



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

Working Draft ~~08, 15 March 2, 5 January~~ 2004

## Document identifier:

sstc-saml-core-2.0-draft-082

## Location:

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

## Editors:

[Scott Cantor, individual \(cantor.2@osu.edu\)](mailto:cantor.2@osu.edu)  
[John Kemp, Nokia \(john.kemp@nokia.com\)](mailto:john.kemp@nokia.com)  
Eve Maler, Sun Microsystems ([eve.maler@sun.com](mailto:eve.maler@sun.com))

## Contributors:

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

38 **Abstract:**

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

41 **Status:**

42 This is a working draft produced by the Security Services Technical Committee. Publication of  
43 this draft does not imply TC endorsement. This is an active working draft that may be updated,  
44 replaced, or obsoleted at any time. **See the Revision History for details of changes made in  
45 this revision.**

46 Committee members should submit comments and potential errata to the [security-  
47 services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them to the [security-services-  
49 comment@lists.oasis-open.org](mailto:security-services-<br/>48 comment@lists.oasis-open.org) list (to post, you must subscribe; to subscribe, send a message  
50 to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with "subscribe" in the body) or use  
51 other OASIS-supported means of submitting comments. The committee will publish vetted errata  
on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

52 For information on whether any patents have been disclosed that may be essential to  
53 implementing this specification, and any offers of patent licensing terms, please refer to the  
54 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-  
open.org/committees/security/ipr.php](http://www.oasis-<br/>55 open.org/committees/security/ipr.php)).

# Table of Contents

57	1 Introduction.....	7
58	1.1 Notation.....	7
59	1.2 Schema Organization and Namespaces.....	8
60	1.2.1 String and URI Values.....	8
61	1.2.2 Time Values.....	8
62	1.2.3 ID and ID Reference Values.....	8
63	1.2.4 Comparing SAML Values.....	9
64	2 SAML Assertions.....	10
65	2.1 Schema Header and Namespace Declarations.....	10
66	2.2 Simple Types.....	10
67	2.2.1 Simple Type DecisionType.....	11
68	2.3 Name Identifiers.....	11
69	2.3.1 Element <BaseIdentifier>.....	11
70	2.3.2 Element <NameIdentifier>.....	12
71	2.3.3 Element <EncryptedIdentifier>.....	12
72	2.3.4 Element <Issuer>.....	13
73	2.4 Assertions.....	13
74	2.4.1 Element <AssertionIDReference>.....	13
75	2.4.2 Element <AssertionURIReference>.....	13
76	2.4.3 Element <Assertion>.....	14
77	2.4.3.1 Element <Subject>.....	15
78	2.4.3.2 Element <Conditions>.....	16
79	2.4.3.2.1 Attributes NotBefore and NotOnOrAfter.....	17
80	2.4.3.2.2 Element <Condition>.....	17
81	2.4.3.2.3 Elements <AudienceRestrictionCondition> and <Audience>.....	17
82	2.4.3.2.4 Element <DoNotCacheCondition>.....	18
83	2.4.3.2.5 Element <ProxyRestrictionCondition>.....	18
84	2.4.3.3 Element <Advice>.....	19
85	2.5 Statements.....	20
86	2.5.1 Element <Statement>.....	20
87	2.5.1.1 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
88	<SubjectConfirmationData>.....	20
89	2.5.2 Element <AuthenticationStatement>.....	21
90	2.5.2.1 Element <SubjectLocality>.....	22
91	2.5.2.2 Element <AuthnContext>.....	22
92	2.5.3 Element <AttributeStatement>.....	23
93	2.5.3.1 Elements <AttributeDesignator> and <Attribute>.....	23
94	2.5.3.1.1 Element <AttributeValue>.....	24
95	2.5.4 Element <AuthorizationDecisionStatement>.....	25
96	2.5.4.1 Element <Action>.....	26
97	2.5.4.2 Element <Evidence>.....	27
98	3 SAML Protocols.....	28
99	3.1 Schema Header and Namespace Declarations.....	28
100	3.2 Requests and Responses.....	29
101	3.2.1 Complex Type RequestAbstractType.....	29
102	3.2.1.1 Element <RelayState>.....	30
103	3.2.2 Complex Type StatusResponseType.....	30
104	3.2.2.1 Element <Status>.....	31

105	3.2.2.2 Element <StatusCode>.....	32
106	3.2.2.3 Element <StatusMessage>.....	34
107	3.2.2.4 Element <StatusDetail>.....	34
108	3.3 Assertion Query and Request Protocol.....	34
109	3.3.1 Element <AssertionIDRequest>.....	34
110	3.3.2 Queries.....	35
111	3.3.2.1 Element <SubjectQuery>.....	35
112	3.3.2.2 Element <AuthenticationQuery>.....	35
113	3.3.2.3 Element <AttributeQuery>.....	36
114	3.3.2.4 Element <AuthorizationDecisionQuery>.....	37
115	3.3.3 Element <Response>.....	38
116	3.3.3.1 Processing Rules.....	38
117	3.4 Authentication Request Protocol.....	38
118	3.4.1 Element <AuthnRequest>.....	39
119	3.4.1.1 Element <NameIDPolicy>.....	41
120	3.4.1.2 Element <RequestAuthnContext>.....	42
121	3.4.1.3 Element <Scoping>.....	43
122	3.4.1.4 Element <IDPList>.....	43
123	3.4.1.5 Element <IDPEntry>.....	44
124	3.4.1.6 Processing Rules.....	44
125	3.4.1.7 Proxying.....	45
126	3.4.1.7.1 Processing Rules.....	46
127	3.5 Artifact Protocol.....	47
128	3.5.1 Element <ArtifactRequest>.....	47
129	3.5.2 Element <ArtifactResponse>.....	48
130	3.5.3 Processing Rules.....	48
131	3.6 Federated Name Registration Protocol.....	49
132	3.6.1 Element <RegisterNameIdentifierRequest>.....	49
133	3.6.2 Element <RegisterNameIdentifierResponse>.....	50
134	3.6.3 Processing Rules.....	50
135	3.7 Federation Termination Protocol.....	51
136	3.7.1 Element <FederationTerminationNotification>.....	51
137	3.7.2 Element <FederationTerminationResponse>.....	52
138	3.7.3 Processing Rules.....	52
139	3.8 Single Logout Protocol.....	52
140	3.8.1 Element <LogoutRequest>.....	53
141	3.8.2 Element <LogoutResponse>.....	53
142	3.8.3 Processing Rules.....	54
143	3.8.3.1 Session Participant Rules.....	54
144	3.8.3.2 Session Authority Rules.....	54
145	3.9 Name Identifier Mapping Protocol.....	55
146	3.9.1 Element <NameIdentifierMappingRequest>.....	55
147	3.9.2 Element <NameIdentifierMappingResponse>.....	56
148	3.9.3 Processing Rules.....	57
149	4 SAML Versioning.....	58
150	4.1 SAML Specification Set Version.....	58
151	4.1.1 Schema Version.....	58
152	4.1.2 SAML Assertion Version.....	58
153	4.1.3 SAML Protocol Version.....	59
154	4.1.3.1 Request Version.....	59
155	4.1.4 Response Version.....	59
156	4.1.5 Permissible Version Combinations.....	60

