

1472 **4.2 SAML Namespace Version**

1473 XML schema instances and "qualified names" (QNames) published as part of the specification set contain
1474 one or more target namespaces into which the type, element, and attribute definitions are placed. Each
1475 namespace is distinct from the others, and represents, in shorthand, the structural and syntactical
1476 definitions that make up that part of the specification.

1477 The namespace URIs defined by the specification set will generally contain version information of the
1478 form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to
1479 the major and minor version of the specification set in which the namespace is first introduced and
1480 defined. This information is not typically consumed by an XML processor, which treats the namespace
1481 opaquely, but is intended to communicate the relationship between the specification set and the
1482 namespaces it defines.

1483 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
1484 defined by a major revision of the specification set to remain valid and stable across minor revisions of
1485 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,
1486 but this is expected to be rare. In such cases, the older namespaces and their associated definitions
1487 should be expected to remain valid until a major specification set revision.

1488 **4.2.1 Schema Evolution**

1489 In general, maintaining namespace stability while adding or changing the content of a schema are
1490 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
1491 older implementations will react to any given change, making forward compatibility difficult to achieve.
1492 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
1493 stability. Except in special circumstances (for example to correct major deficiencies or fix errors),
1494 implementations should expect forward compatible schema changes in minor revisions, allowing new
1495 messages to validate against older schemas.

1496 Implementations SHOULD expect and be prepared to deal with new extensions and message types in
1497 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types
1498 that leverage the extension facilities described in Section 6. Older implementations SHOULD reject such
1499 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
1500 include new query, statement, or condition types.

5 SAML and XML Signature Syntax and Processing

1501

1502 SAML assertions and SAML protocol request and response messages may be signed, with the following
1503 benefits:

- 1504 • An assertion signed by the SAML authority supports:
 - 1505 – Assertion integrity.
 - 1506 – Authentication of the SAML authority to a SAML relying party.
 - 1507 – If the signature is based on the SAML authority's public-private key pair, then it also provides for
1508 non-repudiation of origin.
- 1509 • A SAML protocol request or response message signed by the message originator supports:
 - 1510 – Message integrity.
 - 1511 – Authentication of message origin to a destination.
 - 1512 – If the signature is based on the originator's public-private key pair, then it also provides for non-
1513 repudiation of origin.

1514 A digital signature is not always required in SAML. For example, it may not be required in the following
1515 situations:

- 1516 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains
1517 protection from a signature on the containing protocol response message. "Inherited" signatures
1518 should be used with care when the contained object (such as the assertion) is intended to have a
1519 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,
1520 exposing the XML content and adding potentially unnecessary overhead.
- 1521 • The SAML relying party or SAML requester may have obtained an assertion or protocol message
1522 from the SAML authority or SAML responder directly (with no intermediaries) through a secure
1523 channel, with the SAML authority or SAML responder having authenticated to the relying party or
1524 SAML responder by some means other than a digital signature.

1525 Many different techniques are available for "direct" authentication and secure channel establishment
1526 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1527 the applicable security requirements depend on the communicating applications and the nature of the
1528 assertion or message transported.

1529 It is recommended that, in all other contexts, digital signatures be used for assertions and request and
1530 response messages. Specifically:

- 1531 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
1532 SHOULD be signed by the SAML authority.
- 1533 • A SAML protocol message arriving at a destination from an entity other than the originating site
1534 SHOULD be signed by the origin site.

1535 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that
1536 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures
1537 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure
1538 compatibility with profiles that may require other mechanisms.

1539 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST
1540 be used if signing.

5.1 Signing Assertions

1541

1542 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema
1543 as described in Section 2.3.

1544 5.2 Request/Response Signing

1545 All SAML protocol request and response messages MAY be signed using the XML Signature. This is
1546 reflected in the schema as described in Sections 3.2 and 3.4.

1547 5.3 Signature Inheritance

1548 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
1549 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a
1550 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
1551 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,
1552 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
1553 interpretation should be equivalent to the case where the assertion itself was signed with the same key
1554 and signature options.

