



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

Last Call Working Draft 10, 2 May 2003

Document identifier:

sstc-saml-core-1.1-draft-10

Location:

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

Editors:

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

Contributors:

Stephen Farrell, Baltimore Technologies
Irving Reid, Baltimore Technologies
Hal Lockhart, BEA Systems (formerly with Entegriety)
David Orchard, BEA Systems
Krishna Sankar, Cisco Systems
Simon Godik, Crosslogix
Carlisle Adams, Entrust Inc.
Tim Moses, Entrust Inc.
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Marc Chanliau, Netegrity
Chris McLaren, Netegrity
Charles Knouse, Oblix
Scott Cantor, Ohio State University
Darren Platt, formerly with RSA Security
Jahan Moreh, Sigaba
Jeff Hodges, Sun Microsystems
Bob Blakley Tivoli
Marlena Erdos, Tivoli
RL "Bob" Morgan, University of Washington and Internet2
Phillip Hallam-Baker, VeriSign (former editor)

Abstract:

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

39 **Status:**

40 This document is a **last-call working draft** of the OASIS Security Services Technical Committee.
41 We solicit your comments; they must be received by Friday, 16 May 2003 in order for the
42 committee to consider them for inclusion in the Committee Specification.

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

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

Table of Contents

53	1	Introduction.....	6
54	1.1	Notation.....	6
55	1.2	Schema Organization and Namespaces.....	6
56	1.2.1	String and URI Values.....	7
57	1.2.2	Time Values.....	7
58	1.2.3	Comparing SAML Values.....	7
59	1.3	SAML Concepts (Non-Normative).....	7
60	1.3.1	Overview.....	7
61	1.3.2	SAML and URI-Based Identifiers.....	9
62	1.3.3	SAML and Extensibility.....	9
63	2	SAML Assertions.....	10
64	2.1	Schema Header and Namespace Declarations.....	10
65	2.2	Simple Types.....	10
66	2.2.1	Simple Types IDType and IDReferenceType.....	11
67	2.2.2	Simple Type DecisionType.....	11
68	2.3	Assertions.....	11
69	2.3.1	Element <AssertionIDReference>.....	12
70	2.3.2	Element <Assertion>.....	12
71	2.3.2.1	Element <Conditions>.....	13
72	2.3.2.1.1	Attributes NotBefore and NotOnOrAfter.....	14
73	2.3.2.1.2	Element <Condition>.....	14
74	2.3.2.1.3	Elements <AudienceRestrictionCondition> and <Audience>.....	14
75	2.3.2.1.4	Element <DoNotCacheCondition>.....	15
76	2.3.2.2	Element <Advice>.....	15
77	2.4	Statements.....	16
78	2.4.1	Element <Statement>.....	16
79	2.4.2	Element <SubjectStatement>.....	16
80	2.4.2.1	Element <Subject>.....	16
81	2.4.2.2	Element <NameIdentifier>.....	17
82	2.4.2.3	Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>.....	17
83	2.4.3	Element <AuthenticationStatement>.....	18
84	2.4.3.1	Element <SubjectLocality>.....	19
85	2.4.3.2	Element <AuthorityBinding>.....	19
86	2.4.4	Element <AttributeStatement>.....	20
87	2.4.4.1	Elements <AttributeDesignator> and <Attribute>.....	20
88	2.4.4.1.1	Element <AttributeValue>.....	21
89	2.4.5	Element <AuthorizationDecisionStatement>.....	21
90	2.4.5.1	Element <Action>.....	22
91	2.4.5.2	Element <Evidence>.....	23
92	3	SAML Protocol.....	24
93	3.1	Schema Header and Namespace Declarations.....	24
94	3.2	Requests.....	24

95	3.2.1 Complex Type RequestAbstractType	25
96	3.2.1.1 Element <RespondWith>	25
97	3.2.2 Element <Request>	26
98	3.2.2.1 Requests for Assertions by Reference	27
99	3.2.2.2 Element <AssertionArtifact>	27
100	3.3 Queries	27
101	3.3.1 Element <Query>	27
102	3.3.2 Element <SubjectQuery>	27
103	3.3.3 Element <AuthenticationQuery>	27
104	3.3.4 Element <AttributeQuery>	28
105	3.3.5 Element <AuthorizationDecisionQuery>	29
106	3.4 Responses	30
107	3.4.1 Complex Type ResponseAbstractType	30
108	3.4.2 Element <Response>	31
109	3.4.3 Element <Status>	31
110	3.4.3.1 Element <StatusCode>	32
111	3.4.3.2 Element <StatusMessage>	33
112	3.4.3.3 Element <StatusDetail>	33
113	3.4.4 Responses to Queries	33
114	4 SAML Versioning	34
115	4.1 SAML Specification Set Version	34
116	4.1.1 Schema Version	34
117	4.1.2 SAML Assertion Version	34
118	4.1.3 SAML Protocol Version	35
119	4.1.3.1 Request Version	35
120	4.1.4 Response Version	35
121	4.1.5 Permissible Version Combinations	35
122	4.2 SAML Namespace Version	36
123	4.2.1 Schema Evolution	36
124	5 SAML and XML Signature Syntax and Processing	37
125	5.1 Signing Assertions	37
126	5.2 Request/Response Signing	38
127	5.3 Signature Inheritance	38
128	5.4 XML Signature Profile	38
129	5.4.1 Signing Formats and Algorithms	38
130	5.4.2 References	38
131	5.4.3 Canonicalization Method	38
132	5.4.4 Transforms	39
133	5.4.5 KeyInfo	39
134	5.4.6 Binding Between Statements in a Multi-Statement Assertion	39
135	5.4.7 Interoperability with SAML V1.0	39
136	5.4.8 Example	39
137	6 SAML Extensions	42
138	6.1 Assertion Schema Extension	42
139	6.2 Protocol Schema Extension	42

140	6.3 Use of Type Derivation and Substitution Groups	43
141	7 SAML-Defined Identifiers	44
142	7.1 Authentication Method Identifiers	44
143	7.1.1 Password.....	44
144	7.1.2 Kerberos	44
145	7.1.3 Secure Remote Password (SRP).....	44
146	7.1.4 Hardware Token.....	45
147	7.1.5 SSL/TLS Certificate Based Client Authentication:	45
148	7.1.6 X.509 Public Key	45
149	7.1.7 PGP Public Key	45
150	7.1.8 SPKI Public Key	45
151	7.1.9 XKMS Public Key	45
152	7.1.10 XML Digital Signature.....	45
153	7.1.11 Unspecified.....	45
154	7.2 Action Namespace Identifiers	45
155	7.2.1 Read/Write/Execute/Delete/Control	46
156	7.2.2 Read/Write/Execute/Delete/Control with Negation	46
157	7.2.3 Get/Head/Put/Post	46
158	7.2.4 UNIX File Permissions	46
159	7.3 NameIdentifier Format Identifiers	47
160	7.3.1 Unspecified.....	47
161	7.3.2 Email Address	47
162	7.3.3 X.509 Subject Name	47
163	7.3.4 Windows Domain Qualified Name.....	47
164	8 References	48
165	Appendix A. Acknowledgments	50
166	Appendix B. Notices.....	51
167	Appendix C. Revision History.....	52
168		

169 1 Introduction

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

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

177 1.1 Notation

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

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

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

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

188 Listings of SAML schemas appear like this.

189 Example code listings appear like this.

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

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

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

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

203 1.2 Schema Organization and Namespaces

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

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

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

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

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

212 <http://www.w3.org/2000/09/xmlsig#>

213 See Section 4.2 for information on SAML namespace versioning.

214 1.2.1 String and URI Values

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

221 1.2.2 Time Values

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

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

226 1.2.3 Comparing SAML Values

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

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

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

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

247 1.3 SAML Concepts (Non-Normative)

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

250 1.3.1 Overview

251 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging security
252 information. This security information is expressed in the form of assertions about subjects, where a
253 subject is an entity (either human or computer) that has an identity in some security domain. A typical

