



1

2

3

4

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

5

Document identifier: draft-sstc-core-[3130](#)

6

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

7

Publication date: ~~April 4rd 2002~~[March 29th 2002](#)

8

Maturity Level: Committee Working Draft

9

Send comments to: security-requestors-comment@lists.oasis-open.org

10

Note: Before sending a message to this list you must first subscribe; send an email message to security-

11

requestors-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

12

Editors:

13

Phillip Hallam-Baker, VeriSign,

14

Eve Maler, Sun Microsystems

15

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML)	1
1. INTRODUCTION	10
1.1. NOTATION	10
1.2. SCHEMA ORGANIZATION AND NAMESPACES	10
1.2.1. <i>String and URI Values</i>	11
1.2.2. <i>Time Values.</i>	11
1.2.3. <i>Comparing SAML values</i>	11
1.3. SAML CONCEPTS (NON-NORMATIVE)	11
1.3.1. <i>Overview</i>	12
1.3.2. <i>SAML and URI-Based Identifiers</i>	13
1.3.3. <i>SAML and Extensibility</i>	13
2. SAML ASSERTIONS	14
2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	14
2.2. SIMPLE TYPES	14
2.2.1. <i>Simple Types IDType and IDReferenceType</i>	14
2.2.2. <i>Simple Type DecisionType</i>	15
2.3. ASSERTIONS	15
2.3.1. <i>Element <AssertionID></i>	15
2.3.2. <i>Element <Assertion></i>	15
2.3.2.1. <i>Element <Conditions></i>	17
2.3.2.1.1 <i>Attributes NotBefore and NotOnOrAfter</i>	18
2.3.2.1.2 <i>Element <Condition></i>	18
2.3.2.1.3 <i>Elements <AudienceRestrictionCondition> and <Audience></i>	18
2.3.2.2. <i>Elements <Advice> and <AdviceElement></i>	19
2.4. STATEMENTS	19
2.4.1. <i>Element <Statement></i>	19

43	2.4.2. Element <SubjectStatement>	19
44	2.4.2.1. Element <Subject>	20
45	2.4.2.2. Element <NameIdentifier>	20
46	2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>	21
47	2.4.3. Element <AuthenticationStatement>	22
48	2.4.3.1. Element <SubjectLocality>	22
49	2.4.3.2. Element <AuthorityBinding>	23
50	2.4.4. Element <AuthorizationDecisionStatement>	23
51	2.4.4.1. Element <Action>	25
52	2.4.4.2. Element <Evidence>	25
53	2.4.5. Element <AttributeStatement>	25
54	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	26
55	2.4.5.1.1 Element <AttributeValue>	26
56	3. SAML PROTOCOL	28
57	3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	28
58	3.2. REQUESTS	28
59	3.2.1. Complex Type RequestAbstractType	28
60	3.2.1.1. Element <RespondWith>	29
61	3.2.2. Element <Request>	29
62	3.2.3. Element <AssertionArtifact>	30
63	3.3. QUERIES	30
64	3.3.1. Element <Query>	30
65	3.3.2. Element <SubjectQuery>	31
66	3.3.3. Element <AuthenticationQuery>	31
67	3.3.4. Element <AttributeQuery>	32
68	3.3.5. Element <AuthorizationDecisionQuery>	32
69	3.4. RESPONSES	33
70	3.4.1. Complex Type ResponseAbstractType	33
71	3.4.2. Element <Response>	34
72	3.4.3. Element <Status>	34

73	3.4.3.1. Element <StatusCode>	35
74	3.4.3.2. Element <StatusMessage>	36
75	3.4.3.3. Element <StatusDetail>	36
76	3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>	36
77	4. SAML VERSIONING	38
78	4.1. ASSERTION VERSION	38
79	4.2. REQUEST VERSION	38
80	4.3. RESPONSE VERSION	39
81	5. SAML & XML-SIGNATURE SYNTAX AND PROCESSING	40
82	5.1. SIGNING ASSERTIONS	40
83	5.2. REQUEST /RESPONSE SIGNING	41
84	5.3. SIGNATURE INHERITANCE	41
85	5.3.1. <i>Rationale</i>	41
86	5.3.2. <i>Rules for SAML Signature Inheritance</i>	41
87	5.4. XML SIGNATURE PROFILE	41
88	5.4.1. <i>Signing formats</i>	41
89	5.4.2. <i>CanonicalizationMethod</i>	41
90	5.4.3. <i>Transforms</i>	42
91	5.4.4. <i>KeyInfo</i>	42
92	5.4.5. <i>Binding between statements in a multi-statement assertion</i>	42
93	6. SAML EXTENSIONS	43
94	6.1. ASSERTION SCHEMA EXTENSION	43
95	6.2. PROTOCOL SCHEMA EXTENSION	43
96	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	44
97	7. SAML-DEFINED IDENTIFIERS	45

98	7.1. AUTHENTICATION METHOD IDENTIFIERS	45
99	7.1.1. Password :	46
100	7.1.2. Kerberos	46
101	7.1.3. Secure Remote Password (SRP)	47
102	7.1.4. SSL/TLS Certificate Based Client Authentication:	47
103	7.1.5. X.509 Public Key	47
104	7.1.6. PGP Public Key	47
105	7.1.7. SPKI Public Key	47
106	7.1.8. XKMS Public Key	47
107	7.1.9. XML Digital Signature	48
108	7.2. ACTION NAMESPACE IDENTIFIERS	48
109	7.2.1. Read/Write/Execute/Delete/Control:	48
110	7.2.2. Read/Write/Execute/Delete/Control with Negation:	49
111	7.2.3. Get/Head/Put/Post:	49
112	7.2.4. UNIX File Permissions:	49
113	8. SAML SCHEMA LISTINGS	51
114	8.1. ASSERTION SCHEMA	51
115	8.2. PROTOCOL SCHEMA	54
116	9. REFERENCES	57
117	10. ACKNOWLEDGEMENTS	59
118	APPENDIX A. NOTICES	61
119	ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP	
120	LANGUAGE (SAML)	1
121	1. INTRODUCTION	9
122	1.1. NOTATION	9

123	1.2. SCHEMA ORGANIZATION AND NAMESPACES	9
124	1.2.1. String and URI Values	10
125	1.2.2. Time Values.	10
126	1.2.3. Comparing SAML values	10
127	1.3. SAML CONCEPTS (NON-NORMATIVE)	10
128	1.3.1. Overview	11
129	1.3.2. SAML and URI Based Identifiers	12
130	1.3.3. SAML and Extensibility	12
131	2. SAML ASSERTIONS	13
132	2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	13
133	2.2. SIMPLE TYPES	13
134	2.2.1. Simple Types IDType and IDReferenceType	13
135	2.2.2. Simple Type DecisionType	14
136	2.3. ASSERTIONS	14
137	2.3.1. Element <AssertionID>	14
138	2.3.2. Element <Assertion>	14
139	2.3.2.1. Element <Conditions>	16
140	2.3.2.1.1. Attributes NotBefore and NotOnOrAfter	17
141	2.3.2.1.2. Element <Condition>	17
142	2.3.2.1.3. Elements <AudienceRestrictionCondition> and <Audience>	17
143	2.3.2.2. Elements <Advice> and <AdviceElement>	18
144	2.4. STATEMENTS	18
145	2.4.1. Element <Statement>	18
146	2.4.2. Element <SubjectStatement>	18
147	2.4.2.1. Element <Subject>	19
148	2.4.2.2. Element <NameIdentifier>	19
149	2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>	20
150	2.4.3. Element <AuthenticationStatement>	21
151	2.4.3.1. Element <SubjectLocality>	21
152	2.4.3.2. Element <AuthorityBinding>	22

153	2.4.4. Element <AuthorizationDecisionStatement>	22
154	2.4.4.1. Element <Action>	23
155	2.4.4.2. Element <Evidence>	24
156	2.4.5. Element <AttributeStatement>	24
157	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	25
158	2.4.5.1.1. Element <AttributeValue>	25
159	3. SAML PROTOCOL	26
160	3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	26
161	3.2. REQUESTS	26
162	3.2.1. Complex Type RequestAbstractType	26
163	3.2.1.1. Element <RespondWith>	27
164	3.2.2. Element <Request>	27
165	3.2.3. Element <AssertionArtifact>	28
166	3.3. QUERIES	28
167	3.3.1. Element <Query>	28
168	3.3.2. Element <SubjectQuery>	29
169	3.3.3. Element <AuthenticationQuery>	29
170	3.3.4. Element <AttributeQuery>	30
171	3.3.5. Element <AuthorizationDecisionQuery>	30
172	3.4. RESPONSES	31
173	3.4.1. Complex Type ResponseAbstractType	31
174	3.4.2. Element <Response>	32
175	3.4.3. Element <Status>	32
176	3.4.3.1. Element <StatusCode>	33
177	3.4.3.2. Element <StatusMessage>	34
178	3.4.3.3. Element <StatusDetail>	34
179	3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>	34
180	4. SAML VERSIONING	35
181	4.1. ASSERTION VERSION	35

182	4.2. REQUEST VERSION	35
183	4.3. RESPONSE VERSION	36
184	5. SAML & XML SIGNATURE SYNTAX AND PROCESSING	37
185	5.1. SIGNING ASSERTIONS	37
186	5.2. REQUEST /RESPONSE SIGNING	38
187	5.3. SIGNATURE INHERITANCE	38
188	5.3.1. <i>Rationale</i>	38
189	5.3.2. <i>Rules for SAML Signature Inheritance</i>	38
190	5.4. XML SIGNATURE PROFILE	38
191	5.4.1. <i>Signing formats</i>	38
192	5.4.2. <i>CanonicalizationMethod</i>	38
193	5.4.3. <i>Transforms</i>	39
194	5.4.4. <i>KeyInfo</i>	39
195	5.4.5. <i>Binding between statements in a multi-statement assertion</i>	39
196	6. SAML EXTENSIONS	40
197	6.1. ASSERTION SCHEMA EXTENSION	40
198	6.2. PROTOCOL SCHEMA EXTENSION	40
199	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	41
200	7. SAML DEFINED IDENTIFIERS	42
201	7.1. AUTHENTICATION METHOD AND CONFIRMATION METHOD IDENTIFIERS	42
202	7.1.1. <i>SAML Artifact (SHA-1):</i>	42
203	7.1.2. <i>Holder of Key:</i>	43
204	7.1.3. <i>Bearer Indication:</i>	43
205	7.1.4. <i>Sender Vouches:</i>	43
206	7.1.5. <i>Password (Pass Through):</i>	43

207	<i>7.1.6. Password (One-Way Function SHA-1):</i>	43
208	<i>7.1.7. Kerberos</i>	43
209	<i>7.1.8. SSL/TLS Certificate Based Client Authentication:</i>	43
210	<i>7.1.9. Object Authenticator (SHA-1):</i>	44
211	<i>7.1.10. PKCS#7</i>	44
212	<i>7.1.11. Cryptographic Message Syntax</i>	44
213	<i>7.1.12. XML Digital Signature</i>	44
214	7.2. ACTION NAMESPACE IDENTIFIERS	44
215	<i>7.2.1. Read/Write/Execute/Delete/Control:</i>	44
216	<i>7.2.2. Read/Write/Execute/Delete/Control with Negation:</i>	45
217	<i>7.2.3. Get/Head/Put/Post:</i>	45
218	<i>7.2.4. UNIX File Permissions:</i>	45
219	8. SAML SCHEMA LISTINGS	47
220	8.1. ASSERTION SCHEMA	47
221	8.2. PROTOCOL SCHEMA	50
222	9. REFERENCES	53
223	10. ACKNOWLEDGEMENTS	55
224	APPENDIX A. NOTICES	57
225		

226

1. Introduction

227
228
229
230
231
232

This specification defines the syntax and semantics for XML-encoded 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 [SAMPL-XSD] are available.

233

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

234

1.1. Notation

235
236
237

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.

238
239
240

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]:

241
242

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

243
244
245
246

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.

247
248

Listings of SAML schemas appear like this.

249

Example code listings appear like this.

250
251
252

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:

253

?? The prefix `saml`: stands for the SAML assertion namespace.

254

?? The prefix `samlp`: stands for the SAML request-response protocol namespace.

255

?? The prefix `ds`: stands for the W3C XML Signature namespace.

256
257

?? The prefix `xsd`: stands for the W3C XML Schema namespace in example listings. In schema listings, this is the default namespace and no prefix is shown.