1555 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
1556 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may
1557 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
1558 information from the surrounding context, but no such inheritance should be inferred unless specifically
1559 identified by the profile.

1560 5.4 XML Signature Profile

1561 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1562 and many choices. This section details the constraints on these facilities so that SAML processors do not
1563 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
1564 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply: the
1565 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the
1566 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section
1567 as the identifier attributes.

1568 5.4.1 Signing Formats and Algorithms

1569 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1570 detached.

1571 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
1572 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
1573 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

1574 5.4.2 References

1575 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
1576 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element
1577 may or may not be the root element of the actual XML document containing the signed assertion or
1578 message.

1579 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1580 value of the root element of the message being signed. For example, if the attribute value is "foo", then
1581 the `URI` attribute in the `<ds:Reference>` element MUST be "#foo".

1582 5.4.3 Canonicalization Method

1583 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
1584 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
1585 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
1586 SAML messages embedded in an XML context can be verified independent of that context.

1587 5.4.4 Transforms

1588 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
1589 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
1590 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1591 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1592 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1593 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This
1594 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
1595 applying the transforms manually to the content and reverifying the result as consisting of the same
1596 SAML message.

1597 5.4.5 KeyInfo

1598 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the
1599 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
1600 be absent.

1601 5.4.6 Binding Between Statements in a Multi-Statement Assertion

1602 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section
1603 2.

1604 5.4.7 Interoperability with SAML V1.0

1605 The use of XML Signature [XMLSig] described above is incompatible with the usage described in the
1606 SAML V1.0 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to
1607 ensure interoperability. It was constrained by the inability to use URI references to identify the SAML
1608 content to be signed. With this limitation removed by the addition of SAML identifier attributes, a decision
1609 has been made to forgo backwards compatibility with the older specification in this respect.

1610 5.4.8 Example

1611 Following is an example of a signed response containing a signed assertion. Line breaks have been
1612 added for readability; the signatures are not valid and cannot be successfully verified.

```
1613 <Response  
1614   IssueInstant="2003-04-17T00:46:02Z"  
1615   MajorVersion="1"  
1616   MinorVersion="1"  
1617   Recipient="www.opensaml.org"  
1618   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
1619   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
1620   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
1621   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1622   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
1623   <ds:Signature  
1624     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
1625     <ds:SignedInfo>  
1626     <ds:CanonicalizationMethod  
1627       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1628     <ds:SignatureMethod  
1629       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1630     <ds:Reference  
1631       URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
1632     <ds:Transforms>  
1633     <ds:Transform  
1634       Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />  
1635     <ds:Transform
```

```

1636     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1637 <InclusiveNamespaces
1638     PrefixList="#default saml samlp ds xsd xsi"
1639     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1640 </ds:Transform>
1641 </ds:Transforms>
1642 <ds:DigestMethod
1643     Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
1644 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
1645 </ds:Reference>
1646 </ds:SignedInfo>
1647 <ds:SignatureValue>
1648 x/GyPbzmFEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
1649 EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
1650 w6vKhaqledl0BYyrIzb4KkH04ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
1651 <ds:KeyInfo>
1652 <ds:X509Data>
1653 <ds:X509Certificate>
1654 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1655 MRIwEAYDVQQIEwIeLXAuXNjb25zaW4xZDA0BGNVBAcTB01hZG1zb24xIDAeBgNVBAoT
1656 FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQLEyJEAuXZpc2lvbiBvZiBJ
1657 bmZvcmlhdG1vbiBUZWNobm9sb2d5MSUwIiwYDVQDExxIRVBLSSBTZXXJ2ZXIqQ0Eg
1658 LS0gMjAwMjA3MDFBMB4XDTAyMDEyMjA3MjMjcmVvXDA2MDEyMDEyMDEyMDEyMDEy
1659 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbWJESMBAGA1UEBxMjQwNTU1IEFy
1660 Ym9yMQ4wDAYDVQQKEwVWVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJ1ZXQyLmVh
1661 dTEuMjAwMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEyMDEy
1662 CSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVT8vuRay+x50z7GJj
1663 IHRYQgIv6IqaGG04eTcyVMhoeE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrcP+
1664 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
1665 pmqOI fGTWQIDAQABox0wGzAMBGNVHRMBAf8EAjaAMAsGA1UdDwQEAwIFoDANBgkq
1666 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2N
1667 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
1668 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1yLGPdiowMNTREg8cCx3w/w==
1669 </ds:X509Certificate>
1670 </ds:X509Data>
1671 </ds:KeyInfo>
1672 </ds:Signature>
1673 <Status><StatusCode Value="samlp:Success" /></Status>
1674 <Assertion
1675     AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
1676     IssueInstant="2003-04-17T00:46:02Z"
1677     Issuer="www.opensaml.org"
1678     MajorVersion="1"
1679     MinorVersion="1"
1680     xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
1681     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1682     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
1683 <Conditions
1684     NotBefore="2003-04-17T00:46:02Z"
1685     NotOnOrAfter="2003-04-17T00:51:02Z">
1686 <AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
1687 </AudienceRestrictionCondition></Conditions>
1688 <AuthenticationStatement
1689     AuthenticationInstant="2003-04-17T00:46:00Z"
1690     AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
1691 <Subject>
1692 <NameIdentifier
1693     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
1694 scott@example.org</NameIdentifier>
1695 <SubjectConfirmation>
1696 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
1697 </SubjectConfirmation></Subject>
1698 <SubjectLocality