157	4.2 SAML Namespace Version.....	60
158	4.2.1 Schema Evolution.....	60
159	5 SAML and XML Signature Syntax and Processing.....	61
160	5.1 Signing Assertions.....	62
161	5.2 Request/Response Signing.....	62
162	5.3 Signature Inheritance.....	62
163	5.4 XML Signature Profile.....	62
164	5.4.1 Signing Formats and Algorithms.....	62
165	5.4.2 References.....	62
166	5.4.3 Canonicalization Method.....	63
167	5.4.4 Transforms.....	63
168	5.4.5 KeyInfo.....	63
169	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	63
170	5.4.8 Example.....	63
171	6 SAML Extensions.....	66
172	6.1 Assertion Schema Extension.....	66
173	6.2 Protocol Schema Extension.....	66
174	7 SAML-Defined Identifiers.....	68
175	7.1 Authentication Method Identifiers.....	68
176	7.1.1 Password.....	68
177	7.1.2 Kerberos .....	68
178	7.1.3 Secure Remote Password (SRP).....	68
179	7.1.4 Hardware Token.....	69
180	7.1.5 SSL/TLS Certificate Based Client Authentication.....	69
181	7.1.6 X.509 Public Key .....	69
182	7.1.7 PGP Public Key .....	69
183	7.1.8 SPKI Public Key .....	69
184	7.1.9 XKMS Public Key .....	69
185	7.1.10 XML Digital Signature .....	69
186	7.1.11 Authentication Context.....	70
187	7.1.12 Unspecified .....	70
188	7.2 Action Namespace Identifiers.....	70
189	7.2.1 Read/Write/Execute/Delete/Control.....	70
190	7.2.2 Read/Write/Execute/Delete/Control with Negation.....	70
191	7.2.3 Get/Head/Put/Post.....	71
192	7.2.4 UNIX File Permissions.....	71
193	7.3 NameIdentifier Format Identifiers.....	71
194	7.3.1 Unspecified.....	72
195	7.3.2 Email Address.....	72
196	7.3.3 X.509 Subject Name.....	72
197	7.3.4 Windows Domain Qualified Name.....	72
198	7.3.5 Provider Identifier.....	72
199	7.3.6 Federated Identifier.....	72
200	7.3.7 Transient Identifier.....	73
201	7.4 Attribute NameFormat Identifiers.....	73
202	7.4.1 Unspecified.....	73
203	7.4.2 URI Reference.....	73
204	7.5 Attribute ValueType Identifiers.....	74
205	7.5.1 Application-Specific Value Type.....	74
206	8 References.....	75

207	8.1 Normative References.....	75
208	8.2 Non-Normative References.....	75
209		

---

# 1 Introduction

210

211 This specification defines the syntax and semantics for [Security Assertion Markup Language \(SAML\)](#)  
212 [assertions and the protocols for requesting and returning them. SAML assertions, requests, and](#)  
213 [responses are encoded in XML \[XML\] and use XML namespaces \[XMLNS\]. They are typically embedded](#)  
214 [in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The](#)  
215 [SAML specification for binding XML-encoded Security Assertion Markup Language \(SAML\) assertions,](#)  
216 [protocol requests, and protocol responses. These constructs are typically embedded in other structures](#)  
217 [for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML specification](#)  
218 [for bindings and profiles](#) [SAMLBind] provides frameworks for this embedding and transport. Files  
219 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAML-P-XSD] are  
220 available.

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

## 1.1 Notation

222

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

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

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

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

233         Listings of SAML schemas appear like this.

234         Example code listings appear like this.

236 In cases of disagreement between the SAML schema [documents \[SAML-XSD\] \[SAML-P-XSD\]](#) and [this](#)  
237 [specification, the schema document files \[SAML-XSD\] \[SAML-P-XSD\]](#) and [this specification, the schema](#)  
238 [files](#) take precedence.

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

- 242 • The prefix `saml:` stands for the SAML assertion namespace,  
243 <urn:oasis:names:tc:SAML:2.0:assertion>.
- 244 • The prefix `samlp:` stands for the SAML request-response protocol namespace,  
245 <urn:oasis:names:tc:SAML:2.0:protocol>.
- 246 • The prefix `ds:` stands for the W3C XML Signature namespace,  
247 <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 248 • The prefix `xenc:` stands for the W3C XML Encryption namespace,  
249 <http://www.w3.org/2001/04/xmlenc#> [XMLEnc-XSD].

- 250 • The prefix `xsd:` stands for the W3C XML Schema namespace.  
251 <http://www.w3.org/2001/XMLSchema> [Schema1],- in example listings. In schema listings, this  
252 is the default namespace and no prefix is shown.

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

## 255 1.2 Schema Organization and Namespaces

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

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

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

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

262 The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
263 schema for XML Signature [XMLSIG-XSD], which is associated with the following XML namespace:

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

265 See Section SAML Namespace Version for information on SAML namespace versioning.

### 266 1.2.1 String and URI Values

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

### 273 1.2.2 Time Values

274 All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes  
275 specification [Schema1], and MUST be expressed in UTC form.

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

### 278 1.2.3 ID and ID Reference Values

279 The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses.  
280 Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties [in  
281 addition to those imposed by the definition of the \*\*xsd:ID\*\* type itself](#):

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

286 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the  
287 implementation. In the case that a pseudorandom technique is employed, the probability of two randomly



288 chosen identifiers being identical MUST be less than or equal to  $2^{128}$  and SHOULD be less than or equal  
289 to  $2^{160}$ . This requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in  
290 length. The encoding must conform to the rules defining the **xsd:ID** datatype.

291 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that  
292 **xsd>IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML  
293 reference identifier might actually be defined in a document separate from that in which the identifier  
294 reference is used, ~~which violates the **xsd>IDREF** requirement that its value-XML-requires that names of~~  
295 ~~type **xsd>IDREF** must~~ match the value of an ID attribute on some element in the same XML document.

## 296 1.2.4 Comparing SAML Values

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

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

308 Applications that compare data received in SAML documents to data from external sources MUST take  
309 into account the normalization rules specified for XML. Text contained within elements is normalized so  
310 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the  
311 XML Recommendation [XML] ~~§2.11. Attribute values defined as strings (or types derived from strings) are~~  
312 ~~normalized as described in §2.11. Attribute values defined as strings (or types derived from strings) are~~  
313 ~~normalized as described in~~[XML] §3.3.3. All white space characters are replaced with blanks (ASCII  
314 code 32<sub>Decimal</sub>).

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

## 2 SAML Assertions

318

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

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

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

### 2.1 Schema Header and Namespace Declarations

333

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

```
336 <schema  
337     targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
338     xmlns="http://www.w3.org/2001/XMLSchema"  
339     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
340     xmlns:ds="http://www.w3.org/2000/09/xmldsig#" -  
341     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" -  
342     elementFormDefault="unqualified"  
343     attributeFormDefault="unqualified"  
344     blockDefault="substitution"  
345     version="2.0">  
346     <import namespace="http://www.w3.org/2000/09/xmldsig#" -  
347     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
348     schema.xsd"/>  
349     <import namespace="http://www.w3.org/2001/04/xmlenc#" -  
350     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-  
351     schema.xsd"/>  
352     <annotation>  
353         <documentation>  
354             Document identifier: sstc-saml-schema-assertion-2.0  
355             Location: http://www.oasis-  
356     open.org/committees/documents.php?wg_abbrev=security  
357         </documentation>  
358         Revision history:  
359         V2.0:  
360         Updated the schema and namespace to V2.0.  
361         Removed <AuthorityBinding> and corresponding type.  
362     </documentation>  
363     </annotation>  
364     ...  
365 </schema>
```

### 2.2 Simple Types

366

367 The following section ~~defines(s)~~ ~~define~~ the SAML assertion-related simple types.

## 368 2.2.1 Simple Type DecisionType

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

371 Permit

372     The specified action is permitted.

373 Deny

374     The specified action is denied.

375 Indeterminate

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

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

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

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

## 391 2.3 Name Identifiers

392 The following sections define the SAML constructs that contain descriptive identifiers of subjects and  
393 assertion and message issuers.