258
259

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

260

1.2. Schema Organization and Namespaces

261
262

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

263

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

264
265

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

266

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

267 **Note:** The SAML namespace names are temporary and will change when
268 SAML 1.0 is finalized.

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

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

272 **1.2.1. String and URI Values**

273 All SAML string and URI values have the types string and anyURI respectively, which are built in to
274 the W3C XML Schema Datatypes specification. All strings in SAML messages **MUST** consist of at
275 least one non-whitespace character (whitespace is defined in [XML 1.0 Sec. 2.3]). Empty and
276 whitespace-only values are disallowed. Also, unless otherwise indicated in this specification, all URI
277 values **MUST** consist of at least one non-whitespace character.

278 **1.2.2. Time Values.**

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

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

284 **1.2.3. Comparing SAML values**

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

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

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

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

306 **1.3. SAML Concepts (Non-Normative)**

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

309

1.3.1. Overview

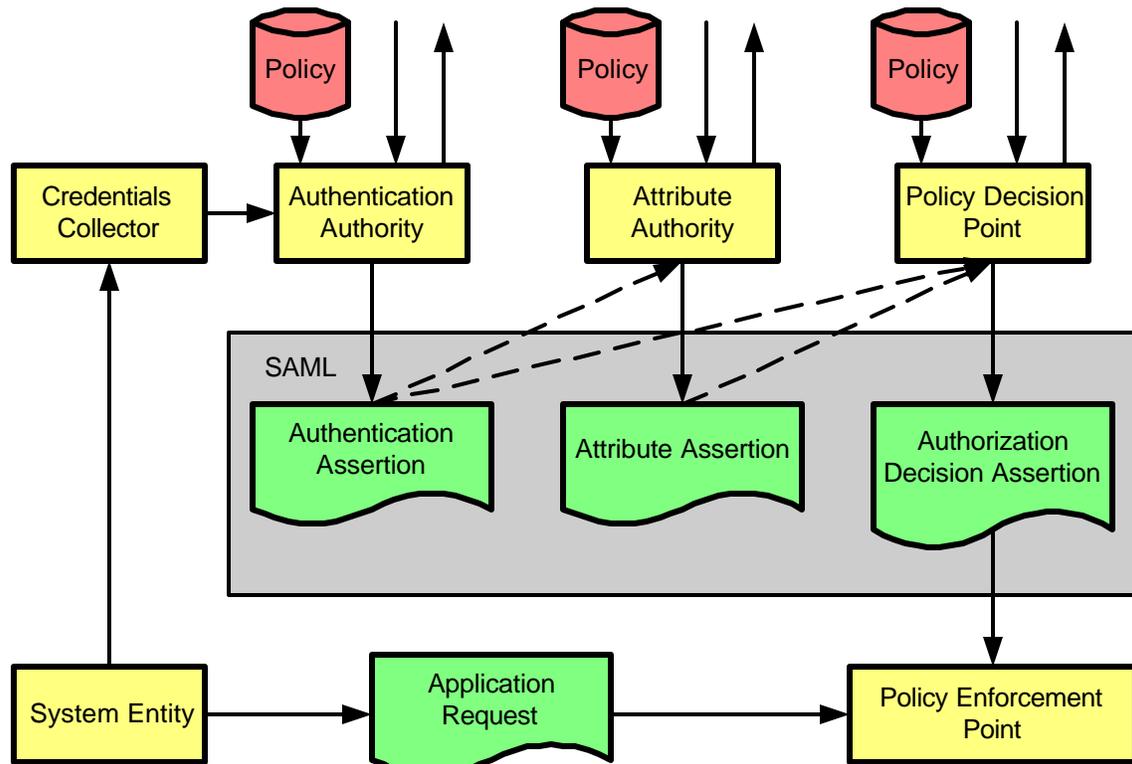
310 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging
311 security information. This security information is expressed in the form of assertions about subjects,
312 where a subject is an entity (either human or computer) that has an identity in some security
313 domain.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

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

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

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

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



331

332

Figure 1 The SAML Domain Model

333 One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in
334 one domain and use resources in other domains without re-authenticating. However, SAML can be

335 used in various configurations to support additional scenarios as well. Several profiles of SAML are
336 currently being defined that support different styles of SSO and the securing of SOAP payloads.

337 The assertion and protocol data formats are defined in this specification. The bindings and profiles
338 are defined in a separate specification [**SAMLBind**]. A conformance program for SAML is defined
339 in the conformance specification [**SAMLConform**]. Security issues are discussed in a separate
340 security and privacy considerations specification [**SAMLSecure**].

341 **1.3.2. SAML and URI-Based Identifiers**

342 SAML defines some identifiers to manage references to well-known concepts and sets of values.
343 For example, the SAML-defined identifier for the [Kerberos subject confirmation password](#)
344 [authentication](#) method is as follows:

345 [urn:oasis:names:tc:SAML:1.0:am:password](#)
346 [urn:ietf:rfc:1540](#)

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

349 [urn:oasis:names:tc:SAML:1.0:action:rwedc](#)

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

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

360 See section 7 for a list of SAML-defined identifiers.

361 **1.3.3. SAML and Extensibility**

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

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

366 2. SAML Assertions

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

369 ?? **Authentication:** The specified subject was authenticated by a particular means at a
370 particular time.

371 ?? **Authorization Decision:** A request to allow the specified subject to access the specified
372 resource has been granted or denied.

373 ?? **Attribute:** The specified subject is associated with the supplied attributes.

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

377 2.1. Schema Header and Namespace Declarations

378 The following schema fragment defines the XML namespaces and other header information for the
379 assertion schema:

```
380 <schema  
381   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
382   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
383   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
384   xmlns="http://www.w3.org/2001/XMLSchema"  
385   elementFormDefault="unqualified">  
386   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
387     schemaLocation="xmldsig-core-schema.xsd"/>  
388   <annotation>  
389     <documentation>draft-sstc-schema-assertion-3130.xsd</documentation>  
390   </annotation>  
391   ...  
392 </schema>
```

393 2.2. Simple Types

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

395 2.2.1. Simple Types IDType and IDReferenceType

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

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

399 ?? Any party that assigns an identifier MUST ensure that there is negligible probability that that
400 party or any other party will accidentally assign the same identifier to a different data object.

401 ?? Where a data object declares that it has a particular identifier, there MUST be exactly one
402 such declaration.

403 The mechanism by which the SAML Requestor or Responder ensures that the identifier is unique is
404 left to the implementation. In the case that a pseudorandom technique is employed, the probability
405 of two randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less
406 than 2^{-160} . This requirement MAY be met by applying Base64 encoding to a randomly chosen value
407 128 or 160 bits in length.

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

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

```
412 <simpleType name="IDType">  
413   <restriction base="string"/>  
414 </simpleType>  
415 <simpleType name="IDReferenceType">  
416   <restriction base="string"/>  
417 </simpleType>
```

418 2.2.2. Simple Type DecisionType

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

421 Permit

422 The specified action is permitted.

423 Deny

424 The specified action is denied.

425 IndeterminateThe issuer cannot determine whether the specified action is permitted or denied.

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

430

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

```
432 <simpleType name="DecisionType">  
433   <restriction base="string">  
434     <enumeration value="Permit"/>  
435     <enumeration value="Deny"/>  
436     <enumeration value="Indeterminate"/>  
437   </restriction>  
438 </simpleType>
```

439 2.3. Assertions

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

441 2.3.1. Element <AssertionID>

442 The <AssertionID> element makes a reference to a SAML assertion by means of the value of
443 the assertion's AssertionID attribute.

444 The following schema fragment defines the <AssertionID> element:

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

446 2.3.2. Element <Assertion>

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

449 MajorVersion [Required]

450 The major version of this assertion. The identifier for the version of SAML defined in this
451 specification is 1. Processing of this attribute is specified in Section 3.4.4.

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

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

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

463 IssueInstant [Required]
 464 The time instant of issue in UTC as described in section 1.2.1.

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

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

470 <Signature> [Optional]
 471 An XML Signature that authenticates the assertion, see section 5.

472 One or more of the following statement elements:

473 <Statement>
 474 A statement defined in an extension schema.

475 <SubjectStatement>
 476 A subject statement defined in an extension schema.

477 <AuthenticationStatement>
 478 An authentication statement.

479 <AuthorizationDecisionStatement>
 480 An authorization decision statement.

481 <AttributeStatement>
 482 An attribute statement.

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

```

485 <element name="Assertion" type="saml:AssertionType"/>
486 <complexType name="AssertionType">
487   <sequence>
488     <element ref="saml:Conditions" minOccurs="0"/>
489     <element ref="saml:Advice" minOccurs="0"/>
490     <choice maxOccurs="unbounded">
491       <element ref="saml:Statement"/>
492       <element ref="saml:SubjectStatement"/>
493       <element ref="saml:AuthenticationStatement"/>
494       <element ref="saml:AuthorizationDecisionStatement"/>
495       <element ref="saml:AttributeStatement"/>
496     </choice>
497     <element ref="ds:Signature" minOccurs="0"/>
498   </sequence>
499   <attribute name="MajorVersion" type="integer" use="required"/>
500   <attribute name="MinorVersion" type="integer" use="required"/>
501   <attribute name="AssertionID" type="saml:IDType" use="required"/>
  
```

```
502     <attribute name="Issuer" type="string" use="required"/>
503     <attribute name="IssueInstant" type="dateTime" use="required"/>
504 </complexType>
```

505 2.3.2.1. Element <Conditions>

506 ~~If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the~~
507 ~~conditions provided. Each condition evaluates to a status of **Valid**, **Invalid**, or **Indeterminate**.~~

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

509 NotBefore [Optional]

510 Specifies the earliest time instant at which the assertion is valid. The time value is encoded
511 in UTC as described in section 1.2.1.

512 NotOnOrAfter [Optional]

513 Specifies the time instant at which the assertion has expired. The time value is encoded in
514 UTC as described in section 1.2.1.

515 <Condition> [Any Number]

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

517 <AudienceRestrictionCondition> [Any Number]

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

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

```
521     <element name="Conditions" type="saml:ConditionsType"/>
522     <complexType name="ConditionsType">
523         <choice minOccurs="0" maxOccurs="unbounded">
524             <element ref="saml:AudienceRestrictionCondition"/>
525             <element ref="saml:Condition"/>
526         </choice>
527         <attribute name="NotBefore" type="dateTime" use="optional"/>
528         <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
529     </complexType>
```

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

- 534 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the
535 assertion is considered to be **Valid**.
- 536 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid,
537 then the assertion is **Invalid**.
- 538 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then
539 the validity of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 540 4. If all sub-elements and attributes of the <Conditions> element are determined to be
541 **Valid**, then the assertion is considered to be **Valid**.

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

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

548 **2.3.2.1.1 Attributes *NotBefore* and *NotOnOrAfter***

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

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

552 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
553 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to
554 `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter`
555 attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied
556 evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore`
557 attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied
558 evaluate to `Valid`), the assertion is valid at any time.

559 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that
560 is built in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are
561 specified in Universal Coordinated Time (UTC) as described in section 1.2.1. Implementations
562 MUST NOT generate time instants that specify leap seconds.

563 **2.3.2.1.2 Element *<Condition>***

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

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

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

571 **2.3.2.1.3 Elements *<AudienceRestrictionCondition>* and *<Audience>***

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

577 `<Audience>`

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

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

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

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

```
590 <element name="AudienceRestrictionCondition"  
591 type="saml:AudienceRestrictionConditionType"/>  
592 <complexType name="AudienceRestrictionConditionType">  
593 <complexContent>  
594 <extension base="saml:ConditionAbstractType">
```

```

595         <sequence>
596             <element ref="saml:Audience" maxOccurs="unbounded" />
597         </sequence>
598     </extension>
599 </complexContent>
600 </complexType>
601 <element name="Audience" type="anyURI" />

```

602 2.3.2.2. Elements <Advice> and <AdviceElement>

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

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

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

- 610 ?? Include evidence supporting the assertion claims to be cited, either directly (through
- 611 incorporating the claims) or indirectly (by reference to the supporting assertions).
- 612 ?? State a proof of the assertion claims.
- 613 ?? Specify the timing and distribution points for updates to the assertion.