```

```

1699     IPAddress="127.0.0.1"/>
1700 </AuthenticationStatement>
1701 <ds:Signature
1702     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1703 <ds:SignedInfo>
1704 <ds:CanonicalizationMethod
1705     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1706 <ds:SignatureMethod
1707     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1708 <ds:Reference
1709     URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
1710 <ds:Transforms>
1711 <ds:Transform
1712     Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
1713 <ds:Transform
1714     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1715 <InclusiveNamespaces
1716     PrefixList="#default saml samlp ds xsd xsi"
1717     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1718 </ds:Transform>
1719 </ds:Transforms>
1720 <ds:DigestMethod
1721     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1722 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
1723 </ds:Reference>
1724 </ds:SignedInfo>
1725 <ds:SignatureValue>
1726 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
1727 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
1728 MWuL/cBUj2OtBZOQMFN7jQ9YB7k1Iz3RqVL+wNmeWI4=</ds:SignatureValue>
1729 <ds:KeyInfo>
1730 <ds:X509Data>
1731 <ds:X509Certificate>
1732 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1733 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
1734 F1VuaXZlcnNpdHkkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
1735 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
1736 Ls0gMjAwMjA3MDFBMB4XDTEyMDcyNjA3Mjc1MVoXDTE2MDkwNDA3Mjc1MVoYYSx
1737 CzAJBgNVBAYTA1VTMREwDwYDVQQQIEwhNaWNoaWdhbWJESMBAGAlUEBxMJQW5uIEFy
1738 Ym9yMQ4wDAYDVQQKEwVWQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJ2ZXQyLmVh
1739 dTEEnMCUGCSqGSIB3DQEJARYYcm9vdEBzaG1iMS5pbmRlcm5ldIuZWR1MIGfMAOG
1740 CSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+xs0z7GJj
1741 IHRYQgIv6IqaGG04eTcyVMhoekE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrCP+
1742 c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
1743 pmqOIfGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
1744 hkiG9w0BAQQFAA0BgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
1745 qgi7lFV6MDkHmTvTqTbjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
1746 8I3bsbmRAUg4UP9hH6ABVq4KQKMknulxQxLhpRlyLGPdiowMNTreEG8cCx3w/w==
1747 </ds:X509Certificate>
1748 </ds:X509Data>
1749 </ds:KeyInfo>
1750 </ds:Signature></Assertion></Response>

```

1751

6 SAML Extensions

1752 The SAML schemas support extensibility. An example of an application that extends SAML assertions is
1753 the Liberty Protocols and Schema Specification [**LibertyProt**]. The following sections explain how to use
1754 the extensibility features in SAML to create extension schemas.