### 394 2.3.1 Element `<BaseIdentifier>`

395 The `<BaseIdentifier>` element is an extension point that allows applications to add new kinds of  
396 identifiers. Its `BaseIdentifierAbstractType` complex type is abstract and is thus usable only as the base  
397 of a derived type. It defines the following common attributes for all identifier representations:

398 `NameQualifier` [Optional]

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

401 `SPNameQualifier` [Optional]

402     Further qualifies a federated identifier with the name of the service provider or affiliation of  
403     providers which has federated the principal's identity.

404 The following schema fragment defines the `<BaseIdentifier>` element and its `BaseIdentifierType`  
405 complex type:

```
406 <element name="BaseIdentifier" type="saml:BaseIdentifierAbstractType"/>  
407 <complexType name="BaseIdentifierAbstractType" abstract="true">  
408   <complexContent>  
409     <extension base="anyType"/>
```

```
410 |         <attribute name="NameQualifier" type="string" use="optional"/>
411 |         <attribute name="SPNameQualifier" type="string" use="optional"/>
412 |     </extension>
413 | </complexContent>
414 | </complexType>
```

### 415 | **2.3.2 Element <NameIdentifier>**

416 | The <NameIdentifier> element is of type **NameIdentifierType**, which restricts  
417 | **BaseIdentifierAbstractType** to simple string content and provides additional attributes as follows:

#### 418 | Format [Optional]

419 | A URI reference representing the classification of string-based identifier information. See Section  
420 | NameIdentifier Format Identifiers for some URI references that MAY be used as the value of the  
421 | Format attribute and their associated descriptions and processing rules. If no Format value is  
422 | provided, the identifier urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see Section  
423 | Unspecified) is in effect.

424 | When a Format value other than those specified in Section NameIdentifier Format Identifiers is  
425 | used, the content of the <NameIdentifier> element is to be interpreted according to the  
426 | specification of that format as defined outside of this specification. If not otherwise indicated by  
427 | the specification of the format, issues of anonymity, pseudonymity, and the persistence of the  
428 | identifier with respect to the asserting and relying parties are implementation-specific.

#### 429 | SPProvidedIdentifier [Optional]

430 | The name identifier established by the service provider or affiliation of providers for the principal,  
431 | if different from the primary name identifier given in the content of the <NameIdentifier>  
432 | element.

433 | The following schema fragment defines the <NameIdentifier> element and its **NameIdentifierType**  
434 | complex type:

```
435 | <element name="NameIdentifier" type="saml:NameIdentifierType"/>
436 | <complexType name="NameIdentifierType" mixed="false">
437 |     <simpleContent>
438 |         <restriction base="saml:BaseIdentifierAbstractType">
439 |             <simpleType>
440 |                 <restriction base="string"/>
441 |             </simpleType>
442 |             <attribute name="Format" type="anyURI" use="optional"/>
443 |             <attribute name="SPProvidedIdentifier" type="string"
444 | use="optional"/>
445 |         </restriction>
446 |     </simpleContent>
447 | </complexType>
```

### 448 | **2.3.3 Element <EncryptedIdentifier>**

449 | The <EncryptedIdentifier> element extends **BaseIdentifierAbstractType** to carry the content of  
450 | the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification  
451 | [XMLEnc]. The <EncryptedIdentifier> element contains the following additional elements and  
452 | attributes:

#### 453 | <xenc:EncryptedData> [Required]

454 | The encrypted content and associated encryption details, as defined by [XMLEnc]. The  
455 | encrypted content MUST contain an element that has a type that is derived from  
456 | **BaseIdentifierAbstractType** or from **AssertionType**.

457 `<xenc:EncryptedKey>` [Zero or more]

458 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a  
459 Recipient attribute that specifies the entity for whom the key has been encrypted.

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

463 The following schema fragment defines the `<EncryptedIdentifier>` element and its  
464 `EncryptedIdentifierType` complex type:

```
465 <element name="EncryptedIdentifier" type="saml:EncryptedIdentifierType"/>  
466 <complexType name="EncryptedIdentifierType" mixed="false">  
467 <complexContent>  
468 <restriction base="saml:BaseIdentifierType">  
469 <sequence>  
470 <element ref="xenc:EncryptedData"/>  
471 <element ref="xenc:EncryptedKey" minOccurs="0"  
472 maxOccurs="unbounded"/>  
473 </sequence>  
474 </restriction>  
475 </complexContent>  
476 </complexType>
```

### 477 **2.3.4 Element `<Issuer>`**

478 The `<Issuer>` element, with complex type `NameIdentifierType`, provides information about the issuer  
479 of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's  
480 name, but permits various attributes of descriptive metadata.

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

```
482 <element name="Issuer" type="saml:NameIdentifierType"/>
```

## 483 **2.4 Assertions**

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

### 485 **2.4.1 Element `<AssertionIDReference>`**

486 The `<AssertionIDReference>` element makes a reference to a SAML assertion by its unique  
487 identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is  
488 not specified as part of the reference.

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

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

### 491 **2.4.2 Element `<AssertionURIReference>`**

492 The `<AssertionURIReference>` element makes a reference to a SAML assertion by its uniform  
493 resource identifier (URI). Dereferencing the URI (in a fashion dictated by the URI) is intended to produce  
494 the assertion.

495 The following schema fragment defines the `<AssertionURIReference>` element:

```
496 <element name="AssertionURIReference" type="anyURI"/>
```

### 497 | 2.4.3 Element <Assertion>

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

500 | MajorVersion [Required]

501 |     The major version of this assertion. The identifier for the version of SAML defined in this  
502 |     specification is 2±. SAML versioning is discussed in Section SAML Versioning.

503 | MinorVersion [Required]

504 |     The minor version of this assertion. The identifier for the version of SAML defined in this  
505 |     specification is 0±. SAML versioning is discussed in Section SAML Versioning.

506 | AssertionID [Required]

507 |     The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in  
508 |     Section 1.2.3 for identifier uniqueness.

509 | ~~Issuer [Required]~~

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

513 | IssueInstant [Required]

514 |     The time instant of issue in UTC, as described in Section Time Values.

515 | ~~<Issuer> [Required]~~

516 |     ~~The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be~~  
517 |     ~~unambiguous to the intended relying parties. If the Format attribute is omitted, the identifier~~  
518 |     ~~urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see section 7.3.1) is~~  
519 |     ~~assumed.~~

520 |     ~~This specification defines no relationship between the entity represented by this element and the~~  
521 |     ~~signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the~~  
522 |     ~~assertion or to specific profiles are application-specific.~~

523 | ~~<Subject> [Required]~~

524 |     ~~The subject of the statement(s) in the assertion.~~

525 | ~~<ds:Signature> [Optional]~~

526 |     ~~An XML Signature that authenticates the assertion, as described in Section SAML and XML~~  
527 |     ~~Signature Syntax and Processing.~~

528 | <Conditions> [Optional]

529 |     Conditions that MUST be taken into account in assessing the validity of and/or using the assertion.

530 | <Advice> [Optional]

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

533 | ~~<ds:Signature> [Optional]~~

534 |     ~~An XML Signature that authenticates the assertion, as described in Section SAML and XML~~  
535 |     ~~Signature Syntax and Processing.~~

536 | One or more of the following statement elements:

537 <Statement>  
538 A statement defined in an extension schema.