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

```

615 <element name="Advice" type="saml:AdviceType" />
616 <complexType name="AdviceType">
617     <choice minOccurs="0" maxOccurs="unbounded">
618         <element ref="saml:AssertionIDReference" />
619         <element ref="saml:Assertion" />
620         <any namespace="##other" processContents="lax" />
621     </choice>
622 </complexType>

```

623 2.4. Statements

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

625 2.4.1. Element <Statement>

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

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

```

631 <element name="Statement" type="saml:StatementAbstractType" />
632 <complexType name="StatementAbstractType" abstract="true" />

```

633 2.4.2. Element <SubjectStatement>

634 The <SubjectStatement> element is an extension point that allows other assertion-based
635 applications to reuse the SAML assertion framework. It contains a <Subject> element that allows
636 an issuer to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends
637 **StatementAbstractType**, is abstract; extension elements MUST use the `xsi:type` attribute to
638 indicate the derived type.

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

```
641 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
642 <complexType name="SubjectStatementAbstractType" abstract="true">  
643 <complexContent>  
644 <extension base="saml:StatementAbstractType">  
645 <sequence>  
646 <element ref="saml:Subject"/>  
647 </sequence>  
648 </extension>  
649 </complexContent>  
650 </complexType>
```

651 2.4.2.1. Element <Subject>

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

654 <NameIdentifier>

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

656 <SubjectConfirmation>

657 Information that allows the subject to be authenticated.

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

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

```
665 <element name="Subject" type="saml:SubjectType"/>  
666 <complexType name="SubjectType">  
667 <choice>  
668 <sequence>  
669 <element ref="saml:NameIdentifier"/>  
670 <element ref="saml:SubjectConfirmation" minOccurs="0"/>  
671 </sequence>  
672 <element ref="saml:SubjectConfirmation"/>  
673 </choice>  
674 </complexType>
```

675 2.4.2.2. Element <NameIdentifier>

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

678 NameQualifier [Optional]

679 The security or administrative domain that qualifies the name of the subject.

680 The NameQualifier attribute provides a means to federate names from disparate user
681 stores without collision.

682 Format [Optional]

683 The syntax used to describe the name of the subject

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

687 #emailAddress

688 Indicates that the content of the NameIdentifier element is in the form of an email address,

689 specifically "addr-spec" as defined in section 3.4.1 of RFC 2822 [RFC 2822]. An addr-spec
690 has the form local-part@domain. Note that an addr-spec has no phrase (such as a
691 common name) before it, has no comment (text surrounded in parentheses) after it, and is
692 not surrounded by "<" and ">".

693 #X509SubjectName

694 Indicates that the content of the NameIdentifier element is in the form specified for
695 the contents of <ds:X509SubjectName> element in [DSIG]. Implementors should note that
696 [DSIG] specifies encoding rules for X.509 subject names that differ from the rules given in
697 RFC2253 [RFC2253].

698 #WindowsDomainQualifiedName

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

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

```
704 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
705 <complexType name="NameIdentifierType">  
706 <simpleContent>  
707 <extension base="string">  
708 <attribute name="NameQualifier" type="string" use="optional"/>  
709 <attribute name="Format" type="anyURI" use="optional"/>  
710 </extension>  
711 </simpleContent>  
712 </complexType>
```

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

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

718 **2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 719 **<SubjectConfirmationData>**

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

722 <ConfirmationMethod> [One or more]

723 A URI reference that identifies a protocol to be used to authenticate the subject. URI
724 references identifying common authentication protocols are listed in Section 7. SAML-
725 defined confirmation methods are currently defined with the SAML profiles in [SAMLBind].
726 Additional SAML confirmation methods may be defined in future OASIS-approved SAML
727 profile specifications.

728 <SubjectConfirmationData> [Optional]

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

730 <ds:KeyInfo> [Optional]

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

733 The following schema fragment defines the <SubjectConfirmation> element and its
734 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData>
735 element and the <ConfirmationMethod> element:

```
736 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
737 <complexType name="SubjectConfirmationType">  
738 <sequence>
```

```

739     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded" />
740     <element ref="saml:SubjectConfirmationData" minOccurs="0" />
741     <element ref="ds:KeyInfo" minOccurs="0" />
742   </sequence>
743 </complexType>
744 <element name="SubjectConfirmationData" type="stringanyType" />
745 <element name="ConfirmationMethod" type="anyURI" />

```

746 2.4.3. Element <AuthenticationStatement>

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

751 AuthenticationMethod [Optional]

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

754 AuthenticationInstant [Optional]

755 Specifies the time at which the authentication took place. The time value is encoded in UTC
756 as described in section 1.2.1.

757 <SubjectLocality> [Optional]

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

760 <AuthorityBinding> [Any Number]

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

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

```

764 <element name="AuthenticationStatement"
765     type="saml:AuthenticationStatementType" />
766 <complexType name="AuthenticationStatementType">
767   <complexContent>
768     <extension base="saml:SubjectStatementAbstractType">
769       <sequence>
770         <element ref="saml:SubjectLocality" minOccurs="0" />
771         <element ref="saml:AuthorityBinding"
772             minOccurs="0" maxOccurs="unbounded" />
773       </sequence>
774       <attribute name="AuthenticationMethod" type="anyURI" />
775       <attribute name="AuthenticationInstant" type="dateTime" />
776     </extension>
777   </complexContent>
778 </complexType>

```

779 2.4.3.1. Element <SubjectLocality>

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

782 IPAddress [Optional]

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

784 DNSAddress [Optional]

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

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

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

```
790 <element name="SubjectLocality"  
791         type="saml: SubjectLocalityType" />  
792 <complexType name="SubjectLocalityType">  
793     <attribute name="IPAddress" type="string" use="optional" />  
794     <attribute name="DNSAddress" type="string" use="optional" />  
795 </complexType>
```

796 2.4.3.2. Element <AuthorityBinding>

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

802 AuthorityKind [Required]

803 The type of SAML Protocol queries to which the authority described by this element will
804 respond. The value is specified as an XML Schema QName. The acceptable values for
805 AuthorityKind are the namespace-qualified names of element types or elements
806 derived from the SAML Protocol Query element (see Section 3.3). For example, an
807 attribute authority would be identified by AuthorityKind="samlp:AttributeQuery".
808 For extension schemas, where the actual type of the samlp:Query would be identified by
809 an xsi:type attribute, the value of AuthorityKind MUST be the same as the value of
810 the xsi:type attribute for the corresponding query.

811 Location [Required]

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

815 Binding [Required]

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

818 The following schema fragment defines the <AuthorityBinding> element and its
819 **AuthorityBindingType** complex type and **AuthorityKindType** simple type:

```
820 <element name="AuthorityBinding" type="saml:AuthorityBindingType" />  
821 <complexType name="AuthorityBindingType">  
822     <attribute name="AuthorityKind" type="QName" use="required" />  
823     <attribute name="Location" type="anyURI" use="required" />  
824     <attribute name="Binding" type="anyURI" use="required" />  
825 </complexType>
```

826 2.4.4. Element <AuthorizationDecisionStatement>

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

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

835 *In general, the rules for equivalence and definition of a normal form, if any, are scheme*
836 *dependent. When a scheme uses elements of the common syntax, it will also use the common*
837 *syntax equivalence rules, namely that the scheme and hostname are case insensitive and a*

838 *URL with an explicit ":port", where the port is the default for the scheme, is equivalent to one*
839 *where the port is elided.*

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

842 ?? The assertion issuer SHOULD encode all resource URIs in normalized form.

843 ?? Relying parties SHOULD convert resource URIs to normalized form prior to processing.

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

848 ?? Parts of the URI syntax are case sensitive. If the underlying file system is case insensitive a
849 requestor SHOULD NOT be able to gain access to a denied resource by changing the case
850 of a part of the resource URI.

851 ?? Many file systems support mechanisms such as logical paths and symbolic links which
852 allow users to establish logical equivalences between file system entries. A requestor
853 SHOULD NOT be able to gain access to a denied resource by creating such an
854 equivalence.

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

858 Resource [Required]

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

864 Decision [Required]

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

867 <Action> [One or more]

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

869 <Evidence> [Any Number]

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

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

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

887 2.4.4.1. Element <Action>

888 The <Action> element specifies an action on the specified resource for which permission is
889 sought. It has the following attribute:

890 Namespace [Optional]

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

894 *string data* [Required]

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

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

```
897 <element name="Action" type="saml:ActionType"/>  
898 <complexType name="ActionType">  
899 <simpleContent>  
900 <extension base="string">  
901 <attribute name="Namespace" type="anyURI"/>  
902 </extension>  
903 </simpleContent>  
904 </complexType>
```

905 2.4.4.2. Element <Evidence>

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

909 <AssertionIDReference>

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

911 <Assertion>

912 Specifies an assertion by value.

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

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

```
920 <element name="Evidence" type="saml:EvidenceType"/>  
921 <complexType name="EvidenceType">  
922 <choice maxOccurs="unbounded">  
923 <element ref="saml:AssertionIDReference"/>  
924 <element ref="saml:Assertion"/>  
925 </choice>  
926 </complexType>
```

927 2.4.5. Element <AttributeStatement>

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

931 <Attribute> [One or More]

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

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

```
935 <element name="AttributeStatement" type="saml:AttributeStatementType" />
936 <complexType name="AttributeStatementType">
937   <complexContent>
938     <extension base="saml:SubjectStatementAbstractType">
939       <sequence>
940         <element ref="saml:Attribute" maxOccurs="unbounded" />
941       </sequence>
942     </extension>
943   </complexContent>
944 </complexType>
```

945 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

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

950 AttributeNamespace [Optional]
951 The namespace in which the AttributeName elements are interpreted.

952 AttributeName [Optional]
953 The name of the attribute.

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

```
956 <element name="AttributeDesignator" type="saml:AttributeDesignatorType" />
957 <complexType name="AttributeDesignatorType">
958   <attribute name="AttributeName" type="string" use="required" />
959   <attribute name="AttributeNamespace" type="anyURI" use="required" />
960 </complexType>
```

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

964 <AttributeValue> [Any Number]
965 The value of the attribute.

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

```
968 <element name="Attribute" type="saml:AttributeType" />
969 <complexType name="AttributeType">
970   <complexContent>
971     <extension base="saml:AttributeDesignatorType">
972       <sequence>
973         <element ref="saml:AttributeValue" maxOccurs="unbounded" />
974       </sequence>
975     </extension>
976   </complexContent>
977 </complexType>
```

978 2.4.5.1.1 Element <AttributeValue>

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

981 If the data content of an AttributeValue element is of a XML Schema simple type (e.g. interger,
982 string, etc) the data type MAY be declared explicitly by means of an `xsi:type` declaration in the

983 <AttributeValue> element. If the attribute value contains structured data the necessary data
984 elements may be defined in an extension schema introduced by means of the xmlns= mechanism.