1755 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML elements
1756 MAY serve as the head element of a substitution group. Also, types are not defined as *final*, so that all
1757 SAML types MAY be extended and restricted. The following sections discuss only elements that have
1758 been specifically designed to support extensibility.

6.1 Assertion Schema Extension

1760 The SAML assertion schema is designed to permit separate processing of the assertion package and the
1761 statements it contains, if the extension mechanism is used for either part.

1762 The following elements are intended specifically for use as extension points in an extension schema; their
1763 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 1764 • <Condition>
- 1765 • <Statement>
- 1766 • <SubjectStatement>

1767 The following elements that are directly usable as part of SAML MAY be extended:

- 1768 • <AuthenticationStatement>
- 1769 • <AuthorizationDecisionStatement>
- 1770 • <AttributeStatement>
- 1771 • <AudienceRestrictionCondition>

1772 The following elements are defined to allow elements from arbitrary namespaces within them, which
1773 serves as a built-in extension point without requiring an extension schema:

- 1774 • <AttributeValue>
- 1775 • <Advice>

6.2 Protocol Schema Extension

1777 The following SAML protocol elements are intended specifically for use as extension points in an
1778 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
1779 type:

- 1780 • <Query>
- 1781 • <SubjectQuery>

1782 The following elements that are directly usable as part of SAML MAY be extended:

- 1783 • <Request>
- 1784 • <AuthenticationQuery>
- 1785 • <AuthorizationDecisionQuery>
- 1786 • <AttributeQuery>
- 1787 • <Response>

1788 6.3 Use of Type Derivation and Substitution Groups

1789 W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of an
1790 extended type: type derivation and substitution groups.

1791 For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of the
1792 `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived
1793 from **StatementType**. The following example of a SAML assertion assumes that the extension schema
1794 (represented by the `new:` prefix) has defined this new type:

```
1795 <saml:Assertion ...>  
1796   <saml:Statement xsi:type="new:NewStatementType">  
1797     ...  
1798   </saml:Statement>  
1799 </saml:Assertion>
```

1800 Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a
1801 substitution group that has `<Statement>` as a head element. For the substituted element to be schema-
1802 valid, it needs to have a type that matches or is derived from the head element's type. The following is an
1803 example of an extension schema fragment that defines this new element:

```
1804 <xsd:element "NewStatement" type="new:NewStatementType"  
1805   substitutionGroup="saml:Statement"/>
```

1806 The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML
1807 `<Statement>` element can be used. The following is an example of a SAML assertion that uses the
1808 extension element:

```
1809 <saml:Assertion ...>  
1810   <new:NewStatement>  
1811     ...  
1812   </new:NewStatement>  
1813 </saml:Assertion>
```

1814 The choice of extension method has no effect on the semantics of the XML document but does have
1815 implications for interoperability.

1816 The advantages of type derivation are as follows:

- 1817 • A document can be more fully interpreted by a parser that does not have access to the extension
1818 schema because a “native” SAML element is available.
- 1819 • At the time of this writing, some W3C XML Schema validators do not support substitution groups,
1820 whereas the `xsi:type` attribute is widely supported.

1821 The advantage of substitution groups is that a document can be explained without the need to explain the
1822 functioning of the `xsi:type` attribute.

1823 7 SAML-Defined Identifiers

1824 The following sections define URI-based identifiers for common authentication methods, resource access
1825 actions, and subject name identifier formats.

1826 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of
1827 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
1828 have one of the following stems:

```
1829 urn:oasis:names:tc:SAML:1.0:  
1830 urn:oasis:names:tc:SAML:1.1:
```

1831 7.1 Authentication Method Identifiers

1832 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the
1833 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although
1834 both can refer to the same underlying mechanisms. An authentication statement with an
1835 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The
1836 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the
1837 authentication statement does not provide the means to perform that authentication, such as a password,
1838 key, or certificate.