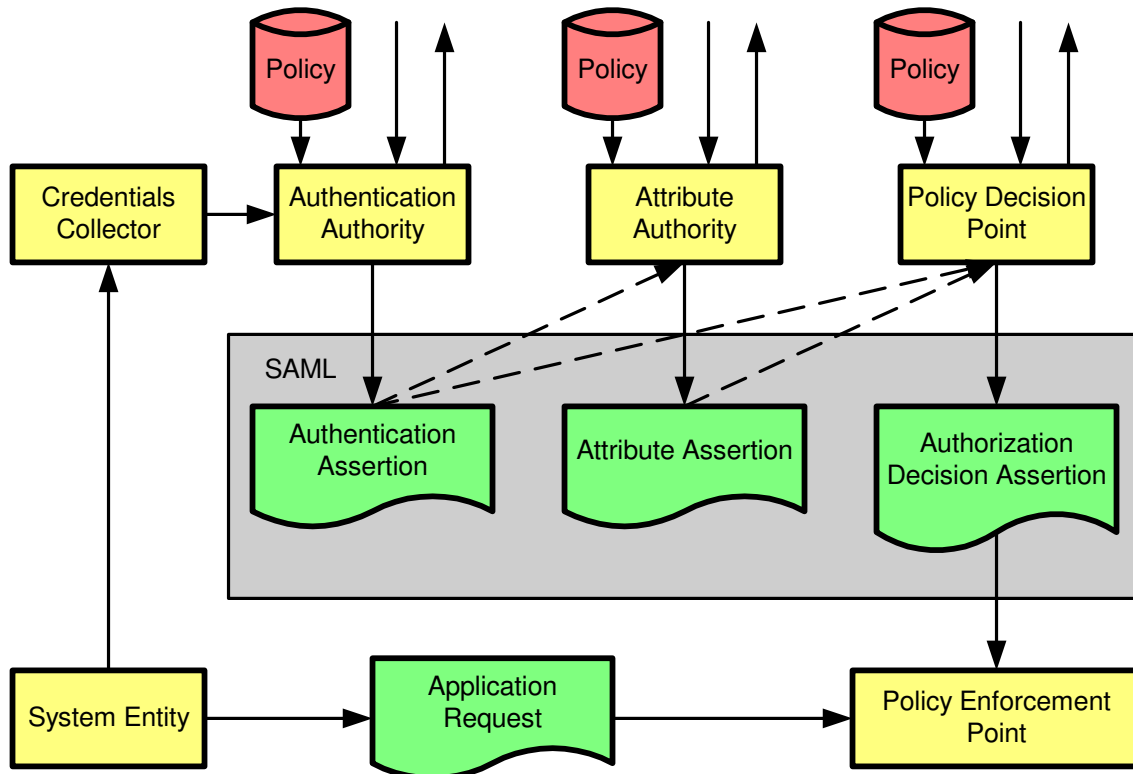
254 example of a subject is a person, identified by his or her email address in a particular Internet DNS
255 domain.

256 Assertions can convey information about authentication acts that were previously performed by subjects,
257 attributes of subjects, and authorization decisions about whether subjects are allowed to access certain
258 resources. Assertions are represented as XML constructs and have a nested structure, whereby a single
259 assertion might contain several different internal statements about authentication, authorization, and
260 attributes.

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

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

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



271
272

Figure 1 The SAML Domain Model

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

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

281 **1.3.2 SAML and URI-Based Identifiers**

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

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

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

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

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

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

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

299 **1.3.3 SAML and Extensibility**

300 The XML formats for SAML assertions and protocol messages have been designed to be extensible.
301 Section 6 describes SAML's design for extensibility in more detail.

302 However, it is possible that the use of extensions will harm interoperability and therefore the use of
303 extensions should be carefully considered.

304 2 SAML Assertions

305 An assertion is a package of information that supplies one or more statements made by a SAML
306 authority. While extensions are permitted, this SAML specification defines three different kinds of
307 assertion statement that can be created by a SAML authority:

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

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

315 2.1 Schema Header and Namespace Declarations

316 The following schema fragment defines the XML namespaces and other header information for the
317 assertion schema:

```
318 <schema  
319   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
320   xmlns="http://www.w3.org/2001/XMLSchema"  
321   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
322   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
323   elementFormDefault="unqualified"  
324   attributeFormDefault="unqualified"  
325   version="1.1">  
326   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
327     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
328 schema.xsd"/>  
329   <annotation>  
330     <documentation>  
331       Document identifier: sstc-saml-schema-assertion-1.1-draft-02  
332       Location: http://www.oasis-  
333 open.org/committees/documents.php?wg_abbrev=security  
334       Revision history:  
335       draft-01 (Eve Maler):  
336         Note that V1.1 of this schema has the same namespace  
337         as V1.0.  
338         Minor cosmetic updates.  
339         Changed IDType to restrict from xsd:ID.  
340         Changed IDReferenceType to restrict from xsd:IDREF.  
341         Set version attribute on schema element to 1.1.  
342       draft-02 (Prateek Mishra, Rob Philpott):  
343         Added DoNotCacheCondition element and  
344         DoNotCacheConditionType  
345     </documentation>  
346   </annotation>  
347   ...  
348 </schema>
```

349 2.2 Simple Types

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

351 2.2.1 Simple Types IDType and IDReferenceType

352 The **IDType** simple type is used to declare identifiers for assertions, requests, and responses, and is
353 based on the built-in XML Schema type **xsd:ID**. The **IDReferenceType** is used to reference identifiers of
354 type **IDType**, and is based on the built-in XML Schema type **xsd:IDREF**. Values declared to be of type
355 **IDType** MUST satisfy the following properties:

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

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

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

```
366 <simpleType name="IDType">  
367   <restriction base="ID"/>  
368 </simpleType>  
369 <simpleType name="IDReferenceType">  
370   <restriction base="IDREF"/>  
371 </simpleType>
```

372 2.2.2 Simple Type DecisionType

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

375 Permit

376 The specified action is permitted.

377 Deny

378 The specified action is denied.

379 Indeterminate

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

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

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

```
385 <simpleType name="DecisionType">  
386   <restriction base="string">  
387     <enumeration value="Permit"/>  
388     <enumeration value="Deny"/>  
389     <enumeration value="Indeterminate"/>  
390   </restriction>  
391 </simpleType>
```

392 2.3 Assertions

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

394 **2.3.1 Element <AssertionIDReference>**

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

396 The following schema fragment defines the <AssertionIDReference> element:

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

398 **2.3.2 Element <Assertion>**

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

401 MajorVersion [Required]

402 The major version of this assertion. The identifier for the version of SAML defined in this specification
403 is 1. SAML versioning is discussed in Section 4.

404 MinorVersion [Required]

405 The minor version of this assertion. The identifier for the version of SAML defined in this specification
406 is 1. SAML versioning is discussed in Section 4.

407 AssertionID [Required]

408 The identifier for this assertion. It is of type **IDType**, and MUST follow the requirements specified by
409 that type for identifier uniqueness.

410 Issuer [Required]

411 The SAML authority that created the assertion. The name of the issuer is provided as a string. The
412 issuer name SHOULD be unambiguous to the intended relying parties. SAML authorities may use an
413 identifier such as a URI reference that is designed to be unambiguous regardless of context.

414 IssueInstant [Required]

415 The time instant of issue in UTC, as described in Section 1.2.2.

416 <Conditions> [Optional]

417 Conditions that MUST be taken into account in assessing the validity of the assertion.

418 <Advice> [Optional]

419 Additional information related to the assertion that assists processing in certain situations but which
420 MAY be ignored by applications that do not support its use.