539 ~~<SubjectStatement>~~  
540 ~~A subject statement defined in an extension schema.~~

541 <AuthenticationStatement>  
542 An authentication statement.

543 <AuthorizationDecisionStatement>  
544 An authorization decision statement.

545 <AttributeStatement>  
546 An attribute statement.

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

```
549 <element name="Assertion" type="saml:AssertionType"/>  
550 <complexType name="AssertionType">  
551   <sequence>  
552     <del>element ref="saml:Issuer"/>  
553     <del>element ref="saml:Subject"/>  
554     <del>element ref="ds:Signature" minOccurs="0"/>  
555     <del>element ref="saml:Conditions" minOccurs="0"/>  
556     <del>element ref="saml:Advice" minOccurs="0"/>  
557     <del>choice maxOccurs="unbounded">  
558       <del>element ref="saml:Statement"/>  
559       <del>element ref="saml:SubjectStatement"/>  
560       <del>element ref="saml:AuthenticationStatement"/>  
561       <del>element ref="saml:AuthorizationDecisionStatement"/>  
562       <del>element ref="saml:AttributeStatement"/>  
563     </del>choice</del>  
564     <del>element ref="ds:Signature" minOccurs="0"/>  
565   </del>sequence</del>  
566   <del>attribute name="MajorVersion" type="integer" use="required"/>  
567   <del>attribute name="MinorVersion" type="integer" use="required"/>  
568   <del>attribute name="AssertionID" type="ID" use="required"/>  
569   <del>attribute name="Issuer" type="string" use="required"/>  
570   <del>attribute name="IssueInstant" type="dateTime" use="required"/>  
571 </del>complexType</del>
```

### 572 **2.4.3.1 Element <Subject>**

573 The <Subject> element specifies the principal that is the subject of all of the (one or more) statements  
574 in the assertion. It contains a name identifier, a series of one or more subject confirmations, or both:

575 <NameIdentifier>, <EncryptedIdentifier>, or <BaseIdentifier>

576 Identifies the subject.

577 <SubjectConfirmation>

578 Information that allows the subject to be authenticated. If more than one subject confirmation is  
579 provided, then usage of any one of them is sufficient to confirm the subject for the purpose of  
580 applying the assertion.

581 If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML  
582 authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it  
583 can treat the entity presenting the assertion to the relying party as the entity that the SAML authority

584 associates with the name identifier. A <Subject> element SHOULD NOT identify more than one  
585 principal.

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

```
587 <element name="Subject" type="saml:SubjectType"/>
588 <complexType name="SubjectType">
589   <choice>
590     <sequence>
591       <choice>
592         <element ref="saml:BaseIdentifier"/>
593         <element ref="saml:NameIdentifier"/>
594         <element ref="saml:EncryptedIdentifier"/>
595       </choice>
596       <element ref="saml:SubjectConfirmation" minOccurs="0"
597 maxOccurs="unbounded"/>
598     </sequence>
599     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
600   </choice>
601 </complexType>
```

### 602 2.4.3.2 Element <Conditions>

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

604 NotBefore [Optional]

605 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC  
606 as described in Section Time Values.

607 NotOnOrAfter [Optional]

608 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as  
609 described in Section Time Values.

610 <Condition> [Any Number]

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

612 <AudienceRestrictionCondition> [Any Number]

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

614 <DoNotCacheCondition> [Any Number]

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

617 <ProxyRestrictionCondition> [Any Number]

618 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
619 assertions of their own on the basis of the information contained in the original assertion.

620 The following schema fragment defines the <Conditions> element and its ConditionsType complex  
621 type:

```
622 <element name="Conditions" type="saml:ConditionsType"/>
623 <complexType name="ConditionsType">
624   <choice minOccurs="0" maxOccurs="unbounded">
625     <element ref="saml:AudienceRestrictionCondition"/>
626     <element ref="saml:DoNotCacheCondition"><element
627 ref="saml:DoNotCacheCondition">
628     <element ref="saml:ProxyRestrictionCondition"/>
629     <element ref="saml:Condition"/>
630   </choice>
631   <attribute name="NotBefore" type="dateTime" use="optional"/>
```



```
632 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
633 </complexType>
```

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

- 638 1. If no sub-elements or attributes are supplied in the `<Conditions>` element, then the assertion is  
639 considered to be **Valid**.
- 640 2. If any sub-element or attribute of the `<Conditions>` element is determined to be invalid, then the  
641 assertion is **Invalid**.
- 642 3. If any sub-element or attribute of the `<Conditions>` element cannot be evaluated, then the validity  
643 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 644 4. If all sub-elements and attributes of the `<Conditions>` element are determined to be **Valid**, then  
645 the assertion is considered to be **Valid**.

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

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

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

#### 655 2.4.3.2.1 Attributes `NotBefore` and `NotOnOrAfter`

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

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

659 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
660 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the  
661 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the  
662 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),  
663 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither  
664 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is  
665 valid at any time.

666 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is  
667 built in to the W3C XML Schema Datatypes specification [Schema2]. All time instants are specified in  
668 Universal Coordinated Time (UTC) as described in Section Time Values.

669 Implementations MUST NOT generate time instants that specify leap seconds.

#### 670 2.4.3.2.2 Element `<Condition>`

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

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

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

### 677 2.4.3.2.3 Elements <AudienceRestrictionCondition> and <Audience>

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

683 <Audience>

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

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

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

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

```
696 <element name="AudienceRestrictionCondition"  
697 type="saml:AudienceRestrictionConditionType"/>  
698 <complexType name="AudienceRestrictionConditionType">  
699 <complexContent>  
700 <extension base="saml:ConditionAbstractType">  
701 <sequence>  
702 <element ref="saml:Audience" maxOccurs="unbounded"/>  
703 </sequence>  
704 </extension>  
705 </complexContent>  
706 </complexType>  
707 <element name="Audience" type="anyURI"/>
```

### 708 2.4.3.2.4 Element <DoNotCacheCondition>

709 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be  
710 retained for future use. Note that no relying party is required to perform caching. However, any that do so  
711 MUST observe this conditionA SAML authority SHOULD NOT include more than one  
712 <DoNotCacheCondition> element within a <Conditions> element of an assertion. Note that no  
713 Relying Party implementation is required to perform caching. However, any that do so MUST observe  
714 this condition. If multiple <DoNotCacheCondition> elements appear within a <Conditions> element,  
715 a Relying Party MUST treat the multiple elements as though a single <DoNotCacheCondition>  
716 element was specified. For the purposes of determining the validity of the <Conditions> element, the  
717 <DoNotCacheCondition> (see Section ) is considered to always be valid.

718 A SAML authority SHOULD NOT include more than one <DoNotCacheCondition> element within a  
719 <Conditions> element of an assertion. If multiple <DoNotCacheCondition> elements appear within

720 a <Conditions> element, a Relying Party MUST treat the multiple elements as though a single  
721 <DoNotCacheCondition> element was specified.

722 For the purposes of determining the validity of the <Conditions> element, the  
723 <DoNotCacheCondition> is considered to always be valid.

724 The following schema fragment defines the <DoNotCacheCondition> element and its  
725 DoNotCacheConditionType complex type:

```
726 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType"/>  
727 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType"/>  
728 <complexType name="DoNotCacheConditionType">  
729   <complexContent>  
730     <extension base="saml:ConditionAbstractType"/> </complexContent>  
731 <extension base="saml:ConditionAbstractType"/>  
732 </complexType></complexContent>
```

### 733 **2.4.3.2.5 Element <ProxyRestrictionCondition>**

734 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
735 assertions of their own on the basis of the information contained in the original assertion. A relying party  
736 MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis  
737 of an assertion containing such a condition.