1839 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,
1840 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML
1841 relying party to confirm that the request or message came from a system entity that corresponds to the
1842 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method
1843 that the relying party can use to do this in the future. This may or may not have any relationship to an
1844 authentication that was performed previously. Unlike the authentication method, the subject confirmation
1845 method may be accompanied by some piece of information, such as a certificate or key, that will allow the
1846 relying party to perform the necessary check.

1847 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML
1848 bindings and profiles specification [**SAMLBind**] for more information. Additional methods may be added
1849 by defining new profiles or by private agreement.

1850 The following identifiers refer to SAML-specified authentication methods.

1851 7.1.1 Password

1852 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

1853 The authentication was performed by means of a password.

1854 7.1.2 Kerberos

1855 **URI:** urn:ietf:rfc:1510

1856 The authentication was performed by means of the Kerberos protocol [**RFC 1510**], an instantiation of the
1857 Needham-Schroeder symmetric key authentication mechanism [**Needham78**].

1858 7.1.3 Secure Remote Password (SRP)

1859 **URI:** urn:ietf:rfc:2945

1860 The authentication was performed by means of Secure Remote Password protocol as specified in [**RFC**
1861 **2945**].

1862 **7.1.4 Hardware Token**

1863 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

1864 The authentication was performed using some (unspecified) hardware token.

1865 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

1866 **URI:** urn:ietf:rfc:2246

1867 The authentication was performed using either the SSL or TLS protocol with certificate-based client
1868 authentication. TLS is described in **[RFC 2246]**.

1869 **7.1.6 X.509 Public Key**

1870 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

1871 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1872 an X.509 PKI **[X.500][PKIX]**. It may have been one of the mechanisms for which a more specific identifier
1873 has been defined below.

1874 **7.1.7 PGP Public Key**

1875 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

1876 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1877 a PGP web of trust **[PGP]**. It may have been one of the mechanisms for which a more specific identifier
1878 has been defined below.

1879 **7.1.8 SPKI Public Key**

1880 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

1881 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1882 a SPKI PKI **[SPKI]**. It may have been one of the mechanisms for which a more specific identifier has
1883 been defined below.

1884 **7.1.9 XKMS Public Key**

1885 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

1886 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1887 a XKMS trust service **[XKMS]**. It may have been one of the mechanisms for which a more specific
1888 identifier has been defined below.

1889 **7.1.10 XML Digital Signature**

1890 **URI:** urn:ietf:rfc:3075

1891 The authentication was performed by means of an XML digital signature **[RFC 3075]**.

1892 **7.1.11 Unspecified**

1893 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

1894 The authentication was performed by an unspecified means.

1895 **7.2 Action Namespace Identifiers**

1896 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see Section
1897 2.4.5.1) to refer to common sets of actions to perform on resources.

1898 **7.2.1 Read/Write/Execute/Delete/Control**

1899 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

1900 Defined actions:

1901 Read Write Execute Delete Control

1902 These actions are interpreted as follows:

1903 Read

1904 The subject may read the resource.

1905 Write

1906 The subject may modify the resource.

1907 Execute

1908 The subject may execute the resource.

1909 Delete

1910 The subject may delete the resource.

1911 Control

1912 The subject may specify the access control policy for the resource.

1913 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

1914 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

1915 Defined actions:

1916 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1917 The actions specified in Section 7.2.1 are interpreted in the same manner described there. Actions
1918 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
1919 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1920 affirmatively denied read permission.

1921 A SAML authority MUST NOT authorize both an action and its negated form.

1922 **7.2.3 Get/Head/Put/Post**

1923 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

1924 Defined actions:

1925 GET HEAD PUT POST

1926 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
1927 the GET action on a resource is authorized to retrieve it.