421 <ds:Signature> [Optional]

422 An XML Signature that authenticates the assertion, as described in Section 5.

423 One or more of the following statement elements:

424 <Statement>

425 A statement defined in an extension schema.

426 <SubjectStatement>

427 A subject statement defined in an extension schema.

428 <AuthenticationStatement>

429 An authentication statement.

430 <AuthorizationDecisionStatement>

431 An authorization decision statement.

432 <AttributeStatement>

433 An attribute statement.

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

```

435 <element name="Assertion" type="saml:AssertionType"/>
436 <complexType name="AssertionType">
437   <sequence>
438     <element ref="saml:Conditions" minOccurs="0"/>
439     <element ref="saml:Advice" minOccurs="0"/>
440     <choice maxOccurs="unbounded">
441       <element ref="saml:Statement"/>
442       <element ref="saml:SubjectStatement"/>
443       <element ref="saml:AuthenticationStatement"/>
444       <element ref="saml:AuthorizationDecisionStatement"/>
445       <element ref="saml:AttributeStatement"/>
446     </choice>
447     <element ref="ds:Signature" minOccurs="0"/>
448   </sequence>
449   <attribute name="MajorVersion" type="integer" use="required"/>
450   <attribute name="MinorVersion" type="integer" use="required"/>
451   <attribute name="AssertionID" type="saml:IDType" use="required"/>
452   <attribute name="Issuer" type="string" use="required"/>
453   <attribute name="IssueInstant" type="dateTime" use="required"/>
454 </complexType>

```

455 2.3.2.1 Element <Conditions>

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

457 NotBefore [Optional]

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

460 NotOnOrAfter [Optional]

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

463 <Condition> [Any Number]

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

465 <AudienceRestrictionCondition> [Any Number]

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

467 <DoNotCacheCondition> [Any Number]

468 Specifies that the assertion SHOULD be used immediately and MUST not be retained for future use.

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

```

471 <element name="Conditions" type="saml:ConditionsType"/>
472 <complexType name="ConditionsType">
473   <choice minOccurs="0" maxOccurs="unbounded">
474     <element ref="saml:AudienceRestrictionCondition"/>
475     <element ref="saml:DoNotCacheCondition"/>
476     <element ref="saml:Condition"/>
477   </choice>
478   <attribute name="NotBefore" type="dateTime" use="optional"/>
479   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
480 </complexType>

```

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

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

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

497 Note that an assertion that has validity status **Valid** may not be trustworthy by reasons such as not being
498 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

499 2.3.2.1.1 Attributes NotBefore and NotOnOrAfter

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

501 The NotBefore attribute specifies the time instant at which the validity interval begins. The
502 NotOnOrAfter attribute specifies the time instant at which the validity interval has ended.

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

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

514 2.3.2.1.2 Element <Condition>

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

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

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

521 2.3.2.1.3 Elements <AudienceRestrictionCondition> and <Audience>

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

527 <Audience>

528 A URI reference that identifies an intended audience. The URI reference MAY identify a document

529 that describes the terms and conditions of audience membership.
530 The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of
531 one or more of the audiences specified.

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

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

```
540 <element name="AudienceRestrictionCondition"  
541   type="saml:AudienceRestrictionConditionType"/>  
542 <complexType name="AudienceRestrictionConditionType">  
543   <complexContent>  
544     <extension base="saml:ConditionAbstractType">  
545       <sequence>  
546         <element ref="saml:Audience" maxOccurs="unbounded"/>  
547       </sequence>  
548     </extension>  
549   </complexContent>  
550 </complexType>  
551 <element name="Audience" type="anyURI"/>
```

552 2.3.2.1.4 Element <DoNotCacheCondition>

553 Indicates that the assertion SHOULD be used immediately by the relying party and MUST not be retained
554 for future use. Note: no implementation is required to perform caching, however any that do so MUST
555 observe this condition.

```
556  
557 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType" />  
558 <complexType name="DoNotCacheConditionType">  
559   <complexContent>  
560     <extension base="saml:ConditionAbstractType"/>  
561   </complexContent>  
562 </complexType>
```

563 2.3.2.2 Element <Advice>

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

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

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

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

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

```
576 <element name="Advice" type="saml:AdviceType"/>
```

```

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

```

584 2.4 Statements

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

586 2.4.1 Element <Statement>

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

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

```

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

```

594 2.4.2 Element <SubjectStatement>

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

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

```

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

```

611 2.4.2.1 Element <Subject>

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

614 <NameIdentifier>

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

616 <SubjectConfirmation>

617 Information that allows the subject to be authenticated.

618 If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the
619 SAML authority is asserting that if the SAML relying party performs the specified
620 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying

621 party is the entity that the SAML authority associates with the `<NameIdentifier>`. A `<Subject>`
622 element SHOULD NOT identify more than one principal.

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

```
624 <element name="Subject" type="saml:SubjectType"/>
625 <complexType name="SubjectType">
626   <choice>
627     <sequence>
628       <element ref="saml:NameIdentifier"/>
629       <element ref="saml:SubjectConfirmation" minOccurs="0"/>
630     </sequence>
631     <element ref="saml:SubjectConfirmation"/>
632   </choice>
633 </complexType>
```

634 2.4.2.2 Element `<NameIdentifier>`

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

638 `NameQualifier` [Optional]

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

641 `Format` [Optional]

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

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

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

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

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

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

666 `<ConfirmationMethod>` [One or more]

667 A URI reference that identifies a protocol to be used to authenticate the subject. URI references
668 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the

669 SAML bindings and profiles specification [**SAMLBind**]. Additional methods may be added by defining
670 new profiles or by private agreement.

671 <SubjectConfirmationData> [Optional]

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

673 <ds:KeyInfo> [Optional]

674 An XML Signature [**XMLSig**] element that provides access to a cryptographic key held by the subject.

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

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

688 2.4.3 Element <AuthenticationStatement>

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

693 AuthenticationMethod [Required]

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

696 AuthenticationInstant [Required]

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

699 <SubjectLocality> [Optional]

700 Specifies the DNS domain name and IP address for the system entity from which the subject was
701 apparently authenticated.

702 <AuthorityBinding> [Any Number]

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

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

```
706 <element name="AuthenticationStatement"
707         type="saml:AuthenticationStatementType"/>
708 <complexType name="AuthenticationStatementType">
709   <complexContent>
710     <extension base="saml:SubjectStatementAbstractType">
711       <sequence>
712         <element ref="saml:SubjectLocality" minOccurs="0"/>
713         <element ref="saml:AuthorityBinding"
714                 minOccurs="0" maxOccurs="unbounded"/>
715       </sequence>
716       <attribute name="AuthenticationMethod" type="anyURI"
717                 use="required"/>

```

718
719
720
721
722

```
                <attribute name="AuthenticationInstant" type="dateTime"
use="required"/>
            </extension>
        </complexContent>
    </complexType>
```

723 2.4.3.1 Element <SubjectLocality>

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

726 IPAddress [Optional]

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

728 DNSAddress [Optional]

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

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

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

734
735
736
737
738
739

```
<element name="SubjectLocality"
type="saml: SubjectLocalityType"/>
<complexType name="SubjectLocalityType">
  <attribute name="IPAddress" type="string" use="optional"/>
  <attribute name="DNSAddress" type="string" use="optional"/>
</complexType>
```

740 2.4.3.2 Element <AuthorityBinding>

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

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

748 The <AuthorityBinding> element has the following attributes:

749 AuthorityKind [Required]

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

755 For example, an attribute authority would be identified by

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

758 Location [Required]

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

762 Binding [Required]

763 A URI reference identifying the SAML protocol binding to use in communicating with the authority. All

764 SAML protocol bindings will have an assigned URI reference.

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

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

773 2.4.4 Element <AttributeStatement>

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

777 <Attribute> [One or More]

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

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

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

791 2.4.4.1 Elements <AttributeDesignator> and <Attribute>

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

796 AttributeNamespace [Required]

797 The namespace in which the AttributeName elements are interpreted.

798 AttributeName [Required]

799 The name of the attribute.

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

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

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

810 <AttributeValue> [Any Number]

811 The value of the attribute.

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

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

824 2.4.4.1.1 Element <AttributeValue>

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

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

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

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

833 2.4.5 Element <AuthorizationDecisionStatement>

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

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

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

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

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

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

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

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

864 `Resource` [Required]

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

868 `Decision` [Required]

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

871 `<Action>` [One or more]

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

873 `<Evidence>` [Optional]

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

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

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

892 2.4.5.1 Element `<Action>`

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

895 `Namespace` [Optional]

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

899 `string data` [Required]

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

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

```
902 <element name="Action" type="saml:ActionType"/>
```

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

910 **2.4.5.2 Element <Evidence>**

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

914 <AssertionIDReference>

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

916 <Assertion>

917 Specifies an assertion by value.

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

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

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

3 SAML Protocol

931

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

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



938

939

Figure 2: SAML Request-Response Protocol

3.1 Schema Header and Namespace Declarations

940

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

```
943 <schema  
944   targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"  
945   xmlns="http://www.w3.org/2001/XMLSchema"  
946   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
947   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
948   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"   
949   elementFormDefault="unqualified"  
950   attributeFormDefault="unqualified"  
951   version="1.1">  
952   <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"  
953     schemaLocation=" sstc-saml-schema-assertion-1.1-draft-02.xsd"/>  
954   <import namespace="http://www.w3.org/2000/09/xmldsig#"   
955     schemaLocation=" http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
956   schema.xsd "/>  
957   <annotation>  
958     <documentation>  
959       Document identifier: sstc-saml-schema-protocol-1.1-draft-03  
960       Location: http://www.oasis-  
961   open.org/committees/document.php?wg_abbrev=security  
962       Revision history:  
963         draft-01 (Eve Maler):  
964           Note that V1.1 of this schema has the same namespace  
965           as V1.0.  
966           Minor cosmetic updates.  
967           Set version attribute on schema element to 1.1.  
968         draft-02 (Eve Maler):  
969           Fix document Identifier.  
970         draft-03 (Prateek Mishra, Rob Philpott):  
971           Added DoNotCacheCondition and Do NotCacheConditionType.  
972     </documentation>  
973   </annotation>  
974   ...  
975 </schema>
```