738 The <ProxyRestrictionCondition> element contains the following elements and attributes:

739 Count [Optional]

740 Specifies the number of indirections that MAY exist between this assertion and an assertion which  
741 has ultimately been issued on the basis of it.

742 <Audience> [Zero or More]

743 Specifies the set of audiences to whom new assertions MAY be issued on the basis of this assertion.

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

748 If no <Audience> elements are specified, then no restrictions are made upon the relying parties to  
749 whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves  
750 contain an <AudienceRestrictionCondition> element with at least one of the <Audience>  
751 elements present in the previous <ProxyRestrictionCondition> element, and no <Audience>  
752 elements present that were not in the previous <ProxyRestrictionCondition> element.

753 A SAML authority SHOULD NOT include more than one <ProxyRestrictionCondition> element  
754 within a <Conditions> element of an assertion. If multiple <ProxyRestrictionCondition>  
755 elements appear within a <Conditions> element, a relying party MUST treat the multiple elements as  
756 though a single <ProxyRestrictionCondition> element was specified, with a Count value equal to  
757 the lowest of any specified, and the set of <Audience> elements consisting of the union of the elements  
758 specified.

759 For the purposes of determining the validity of the <Conditions> element, the  
760 <ProxyRestrictionCondition> is considered to always be valid.

761 The following schema fragment defines the <ProxyRestrictionCondition> element and its  
762 ProxyRestrictionConditionType complex type:

```

763 <element name="ProxyRestrictionCondition"
764 type="saml:ProxyRestrictionConditionType"/>
765 <complexType name="ProxyRestrictionConditionType">
766 <complexContent>
767 <extension base="saml:ConditionAbstractType">
768 <sequence>
769 <element ref="saml:Audience" minOccurs="0"
770 maxOccurs="unbounded"/>
771 </sequence>
772 <attribute name="Count" type="nonNegativeInteger"
773 use="optional"/>
774 </extension>
775 </complexContent>
776 </complexType>

```

### 777 2.4.3.3 Element <Advice>

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

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

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

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

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

```

790 <element name="Advice" type="saml:AdviceType"/>
791 <complexType name="AdviceType">
792 <choice minOccurs="0" maxOccurs="unbounded">
793 <element ref="saml:AssertionIDReference"/>
794 <element ref="saml:AssertionURIReference"/>
795 <element ref="saml:Assertion"/>
796 <any namespace="##other" processContents="lax"/>
797 </choice>
798 </complexType>

```

## 799 2.5 Statements

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

### 801 2.5.1 Element <Statement>

802 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
803 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable  
804 only as the base of a derived type. This element has an optional attribute:

805 SessionIndex [Optional]

806 Indexes a particular session between the subject and the authority issuing this statement. The value  
807 of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST

808 | NOT be a globally unique value for a principal's session at the authority.

809 | The following schema fragment defines the <Statement> element and its **StatementAbstractType**  
810 | complex type:

```
811 | <element name="Statement" type="saml:StatementAbstractType"/>  
812 | <complexType name="StatementAbstractType" abstract="true">  
813 |   <attribute name="SessionIndex" type="string" use="optional"/></Element  
814 | <SubjectStatement>
```

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

819 | The following schema fragment defines the <SubjectStatement> element and its  
820 | **SubjectStatementAbstractType** abstract type:

```
821 | <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
822 | <complexType name="SubjectStatementAbstractType" abstract="true">  
823 |   <complexContent>  
824 |     <extension base="saml:StatementAbstractType">  
825 |       <sequence>  
826 |         <element ref="saml:Subject"/>  
827 |       </sequence>  
828 |     </extension>  
829 |   </complexContent>  
830 | </complexType>
```

### 831 | 2.5.1.1 Element <Subject>

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

834 | <NameIdentifier>

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

836 | <SubjectConfirmation>

837 | Information that allows the subject to be authenticated.

838 | If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the  
839 | SAML authority is asserting that if the SAML relying party performs the specified  
840 | <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying  
841 | party is the entity that the SAML authority associates with the <NameIdentifier>. A <Subject>  
842 | element SHOULD NOT identify more than one principal.

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

```
844 | <element name="Subject" type="saml:SubjectType"/>  
845 | <complexType name="SubjectType">  
846 |   <choice>  
847 |     <sequence>  
848 |       <element ref="saml:NameIdentifier"/>  
849 |       <element ref="saml:SubjectConfirmation" minOccurs="0"/>  
850 |     </sequence>  
851 |     <element ref="saml:SubjectConfirmation"/>  
852 |   </choice>  
853 | </complexType>
```

### 854 **2.5.1.2 Element <BaseNameIdentifier>**

855 The <BaseNameIdentifier> element is an extension point that allows applications to add new kinds  
856 of name identifiers. Its **BaseNameIdentifierAbstractType** complex type is abstract and is thus usable  
857 only as the base of a derived type. It defines the following common attributes for all name identifier  
858 representations:

#### 859 **NameQualifier** [Optional]

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

#### 862 **SPNameQualifier** [Optional]

863 Further qualifies a federated name identifier with the name of the service provider or affiliation of  
864 providers which has federated the principal's identity.

#### 865 **NotBefore** [Optional]

866 The date and time at which the name identifier becomes usable for referring to the subject.  
867 Generally used when encrypting the resulting element to indicate the time at which the  
868 encryption was performed, so that decrypting parties may enforce time-sensitive policies on use.

#### 869 **NotOnOrAfter** [Optional]

870 Indicates the time at which the identifier should no longer be used to refer to the subject.  
871 Generally used with encrypted or transient identifiers.

872 The **NotBefore** and **NotOnOrAfter** attributes do not affect or interact with the validity of an assertion  
873 whose subject contains a name identifier decorated with them. Rather, they represent the validity of the  
874 binding of the name identifier to the subject of the assertion.

875 The following schema fragment defines the <BaseNameIdentifier> element and its  
876 **BaseNameIdentifierType** complex type:

```
877 <element name="BaseNameIdentifier" type="saml:BaseNameIdentifierAbstractType"/>  
878 <complexType name="BaseNameIdentifierAbstractType" abstract="true">  
879   <complexContent>  
880     <extension base="anyType">  
881       <attribute name="NameQualifier" type="string" use="optional"/>  
882       <attribute name="SPNameQualifier" type="string" use="optional"/>  
883       <attribute name="NotBefore" type="dateTime" use="optional"/>  
884       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
885     </extension>  
886   </complexContent>  
887 </complexType>
```

### 888 **2.5.1.3 Element <NameIdentifier>**

889 The <NameIdentifier> element is of type **NameIdentifierType**, which restricts  
890 **BaseNameIdentifierAbstractType** to simple string content and provides additional attributes as follows:

#### 891 **Format** [Optional]

892 A URI reference representing the classification of string-based identifier information. See Section  
893 **NameIdentifier-Format Identifiers** for some URI references that MAY be used as the value of the  
894 **Format** attribute, and associated descriptions of the content, and processing rules. If no  
895 **Format** value is provided, the identifier urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified  
896 (see Section **Unspecified**) is in effect. When a **Format** value other than those specified in  
897 Section **NameIdentifier-Format Identifiers** is used, the content of the <NameIdentifier>  
898 element is to be interpreted according to the specification of that format as defined outside of this

899 specification. If not otherwise indicated by the specification of the format, issues of anonymity,  
900 pseudonymity, and the persistence of the identifier with respect to the asserting and relying  
901 parties are implementation-specific.