1928 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
1929 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
1930 operation may cause data to be modified and a POST operation may cause modification to a resource
1931 other than the one specified in the request. For this reason a separate Action URI reference specifier is
1932 provided.

1933 **7.2.4 UNIX File Permissions**

1934 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1935 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

1936 The action string is a four-digit numeric code:

1937 *extended user group world*

1938 Where the *extended* access permission has the value

- 1939 +2 if sgid is set
- 1940 +4 if suid is set
- 1941 The *user group* and *world* access permissions have the value
- 1942 +1 if execute permission is granted
- 1943 +2 if write permission is granted
- 1944 +4 if read permission is granted
- 1945 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read
- 1946 and execute; and world read.

1947 **7.3 NameIdentifier Format Identifiers**

- 1948 The following identifiers MAY be used in the Format attribute of the <NameIdentifier> element (see
- 1949 Section 2.4.2.2) to refer to common formats for the content of the <NameIdentifier> element. The
- 1950 recommended identifiers shown below SHOULD be used in preference to the deprecated identifiers,
- 1951 which are planned to be removed in the next major version of the SAML assertion specification.

1952 **7.3.1 Unspecified**

- 1953 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
- 1954 The interpretation of the content of the <NameQualifier> element is left to individual implementations.

1955 **7.3.2 Email Address**

- 1956 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
- 1957 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#emailAddress
- 1958 Indicates that the content of the <NameIdentifier> element is in the form of an email address,
- 1959 specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form
- 1960 local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no
- 1961 comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

1962 **7.3.3 X.509 Subject Name**

- 1963 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- 1964 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName
- 1965 Indicates that the content of the <NameIdentifier> element is in the form specified for the contents of
- 1966 the <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors
- 1967 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
- 1968 differ from the rules given in IETF RFC 2253 [RFC 2253].

1969 **7.3.4 Windows Domain Qualified Name**

- 1970 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName
- 1971 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName
- 1972 Indicates that the content of the <NameIdentifier> element is a Windows domain qualified name. A
- 1973 Windows domain qualified user name is a string of the form "DomainName\UserName". The domain
- 1974 name and "\" separator MAY be omitted.

8 References

- 1975
- 1976 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 1977
- 1978 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty Alliance Project, January 2003, http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.
- 1979
- 1980
- 1981
- 1982 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December 1978.
- 1983
- 1984
- 1985 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*. IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 1986
- 1987 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF RFC 2459, January 1999. <http://www.ietf.org/rfc/rfc2459.txt>.
- 1988
- 1989
- 1990 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 1991
- 1992 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 1993
- 1994 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. <http://www.ietf.org/rfc/rfc2246.txt>.
- 1995
- 1996 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. <http://www.ietf.org/rfc/rfc2253.txt>.
- 1997
- 1998
- 1999 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.
- 2000
- 2001 **[RFC 2630]** R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999. <http://www.ietf.org/rfc/rfc2630.txt>.
- 2002
- 2003 **[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. <http://www.ietf.org/rfc/rfc2822.txt>.
- 2004
- 2005 **[RFC 2945]** T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945, September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.
- 2006
- 2007 **[RFC 3075]** D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.
- 2008
- 2009 **[SAMLBind]** E. Maler et al. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-1.1. <http://www.oasis-open.org/committees/security/>.
- 2010
- 2011
- 2012 **[SAMLConform]** E. Maler et al. *Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-conform-1.1. <http://www.oasis-open.org/committees/security/>.
- 2013
- 2014
- 2015 **[SAMLCore1.0]** E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. OASIS, November 2002. <http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf>.
- 2016
- 2017
- 2018 **[SAMLGloss]** E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1. <http://www.oasis-open.org/committees/security/>.
- 2019
- 2020
- 2021 **[SAMLXSD]** E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID oasis-sstc-saml-schema-protocol-1.1. <http://www.oasis-open.org/committees/security/>.
- 2022
- 2023