3.2 Requests

976

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

978 **3.2.1 Complex Type RequestAbstractType**

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

981 RequestID [Required]

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

985 MajorVersion [Required]

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

988 MinorVersion [Required]

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

991 IssueInstant [Required]

992 The time instant of issue of the request. The time value is encoded in UTC as described in Section
993 1.2.2.

994 <RespondWith> [Any Number]

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

996 <ds:Signature> [Optional]

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

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

```
999 <complexType name="RequestAbstractType" abstract="true">  
1000 <sequence>  
1001 <element ref="saml:RespondWith"  
1002 minOccurs="0" maxOccurs="unbounded"/>  
1003 <element ref="ds:Signature" minOccurs="0"/>  
1004 </sequence>  
1005 <attribute name="RequestID" type="saml:IDType" use="required"/>  
1006 <attribute name="MajorVersion" type="integer" use="required"/>  
1007 <attribute name="MinorVersion" type="integer" use="required"/>  
1008 <attribute name="IssueInstant" type="dateTime" use="required"/>  
1009 </complexType>
```

1010 **3.2.1.1 Element <RespondWith>**

1011 The <RespondWith> element specifies the type of statement the SAML relying party wants from the
1012 SAML authority. Multiple <RespondWith> elements MAY be included to indicate that the relying party
1013 will accept assertions containing any of the specified types. If no <RespondWith> element is given, the
1014 SAML authority MAY return assertions containing statements of any type.

1015 NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be
1016 removed in the next major version of SAML.

1017 If the <Request> element contains one or more <RespondWith> elements, the SAML authority MUST
1018 NOT respond with assertions containing statements of any type not specified in one of the
1019 <RespondWith> elements.

1020 Inability to find assertions that meet <RespondWith> criteria should be treated as identical to any other
1021 query for which no assertions are available. In both cases a status of success MUST be returned in the
1022 Response message, but no assertions will be included.

1023 The content of each <RespondWith> element is an XML QName. The <RespondWith> content is
1024 either the QName of the desired SAML statement element name or, in the case of an extension schema,
1025 it is the QName of the SAML **StatementAbstractType** complex type or some type that was derived from
1026 it. In the case of an extension schema, all statements of the specified type are requested.

1027 For example, a relying party that wishes to receive assertions containing only attribute statements would
1028 specify <RespondWith>saml:AttributeStatement</RespondWith>, where the prefix is bound to
1029 the SAML assertion namespace in a namespace declaration that is in the scope of this element.

1030 The following schema fragment defines the <RespondWith> element:

1031

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

1032 3.2.2 Element <Request>

1033 The <Request> element specifies a SAML request. It provides either a query or a request for a specific
1034 assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has the complex
1035 type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following
1036 elements:

1037 <Query>

1038 An extension point that allows extension schemas to define new types of query.

1039 <SubjectQuery>

1040 An extension point that allows extension schemas to define new types of query that specify a single
1041 SAML subject.

1042 <AuthenticationQuery>

1043 Makes a query for authentication information.

1044 <AttributeQuery>

1045 Makes a query for attribute information.

1046 <AuthorizationDecisionQuery>

1047 Makes a query for an authorization decision.

1048 <AssertionIDReference> [One or more]

1049 Requests an assertion by reference to the value of its `AssertionID` attribute.

1050 <AssertionArtifact> [One or more]

1051 Requests assertions by supplying an assertion artifact that represents it.

1052 The following schema fragment defines the <Request> element and its **RequestType** complex type:

1053

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

1070

1071 3.2.2.1 Requests for Assertions by Reference

1072 In the context of a <Request> element, the <saml:AssertionIDReference> element is used to
1073 request an assertion by means of its ID. See Section 2.3.1 for more information on this element.

1074 3.2.2.2 Element <AssertionArtifact>

1075 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
1076 assertion being requested. Its use is governed by the specific profile of SAML that is being used; see the
1077 SAML specification for bindings and profiles [SAMLBind] for more information on the use of assertion
1078 artifacts in profiles.

1079 The following schema fragment defines the <AssertionArtifact> element:

```
1080 <element name="AssertionArtifact" type="string"/>
```

1081 3.3 Queries

1082 The following sections define the SAML constructs that contain query information.

1083 3.3.1 Element <Query>

1084 The <Query> element is an extension point that allows new SAML queries to be defined. Its
1085 **QueryAbstractType** is abstract and is thus usable only as the base of a derived type.
1086 **QueryAbstractType** is the base type from which all SAML query elements are derived.

1087 The following schema fragment defines the <Query> element and its **QueryAbstractType** complex type:

```
1088 <element name="Query" type="saml:QueryAbstractType"/>  
1089 <complexType name="QueryAbstractType" abstract="true"/>
```

1090 3.3.2 Element <SubjectQuery>

1091 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a single
1092 SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and is thus usable only as the
1093 base of a derived type. **SubjectQueryAbstractType** adds the <Subject> element.

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

```
1096 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>  
1097 <complexType name="SubjectQueryAbstractType" abstract="true">  
1098   <complexContent>  
1099     <extension base="saml:QueryAbstractType">  
1100       <sequence>  
1101         <element ref="saml:Subject"/>  
1102       </sequence>  
1103     </extension>  
1104   </complexContent>  
1105 </complexType>
```

1106 3.3.3 Element <AuthenticationQuery>

1107 The <AuthenticationQuery> element is used to make the query "What assertions containing
1108 authentication statements are available for this subject?" A successful response will be in the form of
1109 assertions containing authentication statements.

1110 The <AuthenticationQuery> element MUST NOT be used as a request for a new authentication
1111 using credentials provided in the request. <AuthenticationQuery> is a request for statements about

1112 authentication acts that have occurred in a previous interaction between the indicated subject and the
1113 Authentication Authority.

1114 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the
1115 addition of the following element:

1116 <AuthenticationMethod> [Optional]

1117 A filter for possible responses. If it is present, the query made is “What assertions containing
1118 authentication statements do you have for this subject with the supplied authentication method?”

1119 In response to an authentication query, a SAML authority returns assertions with authentication
1120 statements as follows:

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

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

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

1139 3.3.4 Element <AttributeQuery>

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

1144 Resource [Optional]

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

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

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

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

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

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

1169

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

```

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

```

1184 3.3.5 Element <AuthorizationDecisionQuery>

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

1190 Resource [Required]

1191 A URI reference indicating the resource for which authorization is requested.

1192 <Action> [One or More]

1193 The actions for which authorization is requested.

1194 <Evidence> [Optional]

1195 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1196 In response to an authorization decision query, a SAML authority returns assertions with authorization
1197 decision statements as follows:

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

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

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

3.4 Responses

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

3.4.1 Complex Type ResponseAbstractType

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

ResponseID [Required]

An identifier for the response. It is of type **IDType**, and MUST follow the requirements specified by that type for identifier uniqueness.

InResponseTo [Optional]

A reference to the identifier of the request to which the response corresponds, if any. If the response is not generated in response to a request, or if the `RequestID` attribute value of a request cannot be determined (because the request is malformed), then this attribute MUST NOT be present. Otherwise, it MUST be present and its value MUST match the value of the corresponding `RequestID` attribute value.