902 **SPProvidedIdentifier** [Optional]

903 The name identifier established by the service provider or affiliation of providers for the principal,  
904 if different from the primary name identifier given in the content of the `<NameIdentifier>`  
905 element.

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

```
908 <element name="NameIdentifier" type="saml:NameIdentifierType"  
909 substitutionGroup="saml:BaseNameIdentifier"/>  
910 <complexType name="NameIdentifierType" mixed="false">  
911 <simpleContent>  
912 <restriction base="saml:BaseNameIdentifierAbstractType">  
913 <simpleType>  
914 <restriction base="string"/>  
915 </simpleType>  
916 <attribute name="Format" type="anyURI" use="optional"/>  
917 <attribute name="SPProvidedIdentifier" type="string"  
918 use="optional"/>  
919 </restriction>  
920 </simpleContent>  
921 </complexType>
```

#### 922 2.5.1.4 Element `<EncryptedNameIdentifier>`

923 The `<EncryptedNameIdentifier>` element extends **BaseNameIdentifierAbstractType** to carry the  
924 content of the element in encrypted fashion, as defined by [XMLEnc]. The  
925 `<EncryptedNameIdentifier>` element contains the following additional elements and attributes:

926 `<xenc:EncryptedData>` [Required]

927 The encrypted content and associated encryption details, as defined by [XMLEnc]. The  
928 encrypted content MUST be a `<BaseNameIdentifier>` element or a derivation of it.

929 `<xenc:EncryptedKey>` [Optional]

930 A wrapped decryption key, as defined by [XMLEnc].

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

934 The following schema fragment defines the `<EncryptedNameIdentifier>` element and its  
935 **EncryptedNameIdentifierType** complex type:

```
936 <element name="EncryptedNameIdentifier" type="saml:EncryptedNameIdentifierType"  
937 substitutionGroup="saml:BaseNameIdentifier"/>  
938 <complexType name="EncryptedNameIdentifierType" mixed="false">  
939 <complexContent>  
940 <restriction base="saml:BaseNameIdentifierType">  
941 <sequence>  
942 <element ref="xenc:EncryptedData"/>  
943 <element ref="xenc:EncryptedKey" minOccurs="0"/>  
944 </sequence>  
945 </restriction>  
946 </complexContent>  
947 </complexType>
```

### 948 2.5.1.5 Elements <SubjectConfirmation>, <ConfirmationMethod>, and 949 <SubjectConfirmationData>

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

952 <ConfirmationMethod> [RequiredOne-or-more]

953 A URI reference that identifies a protocol to be used to authenticate the subject. URI references  
954 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the  
955 SAML [profiles specification \[SAMLProf\]](#). ~~Additional methods may be added by defining new URIs~~  
956 ~~and bindings and profiles specification [SAMLBind]~~. ~~Additional methods may be added by defining~~  
957 ~~new~~ profiles or by private agreement.

958 <SubjectConfirmationData> [Optional]

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

960 <ds:KeyInfo> [Optional]

961 An XML Signature [XMLSig] element that ~~identifies a cryptographic key provides access to a~~  
962 ~~cryptographic key held by the subject.~~

963 The following schema fragment defines the <SubjectConfirmation> element and its  
964 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData> element and  
965 the <ConfirmationMethod> element:

```
966 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
967 <complexType name="SubjectConfirmationType">  
968   <sequence>  
969     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
970     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
971     <element ref="ds:KeyInfo" minOccurs="0"/>  
972   </sequence>  
973 </complexType>  
974 <element name="SubjectConfirmationData" type="anyType"/>  
975 <element name="ConfirmationMethod" type="anyURI"/>
```

### 976 2.5.2 Element <AuthenticationStatement>

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

981 AuthenticationMethod [Required]

982 A URI reference that specifies the type of authentication that took place. URI references identifying  
983 common authentication protocols are listed in Section Authentication Method Identifiers. [A value of](#)  
984 [urn:oasis:names:tc:SAML:2.0:am:authncontext](#) [indicates that an <AuthnContext>](#)  
985 [element is included in the statement that describes further details of the authentication.](#)

986 AuthenticationInstant [Required]

987 Specifies the time at which the authentication took place. The time value is encoded in UTC as  
988 described in Section Time Values.

989 <SubjectLocality> [Optional]

990 Specifies the DNS domain name and IP address for the system **entity** from which the subject was  
991 apparently authenticated.



992 | [<AuthnContext>](#) [Optional]

993 | [The context used by the identity provider in the authentication event that yielded this statement.](#)  
994 | [Contains an authentication context statement or a reference to one. Optionally contains a reference](#)  
995 | [to an authentication context class.](#)

996 | **Note:** The <AuthorityBinding> element and its corresponding type were removed  
997 | from <AuthenticationStatement> for V2.0 of SAML.

998 | [<AuthenticationStatement> elements MUST contain a SessionIndex value, conforming to the](#)  
999 | [rules specified in section 2.5.1.](#)

1000 | The following schema fragment defines the <AuthenticationStatement> element and its  
1001 | **AuthenticationStatementType** complex type:

```
1002 | <element name="AuthenticationStatement"  
1003 |         type="saml:AuthenticationStatementType"/>  
1004 | <complexType name="AuthenticationStatementType">  
1005 |   <complexContent>  
1006 |     <extension base="saml:SubjectStatementAbstractType">  
1007 |       <sequence>  
1008 |         <element ref="saml:SubjectLocality" minOccurs="0"/>  
1009 |         <element ref="saml:AuthnContext" minOccurs="0"/>  
1010 |       </sequence>  
1011 |       <attribute name="AuthenticationMethod" type="anyURI"  
1012 | use="required"/>  
1013 |       <attribute name="AuthenticationInstant" type="dateTime"  
1014 | use="required"/>  
1015 |     </extension>  
1016 |   </complexContent>  
1017 | </complexType>
```

### 1018 | 2.5.2.1 Element <SubjectLocality>

1019 | The <SubjectLocality> element specifies the DNS domain name and IP address for the system  
1020 | [from which the subject entity that](#) was authenticated. It has the following attributes:

1021 | IPAddress [Optional]

1022 | The IP address of the system [from which the subject entity that](#) was authenticated.

1023 | DNSAddress [Optional]

1024 | The DNS address of the system [from which the subject entity that](#) was authenticated.

1025 | This element is entirely advisory, since both these fields are quite easily “spoofed,” but current practice  
1026 | appears to require its inclusion.

1027 | The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType**  
1028 | complex type:

```
1029 | <element name="SubjectLocality"  
1030 |         type="saml:SubjectLocalityType"/>  
1031 | <complexType name="SubjectLocalityType">  
1032 |   <attribute name="IPAddress" type="string" use="optional"/>  
1033 |   <attribute name="DNSAddress" type="string" use="optional"/>  
1034 | </complexType>
```

## 1035 **2.5.2.2 Element <AuthnContext>**

1036 The <AuthnContext> element specifies the context of an authentication event with an optional context  
1037 class URI followed by an authentication context statement or statement reference. It's complex  
1038 **AuthnContextType** has the following elements:

1039 <AuthnContextClassRef> [Optional]

1040 A URI identifying an authentication context class that describes the authentication context statement  
1041 that follows.

1042 <AuthnContextStatement> or <AuthnContextStatementRef> [Required]

1043 Either an authentication context statement, or a URI that identifies such a statement. The URI MAY  
1044 directly resolve into an XML document containing the referenced statement.