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

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

3. SAML Protocol

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

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



Figure 2: SAML Request-Response Protocol

3.1. Schema Header and Namespace Declarations

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

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol "
  xmlns="http://www.w3.org/2001/XMLSchema "
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol "
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified ">
  <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
    schemaLocation="draft-sstc-schema-assertion-3130.xsd" />
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd" />
  <annotation>
    <documentation>draft-sstc-schema-protocol-3130.xsd</documentation>
  </annotation>
  ...
</schema>
```

3.2. Requests

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

3.2.1. Complex Type RequestAbstractType

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

RequestID [Required]

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

MajorVersion [Required]

The major version of this request. The identifier for the version of SAML defined in this specification is 1. Processing of this attribute is specified in Section 3.4.2.

1029 MinorVersion [Required]
 1030 The minor version of this request. The identifier for the version of SAML defined in this
 1031 specification is 0. Processing of this attribute is specified in Section 3.4.2.

1032 IssueInstant [Required]
 1033 The time instant of issue of the request. The time value is encoded in UTC as described in
 1034 section 1.2.1.

1035 <RespondWith> [Any Number]
 1036 Each <RespondWith> element specifies a type of response that is acceptable to the
 1037 requestor.

1038 <Signature> [Optional]
 1039 An XML Signature that authenticates the assertion, see section 5.

1040 The following schema fragment defines the **RequestAbstractType** complex type:

```

1041 <complexType name="RequestAbstractType" abstract="true">
1042   <sequence>
1043     <element ref="samlp:RespondWith"
1044       minOccurs="0" maxOccurs="unbounded"/>
1045     <element ref="ds:Signature" minOccurs="0"/>
1046   </sequence>
1047   <attribute name="RequestID" type="saml:IDType" use="required"/>
1048   <attribute name="MajorVersion" type="integer" use="required"/>
1049   <attribute name="MinorVersion" type="integer" use="required"/>
1050   <attribute name="IssueInstant" type="dateTime" use="required"/>
1051 </complexType>

```

1052 3.2.1.1. Element <RespondWith>

1053 The <RespondWith> element specifies the type of Statement the requestor wants from the
 1054 responder. Multiple <RespondWith> elements MAY be included to indicate that the requestor will
 1055 accept assertions containing any of the specified types. If no <RespondWith> element is given,
 1056 the responder may return assertions containing statements of any type.

1057 If the requestor sends one or more <RespondWith> elements, the responder MUST NOT respond
 1058 with assertions containing statements of any type not specified in one of the <RespondWith>
 1059 elements.

1060 NOTE: Inability to find assertions that meet <RespondWith> criteria should be treated identical to
 1061 any other query for which no assertions are available. In both cases a status of success would
 1062 normally be returned in the Response message, but no assertions to be found therein.

1063 <RespondWith> element values are XML QNames. The XML namespace and name specifically
 1064 refer to the namespace and element name of the Statement element, exactly as for the
 1065 saml:AuthorityKind attribute; see section 2.4.3.2. For example, a requestor that wishes to
 1066 receive assertions containing only attribute statements must specify
 1067 <RespondWith>saml:AttributeStatement</RespondWith>. To specify extension types,
 1068 the <RespondWith> element MUST contain exactly the extension element type as specified in the
 1069 xsi:type attribute on the corresponding element.

1070 The following schema fragment defines the <RespondWith> element:

```

1071 <element name="RespondWith" type="QName"/>

```

1072 3.2.2. Element <Request>

1073 The <Request> element specifies a SAML request. It provides either a query or a request for a
 1074 specific assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has

1075 the complex type **RequestType**, which extends **RequestAbstractType** by adding a choice of one
1076 of the following elements:

1077 <Query>

1078 An extension point that allows extension schemas to define new types of query.

1079 <SubjectQuery>

1080 An extension point that allows extension schemas to define new types of query that specify
1081 a single SAML subject.

1082 <AuthenticationQuery>

1083 Makes a query for authentication information.

1084 <AttributeQuery>

1085 Makes a query for attribute information.

1086 <AuthorizationDecisionQuery>

1087 Makes a query for an authorization decision.

1088 <AssertionIDReference> [One or more]

1089 Requests assertions by reference to its assertion identifier.

1090 <AssertionArtifact> [One or more]

1091 Requests assertions by supplying an assertion artifact that represents it.

1092 The following schema fragment defines the <Request> element and its **RequestType** complex
1093 type:

```
1094 <element name="Request" type="samlp:RequestType"/>
1095 <complexType name="RequestType">
1096   <complexContent>
1097     <extension base="samlp:RequestAbstractType">
1098       <choice>
1099         <element ref="samlp:Query"/>
1100         <element ref="samlp:SubjectQuery"/>
1101         <element ref="samlp:AuthenticationQuery"/>
1102         <element ref="samlp:AttributeQuery"/>
1103         <element ref="samlp:AuthorizationDecisionQuery"/>
1104         <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>
1105         <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
1106       </choice>
1107     </extension>
1108   </complexContent>
1109 </complexType>
```

1110 3.2.3. Element <AssertionArtifact>

1111 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
1112 assertion.

1113 The following schema fragment defines the <AssertionArtifact> element:

```
1114 <element name="AssertionArtifact" type="string"/>
```

1115 3.3. Queries

1116 The following sections define the SAML constructs that contain query information.

1117 3.3.1. Element <Query>

1118 The <Query> element is an extension point that allows new SAML queries to be defined. Its
1119 **QueryAbstractType** is abstract; extension elements **MUST** use the `xsi:type` attribute to indicate

1120 the derived type. **QueryAbstractType** is the base type from which all SAML query elements are
1121 derived.

1122 The following schema fragment defines the <Query> element and its **QueryAbstractType**
1123 complex type:

```
1124 <element name="Query" type="samlp:QueryAbstractType" />  
1125 <complexType name="QueryAbstractType" abstract="true" />
```

1126 3.3.2. Element <SubjectQuery>

1127 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a
1128 single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements
1129 MUST use the `xsi:type` attribute to indicate the derived type. **SubjectQueryAbstractType** adds
1130 the <Subject> element.

1131 The following schema fragment defines the <SubjectQuery> element and its
1132 **SubjectQueryAbstractType** complex type:

```
1133 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType" />  
1134 <complexType name="SubjectQueryAbstractType" abstract="true">  
1135 <complexContent>  
1136 <extension base="samlp:QueryAbstractType">  
1137 <sequence>  
1138 <element ref="saml:Subject" />  
1139 </sequence>  
1140 </extension>  
1141 </complexContent>  
1142 </complexType>
```

1143 3.3.3. Element <AuthenticationQuery>

1144 The <AuthenticationQuery> element is used to make the query “What assertions containing
1145 authentication statements are available for this subject?” A successful response will be in the form
1146 of assertions containing authentication statements.

1147 Note: The <AuthenticationQuery> MAY NOT be used as a request for a new authentication
1148 using credentials provided in the request. The <AuthenticationQuery> is a request for
1149 statements about authentication acts which have occurred in a previous interaction between the
1150 indicated principal and the Authentication Authority.

1151 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType**
1152 with the addition of the following element:

1153 <AuthenticationMethod> [Optional]

1154 A filter for possible responses. If it is present, the query made is “What assertions
1155 containing authentication statements do you have for this subject with the supplied
1156 authentication method?”

1157 In response to an authentication query, a responder returns assertions with authentication
1158 statements as follows:

1159 ?? First, rules given in section 3.4.4 for matching against the <Subject> element of the query
1160 identify the assertions that may be returned.

1161 ?? Further, if the <AuthenticationMethod> element is present in the query, at least one
1162 <AuthenticationMethod> element in the set of returned assertions MUST match. It is
1163 OPTIONAL for the complete set of all such matching assertions to be returned in the
1164 response.

1165 The <Subject> element in the returned assertions MUST be identical to the <Subject> element
1166 of the query. If the <ConfirmationMethod> element is present in the query, at least one

1167 <ConfirmationMethod> element in the response MUST match. It is OPTIONAL for the complete
1168 set of all such matching assertions to be returned in the response.

1169 The following schema fragment defines the <AuthenticationQuery> type and its
1170 **AuthenticationQueryType** complex type:

```
1171     <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
1172     <complexType name="AuthenticationQueryType">  
1173         <complexContent>  
1174             <extension base="samlp:SubjectQueryAbstractType">  
1175                 <attribute name="AuthenticationMethod" type="anyURI"/>  
1176             </extension>  
1177         </complexContent>  
1178     </complexType>
```

1179 3.3.4. Element <AttributeQuery>

1180 The <AttributeQuery> element is used to make the query “Return the requested attributes for
1181 this subject.” A successful response will be in the form of assertions containing attribute statements.
1182 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the
1183 addition of the following element and attribute:

1184 Resource [Optional]

1185 The Resource attribute if present specifies that the attribute query is made in response to a
1186 specific authorization decision relating to the resource. The responder MAY use the
1187 resource attribute to establish the scope of the request. It is permitted for this attribute to
1188 have the value of the empty URI reference (“”), and the meaning is defined to be “the start
1189 of the current document”, as specified by **[RFC 2396]**§ 4.2.

1190 If the resource attribute is specified and the responder does not wish to support resource-
1191 specific attribute queries, or if the resource value provided is invalid or unrecognized, then it
1192 SHOULD respond with a top-level StatusCode value of Responder and a second-level
1193 code value of ResourceNotRecognized

1194 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1195 Each <AttributeDesignator> element specifies an attribute whose value is to be
1196 returned. If no attributes are specified, it indicates that all attributes allowed by policy are
1197 requested.

1198 The following schema fragment defines the <AttributeQuery> element and its
1199 **AttributeQueryType** complex type:

```
1200     <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1201     <complexType name="AttributeQueryType">  
1202         <complexContent>  
1203             <extension base="samlp:SubjectQueryAbstractType">  
1204                 <sequence>  
1205                     <element ref="saml:AttributeDesignator"  
1206                         minOccurs="0" maxOccurs="unbounded"/>  
1207                 </sequence>  
1208                 <attribute name="Resource" type="anyURI reference" use="optional"/>  
1209             </extension>  
1210         </complexContent>  
1211     </complexType>
```

1212 3.3.5. Element <AuthorizationDecisionQuery>

1213 The <AuthorizationDecisionQuery> element is used to make the query “Should these
1214 actions on this resource be allowed for this subject, given this evidence?” A successful response
1215 will be in the form of assertions containing authorization decision statements. This element is of
1216 type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1217 addition of the following elements and attribute:

1218 Resource [Required]
1219 A URI reference indicating the resource for which authorization is requested.
1220 <Action> [One or More]
1221 The actions for which authorization is requested.
1222 <Evidence> [Any Number]
1223 An assertion that the responder MAY rely on in making its response.

1224 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
1225 **AuthorizationDecisionQueryType** complex type:

```
1226 <element name="AuthorizationDecisionQuery "  
1227 type="samlp:AuthorizationDecisionQueryType"/>  
1228 <complexType name="AuthorizationDecisionQueryType ">  
1229 <complexContent>  
1230 <extension base="samlp:SubjectQueryAbstractType">  
1231 <sequence>  
1232 <element ref="saml:Action" maxOccurs="unbounded"/>  
1233 <element ref="saml:Evidence"  
1234 minOccurs="0" maxOccurs="unbounded" />  
1235 </sequence>  
1236 <attribute name="Resource" type="anyURI" use="required" />  
1237 </extension>  
1238 </complexContent>  
1239 </complexType>
```

1240 3.4. Responses

1241 The following sections define the SAML constructs that contain response information.

1242 3.4.1. Complex Type ResponseAbstractType

1243 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**
1244 complex type. This type defines common attributes and elements that are associated with all SAML
1245 responses:

1246 ResponseID [Required]
1247 An identifier for the response. It is of type **IDType**, and MUST follow the requirements
1248 specified by that type for identifier uniqueness.

1249 InResponseTo [Optional]
1250 A reference to the identifier of the request to which the response corresponds, if any. If the
1251 response is not generated in response to a request, or if the RequestID of a request cannot
1252 be determined (because the request is malformed), then this attribute MUST NOT be
1253 present. Otherwise, it MUST be present and match the value of the corresponding
1254 RequestID attribute.

1255 MajorVersion [Required]
1256 The major version of this response. The identifier for the version of SAML defined in this
1257 specification is 1. Processing of this attribute is specified in Section 3.4.4.

1258 MinorVersion [Required]
1259 The minor version of this response. The identifier for the version of SAML defined in this
1260 specification is 0. Processing of this attribute is specified in Section 3.4.4.

1261 IssueInstant [Optional]
1262 The time instant of issue of the request. The time value is encoded in UTC as described in
1263 section 1.2.1.

1264 Recipient [Optional]
1265 The intended recipient of this response. This is useful to prevent malicious forwarding of
1266 responses to unintended recipients, a protection that is required by some use profiles. It is
1267 set by the generator of the response to a URI reference that identifies the intended
1268 recipient. If present, the actual recipient MUST check that the URI reference identifies the
1269 recipient or a resource managed by the recipient. If it does not, the response MUST be
1270 discarded.

1271 <Signature> [Optional]
1272 An XML Signature that authenticates the assertion, see section 5.

1273 The following schema fragment defines the **ResponseAbstractType** complex type:

```
1274 <complexType name="ResponseAbstractType" abstract="true" >  
1275 <sequence>  
1276 <element ref = "ds:Signature" minOccurs="0"/>  
1277 </sequence>  
1278 <attribute name="ResponseID" type="saml:IDType" use="required"/>  
1279 <attribute name="InResponseTo" type="saml:IDReferenceType"  
1280 use="optional"/>  
1281 <attribute name="MajorVersion" type="integer" use="required"/>  
1282 <attribute name="MinorVersion" type="integer" use="required"/>  
1283 <attribute name="IssueInstant" type="dateTime" use="required"/>  
1284 <attribute name="Recipient" type="anyURI" use="optional"/>  
1285 </complexType>
```

1286 3.4.2. Element <Response>

1287 The <Response> element specifies the status of the corresponding SAML request and a list of
1288 zero or more assertions that answer the request. It has the complex type **ResponseType**, which
1289 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1290 <Status> [Required] (see Section 3.4.3)
1291 A code representing the status of the corresponding request.

1292 <Assertion> [Any Number] (see Section 2.3.2)
1293 Specifies an assertion by value.

1294 The following schema fragment defines the <Response> element and its **ResponseType** complex
1295 type:

```
1296 <element name="Response" type="samlp:ResponseType" />  
1297 <complexType name="ResponseType" >  
1298 <complexContent>  
1299 <extension base="samlp:ResponseAbstractType" >  
1300 <sequence>  
1301 <element ref="samlp:Status" />  
1302 <element ref="saml:Assertion"  
1303 minOccurs="0" maxOccurs="unbounded" />  
1304 </sequence>  
1305 </extension>  
1306 </complexContent>  
1307 </complexType>
```

1308 3.4.3. Element <Status>

1309 The <Status> element :

1310 <StatusCode> [Required]
1311 A code representing the status of the corresponding request.

1312 <StatusMessage> [Any Number]
1313 A message which MAY be returned to an operator.

1314 <StatusDetail> [Optional]
1315 Specifies additional information concerning an error condition.
1316 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1317 <element name="Status" type="samlp:StatusType"/>  
1318 <complexType name="StatusType">  
1319 <sequence>  
1320 <element ref="samlp:StatusCode"/>  
1321 <element ref="samlp:StatusMessage"  
1322 minOccurs="0" maxOccurs="unbounded"/>  
1323 <element ref="samlp:StatusDetail" minOccurs="0"/>  
1324 </sequence>  
1325 </complexType>
```

1326 3.4.3.1. Element <StatusCode>

1327 The <StatusCode> element specifies one or more nested codes representing the status of the
1328 corresponding request. top-most code value MUST be one of the values defined below.
1329 Subsequent nested code values, if present, may provide more specific information concerning a
1330 particular error.

1331 Value [Required]
1332 The status code value as defined below.

1333 <StatusCode> [Optional]
1334 An optional subordinate status code value that provides more specific information on an
1335 error condition.

1336 The following top-level **StatusCode** Value QNames are defined. The responder MUST NOT
1337 include a code not listed below except by nesting it below one of the listed values.

1338 Success
1339 The request succeeded.

1340 VersionMismatch
1341 The receiver could not process the request because the version was incorrect.

1342 Receiver
1343 The request could not be performed due to an error at the receiving end.

1344 Sender
1345 The request could not be performed due to an error in the sender or in the request

1346 The following second-level status codes are referenced at various places in the specification.
1347 Additional subcodes MAY be defined in future versions of the SAML specification.

1348 RequestVersionTooHigh
1349 The protocol version specified in the request is a major upgrade from the highest protocol
1350 version supported by the responder.

1351 RequestVersionTooLow
1352 The responder cannot respond to the particular request using the SAML version specified
1353 in the request because it is too low.

1354 RequestVersionDeprecated
1355 The responder does not respond to any requests with the protocol version specified in the
1356 request.

1357 TooManyResponses
1358 The response would contain more elements than the responder will return.

1359 RequestDenied
1360 The responder is able to process the request but has chosen not to respond. MAY be used

1361 when the responder is concerned about the security context of the request or the sequence
1362 of requests received from a particular client.

1363 All status code values defined in this document are QNames associated with the SAML protocol
1364 namespace [SAML] and MUST be prefixed appropriately when they appear in SAML messages.
1365 SAML extensions and SAML Responders are free to define more specific status codes in other
1366 namespaces, but MAY NOT define additional codes in either the SAML assertion or protocol
1367 namespaces.

1368 The QNames defined as status codes SHOULD only be used in the StatusCode element's Value
1369 attribute and have the above semantics only in that context.

1370 The following schema fragment defines the <StatusCode> element and its **StatusCodeType**
1371 complex type:

```
1372 <element name="StatusCode" type="samlp:StatusCodeType" />  
1373 <complexType name="StatusCodeType">  
1374 <sequence>  
1375 <element ref="samlp:StatusCode" minOccurs="0" />  
1376 </sequence>  
1377 <attribute name="Value" type="QName" use="required" />  
1378 </complexType>
```

1379 3.4.3.2. Element <StatusMessage>

1380 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1381 The following schema fragment defines the <StatusMessage> element and its
1382 **StatusMessageType** complex type:

```
1383 <element name="StatusMessage" type="string" />
```

1384 3.4.3.3. Element <StatusDetail>

1385 The <StatusDetail> element MAY be used to specify additional information concerning an error
1386 condition.

1387 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1388 complex type:

```
1389 <element name="StatusDetail" type="samlp:StatusDetailType" />  
1390 <complexType name="StatusDetailType">  
1391 <sequence>  
1392 <any namespace="##any"  
1393 processContents="lax" minOccurs="0" maxOccurs="unbounded" />  
1394 </sequence>  
1395 </complexType>
```

1396 3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>

1397 Responses to Authentication and Attribute queries are constructed by matching against the
1398 <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>
1399 elements. In response to these queries, every assertion returned by a SAML responder MUST
1400 contain at least one statement whose <saml:Subject> element **strongly matches** the
1401 <saml:Subject> element found in the query.

1402 A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1403 1 If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical
1404 <saml:NameIdentifier> element.
- 1405 2 If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an
1406 identical <saml:SubjectConfirmation> element.

1407 If the responder cannot provide an assertion with any statement(s) satisfying the constraints
1408 expressed by a query, the <saml:Response> element MUST NOT contain an <assertion> element
1409 and MUST include a <saml:StatusCode> with value "Success". It MAY return a
1410 <saml:StatusMessage> with additional information.

4. SAML Versioning

1411

1412 SAML version information appears in the following elements:

1413 ?? <Assertion>

1414 ?? <Request>

1415 ?? <Response>

1416 The version numbering of the SAML assertion is independent of the version number of the SAML
1417 request-response protocol. The version information for each consists of a major version number
1418 and a minor version number, both of which are integers. In accordance with industry practice a
1419 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines
1420 SAML Assertions 1.0 and SAML Protocol 1.0.

1421 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

1422 $Major_B > Major_A ? ((Major_B = Major_A) ? Minor_B > Minor_A)$

1423 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that
1424 are the same as or higher than the corresponding version number in the SAML version that
1425 immediately preceded it.

1426 New versions of SAML SHALL assign new version numbers as follows:

1427 ?? **Documentation change:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1428 If the major and minor version numbers are unchanged, the new version *B* only introduces
1429 changes to the documentation that raise no compatibility issues with an implementation of
1430 version *A*.

1431 ?? **Minor upgrade:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1432 If the major version number of versions *A* and *B* are the same and the minor version
1433 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the
1434 SAML schema and semantics but any changes that are introduced in *B* SHALL be
1435 compatible with version *A*.

1436 ?? **Major upgrade:** $Major_B > Major_A$

1437 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY
1438 introduce changes to the SAML schema and semantics that are incompatible with *A*.

4.1. Assertion Version

1439

1440 A SAML authority MUST NOT issue any assertion whose version number is not supported.

1441 | A SAML [authority-relying party](#) MUST reject any assertion whose major version number is not
1442 | supported.

1443 | A SAML [authority-relying party](#) MAY reject any assertion whose version number is higher than the
1444 | highest supported version.

4.2. Request Version

1445

1446 A SAML authority SHOULD issue requests that specify the highest SAML version supported by
1447 both the sender and recipient.

1448 If the SAML authority does not know the capabilities of the recipient it should assume that it
1449 supports the highest SAML version supported by the sender.

1450 **4.3. Response Version**

1451 A SAML authority MUST NOT issue responses that specify a higher SAML version number than the
1452 corresponding request.

1453 A SAML authority MUST NOT issue a response that has a major version number that is lower than
1454 the major version number of the corresponding request except to report the error
1455 `RequestVersionTooHigh`.

1456 An error response resulting from incompatible protocol versions MUST result in reporting a top-level
1457 `StatusCode` value of `VersionMismatch`, and MAY result in reporting one of the following second-
1458 level values:

1459 `RequestVersionTooHigh`

1460 The protocol version specified in the request is a major upgrade from the highest protocol
1461 version supported by the responder.

1462 `RequestVersionTooLow`

1463 The responder cannot respond to the particular request using the SAML version specified
1464 in the request because it is too low.

1465 `RequestVersionDeprecated`

1466 The responder does not respond to any requests with the protocol version specified in the
1467 request.

5. SAML & XML-Signature Syntax and Processing

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

?? An Assertion signed by the asserting party (AP). This supports :

- (1) Message integrity
- (2) Authentication of the asserting party to a relying party (RP)
- (3) If the signature is based on the asserting party's public-private key pair, then it also provides for non-repudiation of origin.

?? A SAML request or a SAML response message signed by the message originator. This supports :

- (1) Message integrity
- (2) Authentication of message origin to a destination
- (3) If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.

Note :

?? SAML documents may be the subject of signatures from different packaging contexts. **[XMLSig]** provides a framework for signing in XML and is the framework of choice. However, signing may also take place in the context of S/MIME or Java objects that contain SAML documents. One goal is to ensure compatibility with this type of "foreign" digital signing.

?? It is useful to characterize situations when a digital signature is NOT required in SAML.

Assertions:

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

Request/Response messages:

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

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

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

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

5.1. Signing Assertions

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

1510 **5.2. Request/Response Signing**

1511 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the
1512 schema – Section 3.2 & 3.4.

1513 **5.3. Signature Inheritance**

1514 **5.3.1. Rationale**

1515 SAML assertions may be embedded within request or response messages or other XML
1516 messages, which may be signed. Request or response messages may themselves be contained
1517 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the
1518 composite object may be the subject of a signature. Another possibility is that SAML assertions or
1519 request/response messages are embedded within a non-XML messaging object (e.g., MIME
1520 package) and signed.

1521 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as inheriting
1522 a signature from the "super-signature" over the enclosing object, provided certain constraints are
1523 met.

1524 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super
1525 signature applies all the elements within the assertion.

1526 A SAML request or response may be viewed as inheriting a signature from a super signature, if the
1527 super signature applies to all of the elements within the response.

1528 **5.3.2. Rules for SAML Signature Inheritance**

1529 Signature inheritance occurs when SAML message (assertion/request/response) is not signed but
1530 is enclosed within signed SAML such that the signature applies to all of the elements within the
1531 message. In such a case, the SAML message is said to inherit the signature and may be
1532 considered equivalent to the case where it is explicitly signed. The SAML message inherits the
1533 "closest enclosing signature".

1534 But if SAML messages need to be passed around by themselves, or embedded in other messages,
1535 they would need to be signed as per section 5.1

1536 **5.4. XML Signature Profile**

1537 The XML Signature [**XMLSig**] specification calls out a general XML syntax for signing data with
1538 many flexibilities and choices. This section details the constraints on these facilities so that SAML
1539 processors do not have to deal with the full generality of XML Signature processing.

1540 **5.4.1. Signing formats**

1541 XML Signature has three ways of representing signature in a document viz: enveloping, enveloped
1542 and detached.

1543 SAML assertions and protocols MUST use the enveloped signatures for signing assertions and
1544 protocols. SAML processors should support use of RSA signing and verification for public key
1545 operations.

1546 **5.4.2. CanonicalizationMethod**

1547 XML Signature REQUIRES the Canonical XML (omits comments)
1548 (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>). SAML implementations SHOULD use
1549 Canonical XML with no comments.

1550 **5.4.3. Transforms**

1551 [XMLSig] REQUIRES the enveloped signature transform
1552 <http://www.w3.org/2000/09/xmlsig#enveloped-signature>

1553 **5.4.4. KeyInfo**

1554 SAML does not restrict or impose any restrictions in this area. Therefore following [XMLSig]
1555 keyInfo may be absent.

1556 **5.4.5. Binding between statements in a multi-statement assertion**

1557 Use of signing does not affect semantics of statements within assertions in any way, as stated in
1558 this document Sections 1 through 4.

1559 **6. SAML Extensions**

1560 The SAML schemas support extensibility. An example of an application that extends SAML
1561 assertions is the XTAML system for management of embedded trust roots **[XTAML]**. The following
1562 sections explain how to use the extensibility features in SAML to create extension schemas.

1563 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML
1564 elements MAY serve as the head element of a substitution group. Also, types are not defined as
1565 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss
1566 only elements that have been specifically designed to support extensibility.