MajorVersion [Required]

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

MinorVersion [Required]

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

IssueInstant [Required]

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

Recipient [Optional]

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

<ds:Signature> [Optional]

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

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

```
<complexType name="ResponseAbstractType" abstract="true">
```

```

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

```

1261 3.4.2 Element <Response>

1262 The <Response> element specifies the status of the corresponding SAML request and a list of zero or
1263 more assertions that answer the request. It has the complex type **ResponseType**, which extends
1264 **ResponseAbstractType** by adding the following elements in order:

1265 <Status> [Required]

1266 A code representing the status of the corresponding request.

1267 <Assertion> [Any Number]

1268 Specifies an assertion by value. (See Section 2.3.2 for more information.)

1269 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

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

```

1282 3.4.3 Element <Status>

1283 The <Status> element contains the following elements:

1284 <StatusCode> [Required]

1285 A code representing the status of the corresponding request.

1286 <StatusMessage> [Optional]

1287 A message which MAY be returned to an operator.

1288 <StatusDetail> [Optional]

1289 Additional information concerning an error condition.

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

```

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

```

1299 **3.4.3.1 Element <StatusCode>**

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

1302 Value [Required]

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

1306 <StatusCode> [Optional]

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

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

1310 Success

1311 The request succeeded.

1312 VersionMismatch

1313 The SAML responder could not process the request because the version of the request message was
1314 incorrect.

1315 Requester

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

1317 Responder

1318 The request could not be performed due to an error on the part of the SAML responder or SAML
1319 authority.

1320 The following second-level status codes are referenced at various places in the specification. Additional
1321 second-level status codes MAY be defined in future versions of the SAML specification.

1322 RequestVersionTooHigh

1323 The SAML responder cannot process the request because the protocol version specified in the
1324 request message is a major upgrade from the highest protocol version supported by the responder.

1325 RequestVersionTooLow

1326 The SAML responder cannot process the request because the protocol version specified in the
1327 request message is too low.

1328 RequestVersionDeprecated

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

1330 TooManyResponses

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

1332 RequestDenied

1333 The SAML responder or SAML authority is able to process the request but has chosen not to
1334 respond. This status code MAY be used when there is concern about the security context of the
1335 request message or the sequence of request messages received from a particular requester.

1336 ResourceNotRecognized

1337 The SAML authority does not wish to support resource-specific attribute queries, or the resource
1338 value provided in the request message is invalid or unrecognized.

1339 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT
1340 define additional codes in the SAML assertion or protocol namespace.

1341 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value
1342 attribute and have the above semantics only in that context.

1343 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
1344 type:

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

1352 3.4.3.2 Element <StatusMessage>

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

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

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

1357 3.4.3.3 Element <StatusDetail>

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

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

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

1369 3.4.4 Responses to Queries

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

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

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

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

1383 4 SAML Versioning

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

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

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

1392 4.1 SAML Specification Set Version

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

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

1405 4.1.1 Schema Version

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

1411 4.1.2 SAML Assertion Version

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

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

1418 The following processing rules apply:

- 1419 • A SAML authority MUST NOT issue any assertion whose version number is not supported.
- 1420 • A SAML relying party MUST reject any assertion whose major version number is not supported.
- 1421 • A SAML relying party MAY reject any assertion whose minor version number is higher than the
1422 highest supported version that it supports. However, all assertions that share a major version number
1423 MUST share the same general processing rules and semantics, and MAY be treated in a uniform way
1424 by an implementation. That is, if a V1.1 assertion shares the syntax of a V1.0 assertion, an
1425 implementation MAY treat the assertion as a V1.0 assertion without ill effect.

1426 **4.1.3 SAML Protocol Version**

1427 The SAML protocol `<Request>` and `<Response>` elements contain attributes for expressing the major
1428 and minor version of the request or response message using a pair of integers. Each version of the SAML
1429 specification set will be construed so as to document the syntax, semantics, and processing rules of the
1430 protocol messages of the same version. That is, specification set version 1.0 describes request and
1431 response version V1.0, and so on.

1432 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace
1433 that contains the schema definitions for protocol messages for that protocol version.

1434 The version numbers used in SAML protocol `<Request>` and `<Response>` elements will be the same
1435 for any particular revision of the SAML specification set.

1436 **4.1.3.1 Request Version**

1437 The following processing rules apply to requests:

- 1438 • A SAML requester SHOULD issue requests with the highest SAML version supported by both the
1439 SAML requester and the SAML responder.
- 1440 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume
1441 that it supports requests with the highest SAML version supported by the requester.
- 1442 • A SAML requester MUST NOT issue a request with a version number matching a response version
1443 number that it cannot support.
- 1444 • A SAML responder MUST reject any request whose major version number is not supported.
- 1445 • A SAML responder MAY reject any request whose minor version number is higher than the highest
1446 supported request version that it supports. However, all requests that share a major version number
1447 MUST share the same general processing rules and semantics, and MAY be treated in a uniform way
1448 by an implementation. That is, if a V1.1 request shares the syntax of a V1.0 request, a responder
1449 MAY treat the message as a V1.0 request without ill effect.

1450 **4.1.4 Response Version**

1451 The following processing rules apply to responses:

- 1452 • A SAML responder MUST NOT issue response messages that have a higher SAML version number
1453 than the corresponding request message.
- 1454 • A SAML responder MUST NOT issue a response message that has a major version number that is
1455 lower than the major version number of the corresponding request message except to report the error
1456 `RequestVersionTooHigh`.

1457 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-
1458 level `<StatusCode>` value of `VersionMismatch`, and MAY result in reporting one of the following
1459 second-level values: `RequestVersionTooHigh`, `RequestVersionTooLow`, or
1460 `RequestVersionDeprecated`.

1461 **4.1.5 Permissible Version Combinations**

1462 In general, assertions of a particular major version may appear in response messages of the same major
1463 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol
1464 schema. Future versions of this specification are expected to explicitly describe the permitted
1465 combinations across major versions.

1466 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to
1467 appear in a V1.1 response message.

1468 **4.2 SAML Namespace Version**

1469 XML schema instances and "qualified names" (QNames) published as part of the specification set contain
1470 one or more target namespaces into which the type, element, and attribute definitions are placed. Each
1471 namespace is distinct from the others, and represents, in shorthand, the structural and syntactical
1472 definitions that make up that part of the specification.

1473 The namespace URIs defined by the specification set will generally contain version information of the
1474 form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond to
1475 the major and minor version of the specification set in which the namespace is first introduced and
1476 defined. This information is not typically consumed by an XML processor, which treats the namespace
1477 opaquely, but is intended to communicate the relationship between the specification set and the
1478 namespaces it defines.

1479 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
1480 defined by a major revision of the specification set to remain valid and stable across minor revisions of
1481 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,
1482 but this is expected to be rare. In such cases, the older namespaces and their associated definitions
1483 should be expected to remain valid until a major specification set revision.

1484 **4.2.1 Schema Evolution**

1485 In general, maintaining namespace stability while adding or changing the content of a schema are
1486 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
1487 older implementations will react to any given change, making forward compatibility difficult to achieve.
1488 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace
1489 stability. Except in special circumstances (for example to correct major deficiencies or fix errors),
1490 implementations should expect forward compatible schema changes in minor revisions, allowing new
1491 messages to validate against older schemas.

1492 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in
1493 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types
1494 that leverage the extension facilities described in Section 6. Older implementations **SHOULD** reject such
1495 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples
1496 include new query, statement, or condition types.

1497

5 SAML and XML Signature Syntax and Processing

1498 SAML assertions and SAML protocol request and response messages may be signed, with the following
1499 benefits:

- 1500 • An assertion signed by the SAML authority supports:
 - 1501 – Assertion integrity.
 - 1502 – Authentication of the SAML authority to a SAML relying party.
 - 1503 – If the signature is based on the SAML authority's public-private key pair, then it also provides for
1504 non-repudiation of origin.
- 1505 • A SAML protocol request or response message signed by the message originator supports:
 - 1506 – Message integrity.
 - 1507 – Authentication of message origin to a destination.
 - 1508 – If the signature is based on the originator's public-private key pair, then it also provides for non-
1509 repudiation of origin.

1510 A digital signature is not always required in SAML. For example, it may not be required in the following
1511 situations:

- 1512 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains
1513 protection from a signature on the containing protocol response message. "Inherited" signatures
1514 should be used with care when the contained object (such as the assertion) is intended to have a
1515 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,
1516 exposing the message contents and adding potentially unnecessary overhead.
- 1517 • The SAML relying party or SAML requester may have obtained an assertion or protocol message
1518 from the SAML authority or SAML responder directly (with no intermediaries) through a secure
1519 channel, with the SAML authority or SAML responder having authenticated to the relying party or
1520 SAML responder by some means other than a digital signature.

1521 Many different techniques are available for "direct" authentication and secure channel establishment
1522 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1523 the applicable security requirements depend on the communicating applications and the nature of the
1524 assertion or message transported.

1525 It is recommended that, in all other contexts, digital signatures be used for assertions and request and
1526 response messages. Specifically:

- 1527 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
1528 SHOULD be signed by the SAML authority.
- 1529 • A SAML protocol message arriving at a destination from an entity other than the originating site
1530 SHOULD be signed by the origin site.

1531 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that
1532 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures
1533 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure
1534 compatibility with profiles that may require other mechanisms.

1535 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST
1536 be used if signing.

1537 5.1 Signing Assertions

1538 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema
1539 as described in Section 2.3.

1540 5.2 Request/Response Signing

1541 All SAML protocol request and response messages MAY be signed using the XML Signature. This is
1542 reflected in the schema as described in Sections 3.2 and 3.4.

1543 5.3 Signature Inheritance

1544 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
1545 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a
1546 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
1547 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,
1548 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
1549 interpretation should be equivalent to the case where the assertion itself was signed with the same key
1550 and signature options.

1551 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
1552 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may
1553 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
1554 information from the surrounding context, but no such inheritance should be inferred unless specifically
1555 identified by the profile.

1556 5.4 XML Signature Profile

1557 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1558 and many choices. This section details the constraints on these facilities so that SAML processors do not
1559 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
1560 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply: the
1561 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the
1562 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section
1563 as the identifier attributes.

1564 5.4.1 Signing Formats and Algorithms

1565 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1566 detached.

1567 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
1568 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
1569 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

1570 5.4.2 References

1571 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
1572 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element
1573 may or may not be the root element of the actual XML document containing the signed assertion or
1574 message.

1575 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1576 value of the root element of the message being signed. For example, if the attribute value is "foo", then
1577 the `URI` attribute in the `<ds:Reference>` element MUST be "#foo".

1578 5.4.3 Canonicalization Method

1579 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,
1580 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a
1581 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over
1582 SAML messages embedded in an XML context can be verified independent of that context.

1583 5.4.4 Transforms

1584 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
1585 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
1586 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1587 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1588 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1589 not, verifiers MUST insure that no content of the SAML message is excluded from the signature. This can
1590 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
1591 applying the transforms manually to the content and reverifying the result as consisting of the same
1592 SAML message.

1593 5.4.5 KeyInfo

1594 SAML does not impose any restrictions in this area. Therefore, following XML Signature [XMLSig],
1595 `<ds:KeyInfo>` MAY be absent.

1596 5.4.6 Binding Between Statements in a Multi-Statement Assertion

1597 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section
1598 2.

1599 5.4.7 Interoperability with SAML V1.0

1600 The use of XML Signature [XMLSig] described above is incompatible with the usage described in the
1601 SAML V1.0 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to
1602 ensure interoperability. It was constrained by the inability to use URI references to identify the SAML
1603 content to be signed. With this limitation removed by the addition of SAML identifier attributes, a decision
1604 has been made to forgo backwards compatibility with the older specification in this respect.

1605 5.4.8 Example

1606 Following is an example of a signed response containing a signed assertion. Line breaks have been
1607 added for readability; the signatures are not valid and cannot be successfully verified.