1045 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**  
1046 complex type:

```
1047 <element name="AuthnContext" type="saml:AuthnContextType"/>
1048 <complexType name="AuthnContextType">
1049   <sequence>
1050     <element ref="saml:AuthnContextClassRef" minOccurs="0"/>
1051     <choice>
1052       <element ref="saml:AuthnContextStatement"/>
1053       <element ref="saml:AuthnContextStatementRef"/>
1054     </choice>
1055   </sequence>
1056 </complexType>
1057 <element name="AuthnContextClassRef" type="anyURI"/>
1058 <element name="AuthnContextStatementRef" type="anyURI"/>
1059 <element name="AuthnContextStatement" type="anyType"/>
```

## 1060 **2.5.3 Element <AttributeStatement>**

1061 The <AttributeStatement> element describes a statement by the SAML authority asserting that the  
1062 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,  
1063 which extends **SubjectStatementAbstractType** with the addition of the following element:

1064 <Attribute> [One or More]

1065 The <Attribute> element specifies an attribute of the subject.

1066 The following schema fragment defines the <AttributeStatement> element and its  
1067 **AttributeStatementType** complex type:

```
1068 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1069 <complexType name="AttributeStatementType">
1070   <complexContent>
1071     <extension base="saml:SubjectStatementAbstractType">
1072       <sequence>
1073         <element ref="saml:Attribute"
1074           maxOccurs="unbounded"/>
1075       </sequence>
1076     </extension>
1077   </complexContent>
1078 </complexType>
```

### 1079 **2.5.3.1 Elements <AttributeDesignator> and <Attribute>**

1080 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It  
1081 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute

1082 values within a specific namespace be returned (see Section Element <AttributeQuery> for more  
1083 information). The <AttributeDesignator> element contains the following XML attributes:

1084 NameAttributeNamespace [Required]

1085 The namespace in which the AttributeName elements are interpreted.

1086 AttributeName [Required]

1087 The name of the attribute.

1088 NameFormat [Required]

1089 A URI reference representing the classification of the attribute name for purposes of interpreting  
1090 the name. See Section 7.x for some URI references that MAY be used as the value of the  
1091 NameFormat attribute and their associated descriptions and processing rules. If no  
1092 NameFormat value is provided, the identifier urn:oasis:names:tc:SAML:2.0:atname-  
1093 format:unspecified (see Section 7.x) is in effect.

1094 ValueType [Optional]

1095 A URI reference representing the datatype of the desired or supplied attribute. If no ValueType  
1096 value is provided, the identifier urn:oasis:names:tc:saml:2.0:valuetype-format:appSpecific (see  
1097 Section 7.x) is in effect. Note that datatypes specified on the <AttributeValue> element  
1098 using xsi:type have no SAML-defined relationship with ValueType. The ValueType setting  
1099 (default or explicit) in an attribute query using the <AttributeDesignator> element MUST be  
1100 exactly matched (in addition to other exact matches as described in Section x) in order for an  
1101 attribute to be returned.

1102 The following schema fragment defines the <AttributeDesignator> element and its  
1103 **AttributeDesignatorType** complex type:

```
1104 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
1105 <complexType name="AttributeDesignatorType">  
1106   <attribute name="AttributeName" type="string" use="required"/>  
1107   <attribute name="NameFormatAttributeNamespace" type="anyURI"  
1108   use="required"/>  
1109   <attribute name="ValueType" type="anyURI" use="optional"/></complexType>  
1110 </complexType>
```

1111 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the  
1112 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following  
1113 element and attributes:

1114 Source [Optional]

1115 The source location or database from which the attribute came. Interpretation of the source  
1116 information is application-specific.

1117 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the  
1118 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the following  
1119 element:

1120 <AttributeValue> [Any Number]

1121 The value of the attribute. If an attribute contains more than one discrete value, it is  
1122 RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute  
1123 exists but has no value, then the <AttributeValue> element MUST be omitted. If more than one  
1124 <AttributeValue> element is supplied for an attribute, and any of the elements have a datatype  
1125 assigned through xsi:type, then all of the <AttributeValue> elements must have the identical  
1126 datatype assigned.

## 1127 [Arbitrary attributes](#)

1128 [This complex type uses an <xsd:anyAttribute> extension point to allow for arbitrary XML](#)  
1129 [attributes to be added to <Attribute> constructs without the need for an explicit schema](#)  
1130 [extension. This allows additional fields to be added as needed to supply the context in which the](#)  
1131 [attribute should be understood. SAML extensions MUST NOT add local \(non-namespace-](#)  
1132 [qualified\) XML attributes to the AttributeType complex type or to any element bound to this type](#)  
1133 [or a derivation of it; such attributes are reserved for future maintenance and enhancement of](#)  
1134 [SAML itself.](#)

1135 The following schema fragment defines the <Attribute> element and its **AttributeType** complex type:

```
1136 <element name="Attribute" type="saml:AttributeType"/>
1137 <complexType name="AttributeType">
1138   <complexContent>
1139     <extension base="saml:AttributeDesignatorType">
1140       <sequence>
1141         <element ref="saml:AttributeValue" minOccurs="0"
1142 maxOccurs="unbounded"/>
1143       </sequence>
1144       <anyAttribute/>
1145     </extension>
1146   </complexContent>
1147 </complexType>
```

### 1148 **2.5.3.1.1 Element <AttributeValue>**

1149 The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType simple**  
1150 type, which allows any well-formed XML to appear as the content of the element.

1151 If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer**  
1152 or **xsd:string**), the data type MAY be declared explicitly by means of an `xsi:type` declaration in the  
1153 <AttributeValue> element. If the attribute value contains structured data, the necessary data  
1154 elements MAY be defined in an extension schema.

1155 [Note: Specifying a datatype on <AttributeValue> using `xsi:type` will require the](#)  
1156 [presence of the extension schema that defines the datatype in order for schema](#)  
1157 [processing to proceed.](#)

1158 The following schema fragment defines the <AttributeValue> element:

```
1159 <element name="AttributeValue" type="anyType"/>
```

### 1160 **2.5.4 Element <AuthorizationDecisionStatement>**

1161 [Note: The <AuthorizationDecisionStatement> feature has been frozen as of](#)  
1162 [SAML V2.0, with no future enhancements planned. Users who require additional](#)  
1163 [functionality may want to consider the eXtensible Access Control Markup Language](#)  
1164 [\[XACML\], which offers enhanced authorization decision features.](#)

1165 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority  
1166 asserting that a request for access by the statement's subject to the specified resource has resulted in  
1167 the specified authorization decision on the basis of some optionally specified evidence.

1168 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
1169 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference  
1170 in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different

1171 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
1172 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1173 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
1174 dependent. When a scheme uses elements of the common syntax, it will also use the common  
1175 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
1176 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
1177 the port is elided.

1178 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the  
1179 URI normalized form wherever possible as follows:

- 1180 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1181 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1182 Inconsistent URI reference interpretation can also result from differences between the URI reference  
1183 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
1184 employed to specify an access control policy language. The following security conditions should be  
1185 satisfied by the system which employs SAML assertions:

- 1186 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
1187 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
1188 part of the resource URI reference.
- 1189 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
1190 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
1191 gain access to a denied resource by creating such an equivalence.

1192 The `<AuthorizationDecisionStatement>` element is of type  
1193 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the  
1194 addition of the following elements (in order) and attributes:

1195 **Resource** [Required]

1196 A URI reference identifying the resource to which access authorization is sought. It is permitted for  
1197 this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the  
1198 start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1199 **Decision** [Required]

1200 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
1201 the **DecisionType** simple type.