2024	[SAMLSecure]	E. Maler et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, September 2003. Document ID oasis-sstc-saml-sec-consider-1.1. http://www.oasis-open.org/committees/security/ .
2025		
2026		
2027		
2028	[SAML-XSD]	E. Maler et al. <i>SAML assertion schema</i> . OASIS, September 2003. Document ID oasis-sstc-saml-schema-assertion-1.1. http://www.oasis-open.org/committees/security/ .
2029		
2030		
2031	[Schema1]	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/ .
2032		
2033	[Schema2]	P. V. Biron et al. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-2/ .
2034		
2035	[SPKI]	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999. http://www.ietf.org/rfc/rfc2693.txt .
2036		
2037		
2038	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
2039		
2040	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. http://www.w3.org/TR/WD-charreq .
2041		
2042	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. http://www.w3.org/TR/charmod/ .
2043		
2044	[X.500]	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
2045		
2046	[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/ .
2047		
2048		
2049	[XML]	T. Bray, et al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> . World Wide Web Consortium, October 2000. http://www.w3.org/TR/REC-xml .
2050		
2051	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , World Wide Web Consortium, February 2002. http://www.w3.org/TR/xmlsig-core/ .
2052		
2053	[XMLSig-XSD]	XML Signature Schema. World Wide Web Consortium. http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd .
2054		
2055		

2056 **Appendix A. Acknowledgments**

2057 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2058 Committee, whose voting members at the time of publication were:

- 2059 • Frank Siebenlist, Argonne National Laboratory
- 2060 • Irving Reid, Baltimore Technologies
- 2061 • Hal Lockhart, BEA Systems
- 2062 • Steven Lewis, Booz Allen Hamilton
- 2063 • John Hughes, Entegriy Solutions
- 2064 • Carlisle Adams, Entrust
- 2065 • Jason Rouault, Hewlett-Packard
- 2066 • Maryann Hondo, IBM
- 2067 • Anthony Nadalin, IBM
- 2068 • Scott Cantor, individual
- 2069 • RL "Bob" Morgan, individual
- 2070 • Trevor Perrin, individual
- 2071 • Pdraig Moloney, NASA
- 2072 • Prateek Mishra, Netegrity (co-chair)
- 2073 • Frederick Hirsch, Nokia
- 2074 • Senthil Sengodan, Nokia
- 2075 • Timo Skytta, Nokia
- 2076 • Charles Knouse, Oblix
- 2077 • Steve Anderson, OpenNetwork
- 2078 • Simon Godik, Overxeer
- 2079 • Rob Philpott, RSA Security (co-chair)
- 2080 • Dipak Chopra, SAP
- 2081 • Jahan Moreh, Sigaba
- 2082 • Bhavna Bhatnagar, Sun Microsystems
- 2083 • Jeff Hodges, Sun Microsystems
- 2084 • Eve Maler, Sun Microsystems (coordinating editor)
- 2085 • Emily Xu, Sun Microsystems
- 2086 • Phillip Hallam-Baker, VeriSign

2087

Appendix B. Notices

2088 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2089 might be claimed to pertain to the implementation or use of the technology described in this document or
2090 the extent to which any license under such rights might or might not be available; neither does it
2091 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
2092 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
2093 made available for publication and any assurances of licenses to be made available, or the result of an
2094 attempt made to obtain a general license or permission for the use of such proprietary rights by
2095 implementors or users of this specification, can be obtained from the OASIS Executive Director.

2096 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
2097 or other proprietary rights which may cover technology that may be required to implement this
2098 specification. Please address the information to the OASIS Executive Director.

2099 **Copyright © OASIS Open 2003. All Rights Reserved.**

2100 This document and translations of it may be copied and furnished to others, and derivative works that
2101 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
2102 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
2103 and this paragraph are included on all such copies and derivative works. However, this document itself
2104 may not be modified in any way, such as by removing the copyright notice or references to OASIS,
2105 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
2106 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to
2107 translate it into languages other than English.

2108 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2109 or assigns.

2110 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2111 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2112 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2113 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.