1567 **6.1. Assertion Schema Extension**

1568 The SAML assertion schema is designed to permit separate processing of the assertion package
1569 and the statements it contains, if the extension mechanism is used for either part.

1570 The following elements are intended specifically for use as extension points in an extension
1571 schema; their types are set to *abstract*, so that the use of an *xsi:type* attribute with these
1572 elements is REQUIRED:

1573 ?? <Assertion>

1574 ?? <Condition>

1575 ?? <Statement>

1576 ?? <SubjectStatement>

1577 ?? <AdviceElement>

1578 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1579 ?? <AuthenticationStatement>

1580 ?? <AuthorizationDecisionStatement>

1581 ?? <AttributeStatement>

1582 ?? <AudienceRestrictionCondition>

1583 Finally, the following elements are defined to allow elements from arbitrary namespaces within
1584 them, which serves as a built-in extension point without requiring an extension schema:

1585 ?? <AttributeValue>

1586 ?? <Advice>

1587 **6.2. Protocol Schema Extension**

1588 The following elements are intended specifically for use as extension points in an extension
1589 schema; their types are set to *abstract*, so that the use of an *xsi:type* attribute with these
1590 elements is REQUIRED:

1591 ?? <Query>

1592 ?? <SubjectQuery>

1593 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1594 ?? <Request>

1595 ?? <AuthenticationQuery>
1596 ?? <AuthorizationDecisionQuery>
1597 ?? <AttributeQuery>
1598 ?? <Response>

1599 6.3. Use of Type Derivation and Substitution Groups

1600 W3C XML Schema [**Schema1**] provides two principal mechanisms for specifying an element of an
1601 extended type: type derivation and substitution groups.

1602 For example, a <Statement> element can be assigned the type **NewStatementType** by means of
1603 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be
1604 derived from **StatementType**. The following example of a SAML assertion assumes that the
1605 extension schema (represented by the `new:` prefix) has defined this new type:

```
1606 <saml:Assertion ...>  
1607   <saml:Statement xsi:type="new:NewStatementType">  
1608     ...  
1609   </saml:Statement>  
1610 </saml:Assertion>
```

1611 Alternatively, the extension schema can define a <NewStatement> element that is a member of a
1612 substitution group that has <Statement> as a head element. For the substituted element to be
1613 schema-valid, it needs to have a type that matches or is derived from the head element's type. The
1614 following is an example of an extension schema fragment that defines this new element:

```
1615 <xsd:element "NewStatement" type="new:NewStatementType"  
1616   substitutionGroup="saml:Statement" />
```

1617 The substitution group declaration allows the <NewStatement> element to be used anywhere the
1618 SAML <Statement> element can be used. The following is an example of a SAML assertion that
1619 uses the extension element:

```
1620 <saml:Assertion ...>  
1621   <new:NewStatement>  
1622     ...  
1623   </new:NewStatement>  
1624 </saml:Assertion>
```

1625 The choice of extension method has no effect on the semantics of the XML document but does
1626 have implications for interoperability.

1627 The advantages of type derivation are as follows:

1628 ?? A document can be more fully interpreted by a parser that does not have access to the
1629 extension schema because a "native" SAML element is available.

1630 ?? At the time of writing, some W3C XML Schema validators do not support substitution
1631 groups, whereas the `xsi:type` attribute is widely supported.

1632 The advantage of substitution groups is that a document can be explained without the need to
1633 explain the functioning of the `xsi:type` attribute.

7. SAML-Defined Identifiers

1634

1635 The following sections define URI-based identifiers for common authentication protocols and
1636 actions.

1637 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the
1638 URN of the most current RFC that specifies the protocol is used. URI references created
1639 specifically for SAML have the initial stem:

1640 `urn:oasis:names:tc:SAML:1.0:`

7.1. Authentication Method and Confirmation Method Identifiers

1641

1642

1643 The <AuthenticationMethod> and <SubjectConfirmationMethod> elements perform
1644 different functions within the SAML architecture, although both can contain refer to the same
1645 underlying mechanisms~~some of the same values~~. <AuthenticationMethod> is a part of an
1646 Authentication Statement, which describes an authentication act which occurred in the past. The
1647 <AuthenticationMethod> indicates how that authentication was done. Note that the
1648 authentication statement does not provide the means to perform that authentication, such as a
1649 password, key or certificate.

1650 In contrast, <SubjectConfirmationMethod> is a part of the <SubjectConfirmation>, which
1651 is used to allow the Relying Party to confirm that the request or message came from the System
1652 Entity that corresponds to the Subject in the statement. The <SubjectConfirmationMethod>
1653 indicates the method ~~which that~~ the Relying Party can use to do this in the future. This may or may
1654 not have any relationship to an authentication that was performed previously. Unlike the
1655 Authentication Method, the <SubjectConfirmationMethod> will usually may be accompanied
1656 with some piece of information, such as a certificate or key, which will allow the Relying Party to
1657 perform the necessary check.

1658 Subject Confirmation Methods are defined in the SAML Profile or Profiles in which they are used
1659 [SAMLBind]. Additional methods may be added by defining new profiles or by private
1660 agreement. There are many <SubjectConfirmationMethod>, because there are many different
1661 SAML usage scenarios. A few examples are:

1662 The following identifiers refer to SAML specified Authentication methods.

1663

1664 ~~1. A user logs in with a password, but a temporary passcode or cookie is issued for confirmation~~
1665 ~~purposes to avoid repeated exposure of the long term password.~~

1666 ~~2. There is no login, but an application request is digitally signed. The associated public key is used~~
1667 ~~for confirmation.~~

1668 ~~3. The user logs in using Kerberos and a Kerberos ticket is used subsequently for confirmation.~~
1669 ~~Notice that in this case although both the Authentication Method and the~~
1670 ~~<SubjectConfirmationMethod> are Kerberos, what happens at each step is actually different.~~
1671 ~~(See [RFC 1510])~~

1672 ~~The following identifiers are defined to refer to common authentication protocols. Where Base64~~
1673 ~~encoding is specified the data is encoded as specified by [RFC 2045].~~

1674 ~~SAML Artifact (SHA-1):~~

1675 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:artifact-sha1~~

1676 ~~<SubjectConfirmationData>: Base64(SHA1(Artifact))~~
1677 ~~The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest~~
1678 ~~of the specified artifact matches the value specified in <SubjectConfirmationData>.~~

1679 **7.1.2. Holder of Key:**

1680 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:Holder-Of-Key~~

1681 ~~<ds:KeyInfo>: Any cryptographic key~~

1682 ~~The subject of the assertion is the party that can demonstrate that it is the holder of the private~~
1683 ~~component of the key specified in <ds:KeyInfo>.~~

1684 **7.1.3. Bearer Indication:**

1685 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:BearerIndication~~

1686 ~~The subject of the assertion is the bearer of the assertion.~~

1687 **7.1.4. Sender Vouches:**

1688 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:sender-vouches~~

1689 ~~Indicates that no other information is available about the context of use of the assertion. The~~
1690 ~~Relying party SHOULD utilize other means to determine if it should process the assertion further.~~

1691 **7.1.5.7.1.1. Password (Pass-Through):**

1692 ~~URI: urn:oasis:names:tc:SAML:1.0:aem:password~~

1693 ~~<SubjectConfirmationData>: Base64(Password)~~

1694 ~~The subject of the assertion is the party that can present the password value specified in~~
1695 ~~<SubjectConfirmationData>.~~

1696 ~~The username of the subject is specified by means of the <NameIdentifier> element. The~~
1697 ~~authentication was performed by means of a password.~~

1698 **7.1.6. Password (One-Way-Function SHA-1):**

1699 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:password-sha1~~

1700 ~~<SubjectConfirmationData>: Base64(SHA1(Password))~~

1701 ~~The subject of the assertion is the party that can present the password such that the SHA1 digest of~~
1702 ~~the specified password matches the value specified in <SubjectConfirmationData>.~~

1703 ~~The username of the subject is specified by means of the <NameIdentifier> element.~~

1704 **7.1.7.7.1.2. Kerberos**

1705 ~~URI: urn:ietf:rfc:1510~~

1706 ~~<SubjectConfirmationData>: A Kerberos Ticket~~

1707 ~~The subject is authenticated authentication was performed~~ by means of the Kerberos protocol [\[RFC](#)
1708 [1510\]](#)~~[RFC 1510]~~, an instantiation of the Needham-Schroeder symmetric key authentication
1709 mechanism [\[Needham78\]](#) ~~[Needham78]~~.

1710 **7.1.3. Secure Remote Password (SRP)**

1711 URI: urn:ietf:rfc:2945

1712 The authentication was performed by means of Secure Remote Password
1713 protocol as specified in [RFC 2945]

1714 **7.1.4. SSL/TLS Certificate Based Client Authentication:**

1715 URI: urn:ietf:rfc:2246

1716 The authentication was performed using either the SSL or TLS protocol with certificate based client
1717 authentication. TLS is described in [RFC 2246].

1718 **7.1.5. X.509 Public Key**

1719 URI: urn:oasis:names:tc:SAML:1.0:am:X509-PKI

1720 The authentication was performed by some (unspecified) mechanism on a key authenticated by
1721 means of an X.509 PKI [X.509][PKIX]. It may have been one of the mechanisms for which a more
1722 specific identifier has been defined below.

1723 **7.1.6. PGP Public Key**

1724 URI: urn:oasis:names:tc:SAML:1.0:am:PGP

1725 The authentication was performed by some (unspecified) mechanism on a key authenticated by
1726 means of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more
1727 specific identifier has been defined below.

1728 **7.1.7. SPKI Public Key**

1729 URI: urn:oasis:names:tc:SAML:1.0:am:SPKI

1730 The authentication was performed by some (unspecified) mechanism on a key authenticated by
1731 means of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific
1732 identifier has been defined below.

1733 **7.1.8. XKMS Public Key**

1734 URI: urn:oasis:names:tc:SAML:1.0:am:XKMS

1735 The authentication was performed by some (unspecified) mechanism on a key authenticated by
1736 means of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more
1737 specific identifier has been defined below.

1738 **7.1.8. SSL/TLS Certificate Based Client Authentication:**

1739 URI: urn:ietf:rfc:2246

1740 <ds:KeyInfo>: Any cryptographic key

1741 **7.1.9. Object Authenticator (SHA-1):**

1742 URI: urn:oasis:names:tc:SAML:1.0:cm:object-sha1

1743 <SubjectConfirmationData>: Base64 (SHA1 (Object))

1744 This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is
1745 used when the subject can be represented as a binary string, for example when it is an XML
1746 document or the disk image of executable code. Any preprocessing of the subject prior to
1747 computation of the digest is out of scope. The name of the subject should be conveyed in an
1748 accompanying NameIdentifier element.

1749 **7.1.10.PKCS#7**

1750 URI: urn:ietf:rfc:2315

1751 <SubjectConfirmationData>: Base64 (PKCS#7 (Object))

1752 This authenticator element is signed data in PKCS#7 format [PKCS#7]. The posited identity of the
1753 signer must be conveyed in an accompanying NameIdentifier element. This subject type may be
1754 included in the subject field of an authentication query, in which case the corresponding response
1755 indicates whether the posited signer is, indeed, the signer. It may be included in an attribute query,
1756 in which case, the requested attribute values for the subject authenticated by the signed data are
1757 returned. It may be included in an authorization query, in which case, the access request
1758 represented by the signed data shall be identified by the accompanying object element, and the
1759 corresponding assertion containing an authorization decision statement indicates whether the
1760 signer is authorized for the access request represented by the object element.

1761 **7.1.11.Cryptographic Message Syntax**

1762 URI: urn:ietf:rfc:2630

1763 <SubjectConfirmationData>: Base64 (CMS (Object))

1764 This authenticator element is signed data in CMS format [CMS]. See also 7.1.10

1765 **7.1.12.7.1.9. XML Digital Signature**

1766 URI: urn:ietf:rfc:3075

1767 <SubjectConfirmationData>: Base64 (XML-SIG (Object))

1768 <ds:KeyInfo>: A cryptographic signing key

1769 The authentication was performed by means of an XML digital signature [RFC 3075]. This
1770 authenticator element is signed data in XML Signature format. See also 7.1.10

1771 **7.2. Action Namespace Identifiers**

1772 The following identifiers MAY be used in the Namespace attribute of the <Action> element (see
1773 Section 2.4.4.1) to refer to common sets of actions to perform on resources.