```
1608 <Response  
1609   IssueInstant="2003-04-17T00:46:02Z"  
1610   MajorVersion="1"  
1611   MinorVersion="1"  
1612   Recipient="www.opensaml.org"  
1613   ResponseID="c7055387-af61-4fce-8b98-e2927324b306"  
1614   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
1615   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
1616   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1617   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
1618   <ds:Signature  
1619     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
1620     <ds:SignedInfo>  
1621     <ds:CanonicalizationMethod  
1622       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1623     <ds:SignatureMethod  
1624       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1625     <ds:Reference  
1626       URI="#c7055387-af61-4fce-8b98-e2927324b306">  
1627     <ds:Transforms>  
1628     <ds:Transform  
1629       Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />  
1630     <ds:Transform  
1631       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">  
1632     </InclusiveNamespaces
```

```

1633 PrefixList="#default saml samlp ds xsd xsi code kind rw signs"
1634 xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1635 </ds:Transform>
1636 </ds:Transforms>
1637 <ds:DigestMethod
1638 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1639 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
1640 </ds:Reference>
1641 </ds:SignedInfo>
1642 <ds:SignatureValue>
1643 x/GyPbzmFEE85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
1644 EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaK1ywS7gFgsD01qjyen3CP+m3D
1645 w6vKhaqledl0BYyrIzb4KkH04ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
1646 <ds:KeyInfo>
1647 <ds:X509Data>
1648 <ds:X509Certificate>
1649 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1650 MRlWEAyDVQIEWlXaXNjb25zaW4xEDAOBgNVBACTB01hZGlzb24xIDAeBgNVBAoT
1651 FlVuaXZlcnNpdHkgb2YgV2lzY29uc2luMSswKQYDVQLEyJEaXZpc2lvbiBvZiBJ
1652 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
1653 LS0gMjAwMjA3MDFBMB4XDTAyMdcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
1654 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbWJESMBAGAlUEBxMJQW5uIEFy
1655 Ym9yMQ4wDAYDVQQKEWVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJvZmVjLmVh
1656 dTEhMCUGCSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvvhAXnXVIVTx8vuRay+x50z7GJj
1657 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYiE7x4XAWIrcP+
1658 c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7027rhRjE
1659 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
1660 hkiG9w0BAQFFAAOBgQBfdqEW+OI3jzBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
1661 qgi7lFV6MDkxhTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
1662 8I3bsbmRAUg4UP9hH6ABVq4KQKMKnxulxQxLhpR1y1GpdiowMNTREg8cCx3w/w==
1663 </ds:X509Certificate>
1664 </ds:X509Data>
1665 </ds:KeyInfo>
1666 </ds:Signature>
1667 <Status><StatusCode Value="samlp:Success" /></Status>
1668 <Assertion
1669 AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"
1670 IssueInstant="2003-04-17T00:46:02Z"
1671 Issuer="www.opensaml.org"
1672 MajorVersion="1"
1673 MinorVersion="1"
1674 xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
1675 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1676 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
1677 <Conditions
1678 NotBefore="2003-04-17T00:46:02Z"
1679 NotOnOrAfter="2003-04-17T00:51:02Z">
1680 <AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
1681 </AudienceRestrictionCondition></Conditions>
1682 <AuthenticationStatement
1683 AuthenticationInstant="2003-04-17T00:46:00Z"
1684 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
1685 <Subject>
1686 <NameIdentifier
1687 Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
1688 scott@example.org</NameIdentifier>
1689 <SubjectConfirmation>
1690 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
1691 </SubjectConfirmation></Subject>
1692 <SubjectLocality
1693 IPAddress="127.0.0.1" />
1694 </AuthenticationStatement>

```


1746

6 SAML Extensions

1747 The SAML schemas support extensibility. An example of an application that extends SAML assertions is
1748 the Liberty Protocols and Schema Specification [**LibertyProt**]. The following sections explain how to use
1749 the extensibility features in SAML to create extension schemas.

1750 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML elements
1751 MAY serve as the head element of a substitution group. Also, types are not defined as *final*, so that all
1752 SAML types MAY be extended and restricted. The following sections discuss only elements that have
1753 been specifically designed to support extensibility.

6.1 Assertion Schema Extension

1754 The SAML assertion schema is designed to permit separate processing of the assertion package and the
1755 statements it contains, if the extension mechanism is used for either part.

1757 The following elements are intended specifically for use as extension points in an extension schema; their
1758 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 1759 • <Condition>
- 1760 • <Statement>
- 1761 • <SubjectStatement>

1762 The following elements that are directly usable as part of SAML MAY be extended:

- 1763 • <AuthenticationStatement>
- 1764 • <AuthorizationDecisionStatement>
- 1765 • <AttributeStatement>
- 1766 • <AudienceRestrictionCondition>

1767 The following elements are defined to allow elements from arbitrary namespaces within them, which
1768 serves as a built-in extension point without requiring an extension schema:

- 1769 • <AttributeValue>
- 1770 • <Advice>

6.2 Protocol Schema Extension

1772 The following SAML protocol elements are intended specifically for use as extension points in an
1773 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
1774 type:

- 1775 • <Query>
- 1776 • <SubjectQuery>

1777 The following elements that are directly usable as part of SAML MAY be extended:

- 1778 • <Request>
- 1779 • <AuthenticationQuery>
- 1780 • <AuthorizationDecisionQuery>
- 1781 • <AttributeQuery>
- 1782 • <Response>

1783 6.3 Use of Type Derivation and Substitution Groups

1784 W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of an
1785 extended type: type derivation and substitution groups.

1786 For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of the
1787 `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived
1788 from **StatementType**. The following example of a SAML assertion assumes that the extension schema
1789 (represented by the `new:` prefix) has defined this new type:

```
1790 <saml:Assertion ...>  
1791   <saml:Statement xsi:type="new:NewStatementType">  
1792     ...  
1793   </saml:Statement>  
1794 </saml:Assertion>
```

1795 Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a
1796 substitution group that has `<Statement>` as a head element. For the substituted element to be schema-
1797 valid, it needs to have a type that matches or is derived from the head element's type. The following is an
1798 example of an extension schema fragment that defines this new element:

```
1799 <xsd:element "NewStatement" type="new:NewStatementType"  
1800   substitutionGroup="saml:Statement"/>
```

1801 The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML
1802 `<Statement>` element can be used. The following is an example of a SAML assertion that uses the
1803 extension element:

```
1804 <saml:Assertion ...>  
1805   <new:NewStatement>  
1806     ...  
1807   </new:NewStatement>  
1808 </saml:Assertion>
```

1809 The choice of extension method has no effect on the semantics of the XML document but does have
1810 implications for interoperability.

1811 The advantages of type derivation are as follows:

- 1812 • A document can be more fully interpreted by a parser that does not have access to the extension
1813 schema because a “native” SAML element is available.
- 1814 • At the time of this writing, some W3C XML Schema validators do not support substitution groups,
1815 whereas the `xsi:type` attribute is widely supported.

1816 The advantage of substitution groups is that a document can be explained without the need to explain the
1817 functioning of the `xsi:type` attribute.

1818 7 SAML-Defined Identifiers

1819 The following sections define URI-based identifiers for common authentication methods, resource access
1820 actions, and subject name identifier formats.

1821 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of
1822 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
1823 have one of the following stems:

```
1824 urn:oasis:names:tc:SAML:1.0:  
1825 urn:oasis:names:tc:SAML:1.1:
```

1826 7.1 Authentication Method Identifiers

1827 The <AuthenticationMethod> and <SubjectConfirmationMethod> elements perform different
1828 functions, although both can refer to the same underlying mechanisms. <AuthenticationMethod> is a
1829 part of an authentication statement, which describes an authentication act that occurred in the past. The
1830 <AuthenticationMethod> element indicates how that authentication was done. Note that the
1831 authentication statement does not provide the means to perform that authentication, such as a password,
1832 key, or certificate.

1833 In contrast, <SubjectConfirmationMethod> is a part of the <SubjectConfirmation> element,
1834 which is an optional part of a SAML subject. <SubjectConfirmation> is used to allow the SAML
1835 relying party to confirm that the request or message came from a system entity that corresponds to the
1836 subject in the statement. The <SubjectConfirmationMethod> element indicates the method that the
1837 relying party can use to do this in the future. This may or may not have any relationship to an
1838 authentication that was performed previously. Unlike the authentication method, the subject confirmation
1839 method may be accompanied by some piece of information, such as a certificate or key, that will allow the
1840 relying party to perform the necessary check.

1841 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML
1842 bindings and profiles specification [**SAMLBind**] for more information. Additional methods may be added
1843 by defining new profiles or by private agreement.

1844 The following identifiers refer to SAML-specified authentication methods.

1845 7.1.1 Password

1846 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

1847 The authentication was performed by means of a password.

1848 7.1.2 Kerberos

1849 **URI:** urn:ietf:rfc:1510

1850 The authentication was performed by means of the Kerberos protocol [**RFC 1510**], an instantiation of the
1851 Needham-Schroeder symmetric key authentication mechanism [**Needham78**].

1852 7.1.3 Secure Remote Password (SRP)

1853 **URI:** urn:ietf:rfc:2945

1854 The authentication was performed by means of Secure Remote Password protocol as specified in [**RFC**
1855 **2945**].

1856 **7.1.4 Hardware Token**

1857 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

1858 The authentication was performed using some (unspecified) hardware token.

1859 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

1860 **URI:** urn:ietf:rfc:2246

1861 The authentication was performed using either the SSL or TLS protocol with certificate-based client
1862 authentication. TLS is described in **[RFC 2246]**.

1863 **7.1.6 X.509 Public Key**

1864 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

1865 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1866 an X.509 PKI **[X.500][PKIX]**. It may have been one of the mechanisms for which a more specific identifier
1867 has been defined below.

1868 **7.1.7 PGP Public Key**

1869 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

1870 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1871 a PGP web of trust **[PGP]**. It may have been one of the mechanisms for which a more specific identifier
1872 has been defined below.

1873 **7.1.8 SPKI Public Key**

1874 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

1875 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1876 a SPKI PKI **[SPKI]**. It may have been one of the mechanisms for which a more specific identifier has
1877 been defined below.

1878 **7.1.9 XKMS Public Key**

1879 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

1880 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of
1881 a XKMS trust service **[XKMS]**. It may have been one of the mechanisms for which a more specific
1882 identifier has been defined below.

1883 **7.1.10 XML Digital Signature**

1884 **URI:** urn:ietf:rfc:3075

1885 The authentication was performed by means of an XML digital signature **[RFC 3075]**.

1886 **7.1.11 Unspecified**

1887 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

1888 The authentication was performed by an unspecified means.

1889 **7.2 Action Namespace Identifiers**

1890 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see Section
1891 2.4.5.1) to refer to common sets of actions to perform on resources.

1892 **7.2.1 Read/Write/Execute/Delete/Control**

1893 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

1894 Defined actions:

1895 Read Write Execute Delete Control

1896 These actions are interpreted as follows:

1897 Read

1898 The subject may read the resource.

1899 Write

1900 The subject may modify the resource.

1901 Execute

1902 The subject may execute the resource.

1903 Delete

1904 The subject may delete the resource.

1905 Control

1906 The subject may specify the access control policy for the resource.

1907 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

1908 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

1909 Defined actions:

1910 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1911 The actions specified in Section 7.2.1 are interpreted in the same manner described there. Actions
1912 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated
1913 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1914 affirmatively denied read permission.

1915 A SAML authority MUST NOT authorize both an action and its negated form.

1916 **7.2.3 Get/Head/Put/Post**

1917 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

1918 Defined actions:

1919 GET HEAD PUT POST

1920 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
1921 the GET action on a resource is authorized to retrieve it.

1922 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
1923 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
1924 operation may cause data to be modified and a POST operation may cause modification to a resource
1925 other than the one specified in the request. For this reason a separate Action URI reference specifier is
1926 provided.

1927 **7.2.4 UNIX File Permissions**

1928 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1929 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

1930 The action string is a four-digit numeric code:

1931 *extended user group world*

1932 Where the *extended* access permission has the value

- 1933 +2 if sgid is set
1934 +4 if suid is set
1935 The *user group* and *world* access permissions have the value
1936 +1 if execute permission is granted
1937 +2 if write permission is granted
1938 +4 if read permission is granted
1939 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read
1940 and execute; and world read.

1941 7.3 NameIdentifier Format Identifiers

- 1942 The following identifiers MAY be used in the Format attribute of the <NameIdentifier> element (see
1943 Section 2.4.2.2) to refer to common formats for the content of the <NameIdentifier> element. The
1944 recommended identifiers shown below SHOULD be used in preference to the deprecated identifiers,
1945 which are planned to be removed in the next major version of the SAML assertion specification.

1946 7.3.1 Unspecified

- 1947 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
1948 The interpretation of the content of the <NameQualifier> element is left to individual implementations.

1949 7.3.2 Email Address

- 1950 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
1951 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#emailAddress
1952 Indicates that the content of the <NameIdentifier> element is in the form of an email address,
1953 specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form
1954 local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no
1955 comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

1956 7.3.3 X.509 Subject Name

- 1957 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1958 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName
1959 Indicates that the content of the <NameIdentifier> element is in the form specified for the contents of
1960 the <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors
1961 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
1962 differ from the rules given in IETF RFC 2253 [RFC 2253].

1963 7.3.4 Windows Domain Qualified Name

- 1964 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName
1965 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName
1966 Indicates that the content of the <NameIdentifier> element is a Windows domain qualified name. A
1967 Windows domain qualified user name is a string of the form "DomainName\UserName". The domain
1968 name and "\" separator MAY be omitted.

8 References

- 1969
- 1970 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 1971
- 1972 **[Kern-84]** B. Kernighan, Rob Pike. *The UNIX Programming Environment*, March 1984. Prentice Hall Computer Books.
- 1973
- 1974 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty Alliance Project, January 2003, http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf.
- 1975
- 1976
- 1977
- 1978 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December 1978.
- 1979
- 1980
- 1981 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*. IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 1982
- 1983 **[PKCS1]** B. Kaliski. *PKCS #1: RSA Encryption Version 2.0*. RSA Laboratories, also IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>.
- 1984
- 1985 **[PKCS7]** B. Kaliski. *PKCS #7: Cryptographic Message Syntax, Version 1.5*. IETF RFC 2315, March 1998. <http://www.ietf.org/rfc/rfc2315.txt>.
- 1986
- 1987 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF RFC 2459, January 1999. <http://www.ietf.org/rfc/rfc2459.txt>.
- 1988
- 1989
- 1990 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 1991
- 1992 **[RFC 2104]** H. Krawczyk et al. *HMAC: Keyed Hashing for Message Authentication*. IETF RFC 2104, February 1997. <http://www.ietf.org/rfc/rfc2104.txt>.
- 1993
- 1994 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 1995
- 1996 **[RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. <http://www.ietf.org/rfc/rfc2246.txt>.
- 1997
- 1998 **[RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. <http://www.ietf.org/rfc/rfc2253.txt>.
- 1999
- 2000
- 2001 **[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.
- 2002
- 2003 **[RFC 2630]** R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999. <http://www.ietf.org/rfc/rfc2630.txt>.
- 2004
- 2005 **[RFC 2648]** R. Moats. *A URN Namespace for IETF Documents*. IETF RFC 2648, August 1999. <http://www.ietf.org/rfc/rfc2648.txt>.
- 2006
- 2007 **[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. <http://www.ietf.org/rfc/rfc2822.txt>.
- 2008
- 2009 **[RFC 2945]** T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945, September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.
- 2010
- 2011 **[RFC 3075]** D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.
- 2012
- 2013 **[SAMLBind]** P. Mishra et al. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, November 2002. <http://www.oasis-open.org/committees/security/>.
- 2014
- 2015

2016	[SAMLConform]	R. Griffin, et al. <i>Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, November 2002. http://www.oasis-open.org/committees/security/ .
2017		
2018		
2019	[SAMLCore1.0]	Phillip Hallam-Baker et al. <i>Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, November 2002. http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf .
2020		
2021		
2022	[SAMLGloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, November 2002. http://www.oasis-open.org/committees/security/ .
2023		
2024		
2025	[SAMLXP-XSD]	P. Hallam-Baker et al. <i>SAML protocol schema</i> . OASIS, November 2002. http://www.oasis-open.org/committees/security/ .
2026		
2027	[SAMLSecure]	C. McLarn, et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, November 2002. http://www.oasis-open.org/committees/security/ .
2028		
2029		
2030	[SAML-XSD]	P. Hallam-Baker et al. <i>SAML assertion schema</i> . OASIS, November 2002. http://www.oasis-open.org/committees/security/ .
2031		
2032	[Schema1]	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/ .
2033		
2034	[Schema2]	P. V. Biron et al. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-2/ .
2035		
2036	[SPKI]	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999. http://www.ietf.org/rfc/rfc2693.txt .
2037		
2038		
2039	[UNICODE-C]	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. http://www.unicode.org/unicode/reports/tr15/tr15-21.html .
2040		
2041	[W3C-CHAR]	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. http://www.w3.org/TR/WD-charreq .
2042		
2043	[W3C-CharMod]	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. http://www.w3.org/TR/charmod/ .
2044		
2045	[X.500]	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
2046		
2047	[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/ .
2048		
2049		
2050	[XML]	T. Bray, et al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> . World Wide Web Consortium, October 2000. http://www.w3.org/TR/REC-xml .
2051		
2052	[XMLEnc]	T. Imamura, et al. <i>XML Encryption Syntax and Processing</i> . World Wide Web Consortium, December 2002. http://www.w3.org/TR/xmlenc-core/ .
2053		
2054	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , World Wide Web Consortium, February 2002. http://www.w3.org/TR/xmldsig-core/ .
2055		
2056	[XMLSig-XSD]	XML Signature Schema. World Wide Web Consortium. http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd .
2057		
2058		

2059 **Appendix A. Acknowledgments**

2060 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2061 Committee, whose voting members at the time of publication were:

- 2062 • Irving Reid, Baltimore Technologies
- 2063 • Hal Lockhart, BEA Systems
- 2064 • Ronald Jacobson, Computer Associates
- 2065 • John Hughes, Entegriety Solutions
- 2066 • Carlisle Adams, Entrust
- 2067 • Robert Griffin, Entrust
- 2068 • Scott Cantor, Individual
- 2069 • Bob Morgan, Individual
- 2070 • Clifford Thompson, Individual
- 2071 • Pdraig Moloney, NASA
- 2072 • Prateek Mishra, Netegrity (co-chair)
- 2073 • Frederick Hirsch, Nokia
- 2074 • Senthil Sengodan, Nokia
- 2075 • Timo Skytta, Nokia
- 2076 • Charles Knouse, Oblix
- 2077 • Steve Anderson, OpenNetwork
- 2078 • Simon Godik, OverXeer
- 2079 • Rob Philpott, RSA Security (co-chair)
- 2080 • Dipak Chopra, SAP
- 2081 • Jahan Moreh, Sigaba
- 2082 • Bhavna Bhatnagar, Sun Microsystems
- 2083 • Jeff Hodges, Sun Microsystems
- 2084 • Eve Maler, Sun Microsystems (coordinating editor)
- 2085 • Emily Xu, Sun Microsystems
- 2086 • Phillip Hallam-Baker, VeriSign

2087

Appendix B. Notices

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

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

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

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

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

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

Appendix C. Revision History

Draft	Who	What
01	Eve Maler	Cosmetic changes to bring spec up to 1.1 WD status. Fragment ID changes. (But forgot an important part of this! Completed it in draft 05.)
02	Eve Maler	Started adding new contributors to original list. Others who wish to be added should inform me. Approved SAML versioning changes. Core errata fixes as approved through draft-sstc-saml1.1-errata-05: E4, E5, E6, PE5 (RonM: please check lines 1390-1395), PE7, PE8. Need replacement text from Rob in order to fix PE9.
03	Eve Maler	Core errata fixes as approved through 1 April 2003: proposal for PE11 solution (requests by artifact and assertion ID), PE12 and the first half of PE13 (<RespondWith> and subject-matching clarifications), first half of PE14 (requestor → requester), “element’s type” in <RespondWith> description, new document repository location.
04	Eve Maler	Core errata fixes as approved through 8 April 2003: E7 (MAY NOT → MUST NOT), PE16 (clarifying the use of QNames in content), StatusCode treatment editorially normalized.
05	Eve Maler	Added new SAML Versioning text approved on 15 April 2003. Minor editorial cleanup up through the assertion section. Finished implementing the fragment ID fix.
06	Eve Maler	Made changes corresponding to the new constraint on the base of IDType (xsd:ID) and IDReferenceType (xsd:IDREF). Heavy copyedits. Fixed PE9 (AuthorityKind and deprecation of <AuthorityBinding>), conclusion of PE12 (deprecation of <RespondWith>), second half of PE13 (signature changes and C14N), conclusion of PE 18 (additional versioning wording provided in AI 0028). Made into a candidate last-call working draft.
07	Rob Philpott, Eve Maler	Editorial cleanup, particularly around the use of “issuer”, “authority”, “asserting/relying party”, and “requester/responder”. Removed the incorrect information about xsi:type being required when abstract types are used (it conflicted with the discussion about substitution groups). Added XML Signature example from Scott in Section 5.4.8. Cleaned up numerous cross-reference link problems.
08	Rob Philpott	Last Call Draft – accept all changes.
09	Prateek Mishra, Rob Philpott	Added schema and text for DoNotCacheCondition element. Removed some embedded comments and sorted contributor list by organization.
10	Rob Philpott	Finalized Last Call Drafts – accepted all changes.