1774 **7.2.1. Read/Write/Execute/Delete/Control:**

1775 URI: urn:oasis:names:tc:SAML:1.0:action:rwdc

1776 Defined actions:

1777 Read Write Execute Delete Control

1778 These actions are interpreted in the normal manner, i.e.

1779 Read

1780 The subject may read the resource

1781 Write
1782 The subject may modify the resource
1783 Execute
1784 The subject may execute the resource
1785 Delete
1786 The subject may delete the resource
1787 Control
1788 The subject may specify the access control policy for the resource

1789 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1790 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwdc-negation

1791 Defined actions:

1792 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1793 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions
1794 prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated
1795 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1796 affirmatively denied read permission.

1797 A SAML authority MUST NOT authorize both an action and its negated form.

1798 **7.2.3. Get/Head/Put/Post:**

1799 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

1800 Defined actions:

1801 GET HEAD PUT POST

1802 These actions bind to the corresponding HTTP operations. For example a subject authorized to
1803 perform the GET action on a resource is authorized to retrieve it.

1804 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT
1805 and POST actions to the write permission. The correspondence is not exact however since a HTTP
1806 GET operation may cause data to be modified and a POST operation may cause modification to a
1807 resource other than the one specified in the request. For this reason a separate Action URI
1808 reference specifier is provided.

1809 **7.2.4. UNIX File Permissions:**

1810 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1811 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)
1812 notation.

1813 The action string is a four digit numeric code:

1814 *extended user group world*

1815 Where the *extended* access permission has the value

1816 +2 if sgid is set

1817 +4 if suid is set

1818 The *user group* and *world* access permissions have the value

- 1819 +1 if execute permission is granted
- 1820 +2 if write permission is granted
- 1821 +4 if read permission is granted
- 1822 For example 0754 denotes the UNIX file access permission: user read, write and execute, group
- 1823 read and execute and world read.

8. SAML Schema Listings

1824

1825 The following sections contain complete listings of the assertion and protocol schemas for SAML.

8.1. Assertion Schema

1826

1827 Following is a complete listing of the SAML assertion schema [SAML-XSD].

```
1828 <?xml version="1.0" encoding="UTF-8"?>
1829 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1830 (VeriSign Inc.) -->
1831 <schema
1832     targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"
1833     xmlns="http://www.w3.org/2001/XMLSchema"
1834     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1835     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1836     elementFormDefault="unqualified">
1837     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1838           schemaLocation="xmldsig-core-schema.xsd"/>
1839     <annotation>
1840         <documentation>draft-sstc-schema-assertion-3130.xsd</documentation>
1841     </annotation>
1842     <simpleType name="IDType">
1843         <restriction base="string"/>
1844     </simpleType>
1845     <simpleType name="IDReferenceType">
1846         <restriction base="string"/>
1847     </simpleType>
1848     <simpleType name="DecisionType">
1849         <restriction base="string">
1850             <enumeration value="Permit"/>
1851             <enumeration value="Deny"/>
1852             <enumeration value="Indeterminate"/>
1853         </restriction>
1854     </simpleType>
1855     <element name="AssertionIDReference" type="saml:IDReferenceType"/>
1856     <element name="Assertion" type="saml:AssertionType"/>
1857     <complexType name="AssertionType">
1858         <sequence>
1859             <element ref="saml:Conditions" minOccurs="0"/>
1860             <element ref="saml:Advice" minOccurs="0"/>
1861             <choice maxOccurs="unbounded">
1862                 <element ref="saml:Statement"/>
1863                 <element ref="saml:SubjectStatement"/>
1864                 <element ref="saml:AuthenticationStatement"/>
1865                 <element ref="saml:AuthorizationDecisionStatement"/>
1866                 <element ref="saml:AttributeStatement"/>
1867             </choice>
1868             <element ref="ds:Signature" minOccurs="0"/>
1869         </sequence>
1870         <attribute name="MajorVersion" type="integer" use="required"/>
1871         <attribute name="MinorVersion" type="integer" use="required"/>
1872         <attribute name="AssertionID" type="saml:IDType" use="required"/>
1873         <attribute name="Issuer" type="string" use="required"/>
1874         <attribute name="IssueInstant" type="dateTime" use="required"/>
1875     </complexType>
1876     <element name="Conditions" type="saml:ConditionsType"/>
1877     <complexType name="ConditionsType">
1878         <choice minOccurs="0" maxOccurs="unbounded">
1879             <element ref="saml:AudienceRestrictionCondition"/>
1880             <element ref="saml:Condition"/>
```

```

1881     </choice>
1882     <attribute name="NotBefore" type="dateTime" use="optional"/>
1883     <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1884 </complexType>
1885 <element name="Condition" type="saml:ConditionAbstractType"/>
1886 <complexType name="ConditionAbstractType" abstract="true"/>
1887 <element name="AudienceRestrictionCondition"
1888     type="saml:AudienceRestrictionConditionType"/>
1889 <complexType name="AudienceRestrictionConditionType">
1890     <complexContent>
1891         <extension base="saml:ConditionAbstractType">
1892             <sequence>
1893                 <element ref="saml:Audience" maxOccurs="unbounded"/>
1894             </sequence>
1895         </extension>
1896     </complexContent>
1897 </complexType>
1898 <element name="Audience" type="anyURI"/>
1899 <element name="Advice" type="saml:AdviceType"/>
1900 <complexType name="AdviceType">
1901     <choice minOccurs="0" maxOccurs="unbounded">
1902         <element ref="saml:AssertionIDReference"/>
1903         <element ref="saml:Assertion"/>
1904         <any namespace="##other" processContents="lax"/>
1905     </choice>
1906 </complexType>
1907 <element name="Statement" type="saml:StatementAbstractType"/>
1908 <complexType name="StatementAbstractType" abstract="true"/>
1909 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1910 <complexType name="SubjectStatementAbstractType" abstract="true">
1911     <complexContent>
1912         <extension base="saml:StatementAbstractType">
1913             <sequence>
1914                 <element ref="saml:Subject"/>
1915             </sequence>
1916         </extension>
1917     </complexContent>
1918 </complexType>
1919 <element name="Subject" type="saml:SubjectType"/>
1920 <complexType name="SubjectType">
1921     <choice>
1922         <sequence>
1923             <element ref="saml:NameIdentifier"/>
1924             <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1925         </sequence>
1926         <element ref="saml:SubjectConfirmation"/>
1927     </choice>
1928 </complexType>
1929 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1930 <complexType name="NameIdentifierType">
1931     <simpleContent>
1932         <extension base="string">
1933             <attribute name="NameQualifier" type="string" use="optional"/>
1934             <attribute name="Format" type="anyURI" use="optional"/>
1935         </extension>
1936     </simpleContent>
1937 </complexType>
1938 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1939 <complexType name="SubjectConfirmationType">
1940     <sequence>
1941         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1942         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1943         <element ref="ds:KeyInfo" minOccurs="0"/>

```

```

1944     </sequence>
1945 </complexType>
1946 <element name="SubjectConfirmationData" type="stringanyType" />
1947 <element name="ConfirmationMethod" type="anyURI" />
1948 <element name="AuthenticationStatement"
1949     type="saml:AuthenticationStatementType" />
1950 <complexType name="AuthenticationStatementType" >
1951     <complexContent>
1952         <extension base="saml:SubjectStatementAbstractType" >
1953             <sequence>
1954                 <element ref="saml:SubjectLocality" minOccurs="0" />
1955                 <element ref="saml:AuthorityBinding"
1956                     minOccurs="0" maxOccurs="unbounded" />
1957             </sequence>
1958             <attribute name="AuthenticationMethod" type="anyURI" />
1959             <attribute name="AuthenticationInstant" type="dateTime" />
1960         </extension>
1961     </complexContent>
1962 </complexType>
1963 <element name="SubjectLocality"
1964     type="saml:SubjectLocalityType" />
1965 <complexType name="SubjectLocalityType" >
1966     <attribute name="IPAddress" type="string" use="optional" />
1967     <attribute name="DNSAddress" type="string" use="optional" />
1968 </complexType>
1969 <element name="AuthorityBinding" type="saml:AuthorityBindingType" />
1970 <complexType name="AuthorityBindingType" >
1971     <attribute name="AuthorityKind" type="QName" use="required" />
1972     <attribute name="Location" type="anyURI" use="required" />
1973     <attribute name="Binding" type="anyURI" use="required" />
1974 </complexType>
1975 <element name="AuthorizationDecisionStatement"
1976 type="saml:AuthorizationDecisionStatementType" />
1977 <complexType name="AuthorizationDecisionStatementType" >
1978     <complexContent>
1979         <extension base="saml:SubjectStatementAbstractType" >
1980             <sequence>
1981                 <element ref="saml:Action" maxOccurs="unbounded" />
1982                 <element ref="saml:Evidence" minOccurs="0" />
1983             </sequence>
1984             <attribute name="Resource" type="anyURI" use="required" />
1985             <attribute name="Decision" type="saml:DecisionType" use="required" />
1986         </extension>
1987     </complexContent>
1988 </complexType>
1989 <element name="Action" type="saml:ActionType" />
1990 <complexType name="ActionType" >
1991     <simpleContent>
1992         <extension base="string" >
1993             <attribute name="Namespace" type="anyURI" />
1994         </extension>
1995     </simpleContent>
1996 </complexType>
1997 <element name="Evidence" type="saml:EvidenceType" />
1998 <complexType name="EvidenceType" >
1999     <choice maxOccurs="unbounded" >
2000         <element ref="saml:AssertionIDReference" />
2001         <element ref="saml:Assertion" />
2002     </choice>
2003 </complexType>
2004 <element name="AttributeStatement" type="saml:AttributeStatementType" />
2005 <complexType name="AttributeStatementType" >
2006     <complexContent>

```

```

2007     <extension base="saml:SubjectStatementAbstractType">
2008         <sequence>
2009             <element ref="saml:Attribute" maxOccurs="unbounded"/>
2010         </sequence>
2011     </extension>
2012 </complexContent>
2013 </complexType>
2014 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
2015 <complexType name="AttributeDesignatorType">
2016     <attribute name="AttributeName" type="string" use="required"/>
2017     <attribute name="AttributeNamespace" type="anyURI" use="required"/>
2018 </complexType>
2019 <element name="Attribute" type="saml:AttributeType"/>
2020 <complexType name="AttributeType">
2021     <complexContent>
2022         <extension base="saml:AttributeDesignatorType">
2023             <sequence>
2024                 <element ref="saml:AttributeValue" maxOccurs="unbounded"/>
2025             </sequence>
2026         </extension>
2027     </complexContent>
2028 </complexType>
2029 <element name="AttributeValue" type="saml:anyType"/>
2030 </schema>

```

2031 8.2. Protocol Schema

2032 Following is a complete listing of the SAML protocol schema [SAML-XSD].

```

2033 <?xml version="1.0" encoding="UTF-8"?>
2034 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
2035 (VeriSign Inc.) -->
2036 <schema
2037     targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"
2038     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2039     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2040     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
2041     xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
2042     <import
2043         namespace="urn:oasis:names:tc:SAML:1.0:assertion"
2044         schemaLocation="draft-sstc-schema-assertion-3130.xsd"/>
2045     <import namespace="http://www.w3.org/2000/09/xmldsig#"
2046         schemaLocation="xmldsig-core-schema.xsd"/>
2047     <annotation>
2048         <documentation>draft-sstc-schema-protocol-3130.xsd</documentation>
2049     </annotation>
2050     <complexType name="RequestAbstractType" abstract="true">
2051         <sequence>
2052             <element ref="samlp:RespondWith"
2053                 minOccurs="0" maxOccurs="unbounded"/>
2054             <element ref="ds:Signature" minOccurs="0"/>
2055         </sequence>
2056         <attribute name="RequestID" type="saml:IDType" use="required"/>
2057         <attribute name="MajorVersion" type="integer" use="required"/>
2058         <attribute name="MinorVersion" type="integer" use="required"/>
2059         <attribute name="IssueInstant" type="dateTime" use="required"/>
2060     </complexType>
2061     <element name="RespondWith" type="QName"/>
2062     <element name="Request" type="samlp:RequestType"/>
2063     <complexType name="RequestType">
2064         <complexContent>
2065             <extension base="samlp:RequestAbstractType">
2066                 <choice>

```

```

2067         <element ref="sampl:Query" />
2068         <element ref="sampl:SubjectQuery" />
2069         <element ref="sampl:AuthenticationQuery" />
2070         <element ref="sampl:AttributeQuery" />
2071         <element ref="sampl:AuthorizationDecisionQuery" />
2072         <element ref="saml:AssertionID" maxOccurs="unbounded" />
2073         <element ref="sampl:AssertionArtifact" maxOccurs="unbounded" />
2074     </choice>
2075 </extension>
2076 </complexContent>
2077 </complexType>
2078 <element name="AssertionArtifact" type="string" />
2079 <element name="Query" type="sampl:QueryAbstractType" />
2080 <complexType name="QueryAbstractType" abstract="true" />
2081 <element name="SubjectQuery" type="sampl:SubjectQueryAbstractType" />
2082 <complexType name="SubjectQueryAbstractType" abstract="true">
2083     <complexContent>
2084         <extension base="sampl:QueryAbstractType">
2085             <sequence>
2086                 <element ref="saml:Subject" />
2087             </sequence>
2088         </extension>
2089     </complexContent>
2090 </complexType>
2091 <element name="AuthenticationQuery" type="sampl:AuthenticationQueryType" />
2092 <complexType name="AuthenticationQueryType">
2093     <complexContent>
2094         <extension base="sampl:SubjectQueryAbstractType">
2095             <attribute name="AuthenticationMethod" type="anyURI" />
2096         </extension>
2097     </complexContent>
2098 </complexType>
2099 <element name="AttributeQuery" type="sampl:AttributeQueryType" />
2100 <complexType name="AttributeQueryType">
2101     <complexContent>
2102         <extension base="sampl:SubjectQueryAbstractType">
2103             <sequence>
2104                 <element ref="saml:AttributeDesignator"
2105                     minOccurs="0" maxOccurs="unbounded" />
2106             </sequence>
2107             <attribute name="Resource" type="anyURI" use="optional" />
2108         </extension>
2109     </complexContent>
2110 </complexType>
2111 <element name="AuthorizationDecisionQuery"
2112     type="sampl:AuthorizationDecisionQueryType" />
2113 <complexType name="AuthorizationDecisionQueryType">
2114     <complexContent>
2115         <extension base="sampl:SubjectQueryAbstractType">
2116             <sequence>
2117                 <element ref="saml:Action" maxOccurs="unbounded" />
2118                 <element ref="saml:Evidence"
2119                     minOccurs="0" maxOccurs="unbounded" />
2120             </sequence>
2121             <attribute name="Resource" type="anyURI" use="required" />
2122         </extension>
2123     </complexContent>
2124 </complexType>
2125 <complexType name="ResponseAbstractType" abstract="true">
2126     <sequence>
2127         <element ref="ds:Signature" minOccurs="0" />
2128     </sequence>
2129     <attribute name="ResponseID" type="saml:IDType" use="required" />

```

```

2130     <attribute name="InResponseTo" type="saml:IDReferenceType"
2131         use="optional"/>
2132     <attribute name="MajorVersion" type="integer" use="required"/>
2133     <attribute name="MinorVersion" type="integer" use="required"/>
2134     <attribute name="IssueInstant" type="dateTime" use="required"/>
2135     <attribute name="Recipient" type="anyURI" use="optional"/>
2136 </complexType>
2137 <element name="Response" type="sampl:ResponseType"/>
2138 <complexType name="ResponseType">
2139     <complexContent>
2140         <extension base="sampl:ResponseAbstractType">
2141             <sequence>
2142                 <element ref="sampl:Status"/>
2143                 <element ref="saml:Assertion"
2144                     minOccurs="0" maxOccurs="unbounded"/>
2145             </sequence>
2146         </extension>
2147     </complexContent>
2148 </complexType>
2149 <element name="Status" type="sampl:StatusType"/>
2150 <complexType name="StatusType">
2151     <sequence>
2152         <element ref="sampl:StatusCode"/>
2153         <element ref="sampl:StatusMessage"
2154             minOccurs="0" maxOccurs="unbounded"/>
2155         <element ref="sampl:StatusDetail" minOccurs="0"/>
2156     </sequence>
2157 </complexType>
2158 <element name="StatusCode" type="sampl:StatusCodeType"/>
2159 <complexType name="StatusCodeType">
2160     <sequence>
2161         <element ref="sampl:StatusCode" minOccurs="0"/>
2162     </sequence>
2163     <attribute name="Value" type="QName" use="required"/>
2164 </complexType>
2165 <element name="StatusMessage" type="string"/>
2166 <element name="StatusDetail" type="sampl:StatusDetailType"/>
2167 <complexType name="StatusDetailType">
2168     <sequence>
2169         <any namespace="##any"
2170             processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2171     </sequence>
2172 </complexType>
2173 </schema>
2174

```

9. References

2175

- 2176 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March
2177 1984) Prentice Hall Computer Books;
- 2178 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks*
2179 *of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999,
2180 December 1978.
- 2181 [PGP] ~~Atkins, D., Stallings, W. and P. Zimmermann, *PGP Message Exchange*
2182 *Formats*, RFC 1991, August 1996.~~
- 2183 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also
2184 IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2185 [PKCS7] B. Kaliski., "*PKCS #7: Cryptographic Message Syntax, Version 1.5*", RFC
2186 2315, March 1998.
- 2187 [PKIX] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key*
2188 *Infrastructure Certificate and CRL Profile*. RFC 2459, January 1999.
- 2189 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
2190 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2191 [RFC 2045] ~~N. Freed, N. Borenstein. *Multipurpose Internet Mail Extensions (MIME)*
2192 *Part One: Format of Internet Message Bodies*
2193 <http://www.ietf.org/rfc/rfc2045.txt> IETF RFC 2045, November 1996.~~
- 2194 [RFC 2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,
2195 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2196 [RFC 2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
2197 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2198 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.
2199 <http://www.ietf.org/rfc/rfc2246.txt>
- 2200 [RFC 2396] T. Berners-Lee et. al., *Uniform Resource Identifiers (URI): Generic Syntax*
2201 <http://www.ietf.org/rfc/rfc2396.txt> IETF?
- 2202 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999.
2203 <http://www.ietf.org/rfc/rfc630.txt>
- 2204 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999.
2205 <http://www.ietf.org/rfc/rfc2648.txt>
- 2206 [RFC 2945] ~~T. Wu, *The SRP Authentication and Key Exchange System*, September
2207 2000, <http://www.ietf.org/rfc/rfc2945.txt>~~
- 2208 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*.
2209 March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2210 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion*
2211 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)
2212 [open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf),
2213 OASIS, December 2001.
- 2214 [SAMLConform] **TBS**
- 2215 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*
2216 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)
2217 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,
2218 December 2001.
- 2219 [SAMPL-XSD] P. Hallam-Baker et al., *SAML protocol schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd)
2220 [open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd),
2221 OASIS, December 2001.

2222	[SAMLSecure]	TBS
2223	[SAML-XSD]	P. Hallam-Baker et al., <i>SAML assertion schema</i> , http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd , OASIS, December 2001.
2224		
2225		
2226	[Schema1]	H. S. Thompson et al., <i>XML Schema Part 1: Structures</i> , http://www.w3.org/TR/xmlschema-1/ , World Wide Web Consortium Recommendation, May 2001.
2227		
2228		
2229	[Schema2]	P. V. Biron et al., <i>XML Schema Part 2: Datatypes</i> , http://www.w3.org/TR/xmlschema-2 , World Wide Web Consortium Recommendation, May 2001.
2230		
2231		
2232	[SPKI]	<u>C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i>, RFC 2693, September 1999.</u>
2233		
2234	[UNICODE-C]	M. Davis, M. J. Dürst, http://www.unicode.org/unicode/reports/tr15/tr15-21.html , UNICODE Consortium
2235		
2236	[W3C-CHAR]	M. J. Dürst, <i>Requirements for String Identity Matching and String Indexing</i> http://www.w3.org/TR/WD-charreq , World Wide Web Consortium.
2237		
2238	[W3C-CharMod]	M. J. Dürst,, <i>Unicode Normalization Forms</i> http://www.w3.org/TR/charmod/ , World Wide Web Consortium.
2239		
2240	[X.500]	ITU-T Recommendation X.501: <i>Information Technology - Open Systems Interconnection - The Directory: Models</i> , 1993.
2241		
2242	[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/
2243		
2244		
2245	[XML]	T. Bray et. al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> , http://www.w3.org/TR/REC-xml , World Wide Web Consortium.
2246		
2247	[XMLEnc]	<i>XML Encryption Specification</i> , In development.
2248	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , http://www.w3.org/TR/xmlsig-core/ , World Wide Web Consortium.
2249		
2250	[XMLSig-XSD]	XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd .
2251		
2252	[XTAML]	P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> , http://www.xmltrustcenter.org/ , VeriSign Inc. September 2001.
2253		
2254		

2255

10. Acknowledgements

2256 The editors would like to acknowledge the contributions of the OASIS SAML Technical Committee,
2257 whose voting members at the time of publication were:

- 2258 Allen Rogers, Authentica
- 2259 Irving Reid, Baltimore Technologies
- 2260 Krishna Sankar, Cisco Systems Inc
- 2261 Simon Godik, Crosslogix
- 2262 Gil Pilz, E2open
- 2263 Hal Lockhart, Entegriety Solutions
- 2264 Carlisle Adams, Entrust [Inc. Technologies](#)
- 2265 Robert Griffin, Entrust [Inc. Technologies](#)
- 2266 Don Flinn, Hitachi
- 2267 Joe Pato, Hewlett-Packard ([co-Chair](#))
- 2268 Jason Rouault, Hewlett-Packard
- 2269 Marc Chanliau, Netegrity
- 2270 Chris McLaren, Netegrity
- 2271 Prateek Mishra, Netegrity
- 2272 Charles Knouse, Oblix
- 2273 Steve Anderson, OpenNetwork
- 2274 Rob Philpott, RSA Security
- 2275 Jahan Moreh, Sigaba
- 2276 Bhavna Bhatnagar, Sun Microsystems
- 2277 Jeff Hodges, Sun Microsystems ([co-Chair](#))
- 2278 Eve Maler, Sun Microsystems ([Former Chair](#))
- 2279 Aravindan Ranganathan, Sun Microsystems
- 2280 Emily Xu, Sun Microsystems
- 2281 Bob Morgan, University of Washington
- 2282 Phillip Hallam-Baker, VeriSign Inc.

2283

2284 The editors would also like to thank the following people for their contributions:

- 2285 Stephen Farrell, Baltimore Technologies
- 2286 David Orchard, BEA [Systems](#)
- 2287 Tim Moses, Entrust [Inc.](#)
- 2288 Nigel Edwards, Hewlett-Packard
- 2289 Marc Chanliau, Netegrity
- 2290 Scott Cantor, The Ohio State University
- 2291 Darren Platt, Formerly with RSA Security
- 2292 Bob Blakeley [IBM Tivoli Software](#) ([Former Chair](#))
- 2293 Marlena Erdos, Tivoli

2294

2295

2296

2297

2298

2299

2300

2301

2302

2303

2304

2305

2306

2307
2308

2309

Appendix A. Notices

2310 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2311 that might be claimed to pertain to the implementation or use of the technology described in this
2312 document or the extent to which any license under such rights might or might not be available;
2313 neither does it represent that it has made any effort to identify any such rights. Information on
2314 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
2315 website. Copies of claims of rights made available for publication and any assurances of licenses to
2316 be made available, or the result of an attempt made to obtain a general license or permission for
2317 the use of such proprietary rights by implementors or users of this specification, can be obtained
2318 from the OASIS Executive Director.

2319 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2320 applications, or other proprietary rights which may cover technology that may be required to
2321 implement this specification. Please address the information to the OASIS Executive Director.

2322 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
2323 2001. All Rights Reserved.

2324 This document and translations of it may be copied and furnished to others, and derivative works
2325 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
2326 published and distributed, in whole or in part, without restriction of any kind, provided that the above
2327 copyright notice and this paragraph are included on all such copies and derivative works. However,
2328 this document itself may not be modified in any way, such as by removing the copyright notice or
2329 references to OASIS, except as needed for the purpose of developing OASIS specifications, in
2330 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights
2331 document must be followed, or as required to translate it into languages other than English.

2332 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2333 successors or assigns.

2334 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2335 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
2336 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
2337 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
2338 PARTICULAR PURPOSE.