1202 **<Action>** [One or more]

1203 The set of actions authorized to be performed on the specified resource.

1204 **<Evidence>** [Optional]

1205 A set of assertions that the SAML authority relied on in making the decision.

1206 The following schema fragment defines the `<AuthorizationDecisionStatement>` element and its  
1207 **AuthorizationDecisionStatementType** complex type:

```
1208 <element name="AuthorizationDecisionStatement"  
1209 type="saml:AuthorizationDecisionStatementType"/>  
1210 <complexType name="AuthorizationDecisionStatementType">  
1211 <complexContent>  
1212 <extension base="saml:SubjectStatementAbstractType">  
1213 <sequence>  
1214 <element ref="saml:Action" maxOccurs="unbounded"/>  
1215 <element ref="saml:Evidence" minOccurs="0"/>
```

```

1216         </sequence>
1217         <attribute name="Resource" type="anyURI" use="required"/>
1218         <attribute name="Decision" type="saml:DecisionType"
1219 use="required"/>
1220     </extension>
1221 </complexContent>
1222 </complexType>

```

### 1223 2.5.4.1 Element <Action>

1224 The <Action> element specifies an action on the specified resource for which permission is sought. It  
1225 has the following attribute and string-data content:

1226 Namespace [Optional]

1227 A URI reference representing the namespace in which the name of the specified action is to be  
1228 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwe-dc-  
1229 negotiation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

1230 *string data* [Required]

1231 An action sought to be performed on the specified resource.

1232 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```

1233 <element name="Action" type="saml:ActionType"/>
1234 <complexType name="ActionType">
1235     <simpleContent>
1236         <extension base="string">
1237             <attribute name="Namespace" type="anyURI"/>
1238         </extension>
1239     </simpleContent>
1240 </complexType>

```

### 1241 2.5.4.2 Element <Evidence>

1242 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on  
1243 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one  
1244 or more of the following elements:

1245 <AssertionIDReference> [Any number]

1246 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1247 [<AssertionURIReference> \[Any number\]](#)

1248 [Specifies an assertion by reference to a URI.](#)

1249 <Assertion> [Any number]

1250 Specifies an assertion by value.

1251 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
1252 and the SAML authority making the authorization decision. For example, in the case that the SAML  
1253 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use  
1254 that assertion as evidence in making its authorization decision without endorsing the <Evidence>  
1255 element's assertion as valid either to the relying party or any other third party.

1256 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```

1257 <element name="Evidence" type="saml:EvidenceType"/>
1258 <complexType name="EvidenceType">
1259     <choice maxOccurs="unbounded">

```

1260  
1261  
1262  
1263  
1264

```
<element ref="saml:AssertionIDReference"/>  
<element ref="saml:AssertionURIReference"/>  
<element ref="saml:Assertion"/>  
</choice>  
</complexType>
```

## 3 SAML Protocols

1265

1266 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of  
1267 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting  
1268 queries, assertions, and other messages using existing widely deployed transport~~MAY be generated and~~  
1269 ~~exchanged using a variety of protocols. The bindings and profiles specification for SAML [SAMLBind]~~  
1270 ~~describes specific means of transporting assertions using existing widely deployed protocols.~~

1271 Specific SAML request and response messages derive from common types. The requester sends an  
1272 element derived from RequestAbstractType to a SAML responder, and the responder generates an  
1273 element adhering to or deriving from StatusResponseType~~SAML-aware requesters MAY in addition~~  
1274 ~~use the SAML request-response protocol defined by the <Request> and <Response> elements.~~  
1275 ~~The requester sends a <Request> element to a SAML responder, and the responder generates a~~  
1276 ~~<Response> element, as shown in Figure 1.~~

1277



1279

Figure 1: SAML Request-Response Protocol

1280 The protocols defined by SAML are as follows:

- 1281 • Assertion request (includes a direct request of the desired assertions, as well as querying for  
1282 assertions that meet particular criteria)
- 1283 • Request for authentication to be performed
- 1284 • Request to register a federated name
- 1285 • Request to retrieve a protocol message by means of an artifact
- 1286 • Request to terminate a federated name registration
- 1287 • Request for a near-simultaneous logout of a collection of related sessions ("single logout")
- 1288 • Request a name identifier mapping

1289

### 3.1 Schema Header and Namespace Declarations

1290 The following schema fragment defines the XML namespaces and other header information for the  
1291 protocol schema:

```
1292 <schema  
1293   targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"  
1294   xmlns="http://www.w3.org/2001/XMLSchema"  
1295   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1296   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1297   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1298   elementFormDefault="unqualified"  
1299   attributeFormDefault="unqualified"  
1300   blockDefault="substitution"
```



```

1301 |         version="2.0">
1302 |         <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1303 |             schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
1304 |         <import namespace="http://www.w3.org/2000/09/xmlsig#"
1305 |             schemaLocation="http://www.w3.org/TR/xmlsig-core-
1306 | schema.xsd"/>
1307 |         <annotation>
1308 |             <documentation>
1309 |                 Document identifier: sstc-saml-schema-protocol-2.0
1310 |                 Location: http://www.oasis-
1311 | open.org/committees/documents.php?wg_abbrev=security
1312 | Revision history:
1313 | Updated the schema and namespace to V2.0.
1314 | Removed <RespondWith> and its corresponding type.
1315 |             </documentation>
1316 |         </annotation>
1317 |         ...
1318 |     </schema>

```

## 1319 | 3.2 Requests and Responses

1320 | The following sections define the SAML constructs that underlie request and response messages contain  
1321 | request information.

### 1322 | 3.2.1 Complex Type RequestAbstractType

1323 | All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
1324 | This type defines common attributes and elements that are associated with all SAML requests:

1325 | RequestID [Required]

1326 | An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in  
1327 | Section 1.2.3 for identifier uniqueness. The values of the RequestID attribute in a request and the  
1328 | InResponseTo attribute in the corresponding response MUST match.

1329 | MajorVersion [Required]

1330 | The major version of this request. The identifier for the version of SAML defined in this specification  
1331 | is 2±. SAML versioning is discussed in Section SAML Versioning.

1332 | MinorVersion [Required]

1333 | The minor version of this request. The identifier for the version of SAML defined in this specification  
1334 | is 0±. SAML versioning is discussed in Section SAML Versioning.

1335 | IssueInstant [Required]

1336 | The time instant of issue of the request. The time value is encoded in UTC as described in Section  
1337 | Time Values.

1338 | Consent [Optional]

1339 | Indicates whether or not consent has been obtained from a user in the sending this request.

1340 | <Issuer> [Optional]

1341 | Identifies the entity that generated the request message.

1342 | <ds:Signature> [Optional]

1343 | An XML Signature that authenticates the request, as described in Section SAML and XML Signature  
1344 | Syntax and Processing.

1345 <RelayState> [Optional]

1346 This contains state information that MUST be relayed back in the associated response.

1347 <Extensions> [Optional]

1348 This contains optional protocol message extension elements that are agreed upon between the  
1349 communicating parties.

1350 **Note:** The <RespondWith> element has been removed from <Request> for V2.0 of  
1351 SAML.

1352 The following schema fragment defines the **RequestAbstractType** complex type:

```
1353 <complexType name="RequestAbstractType" abstract="true">
1354   <sequence>
1355     <element ref="saml:Issuer" minOccurs="0"/>
1356     <element ref="ds:Signature" minOccurs="0"/>
1357     <element ref="samlp:RelayState" minOccurs="0"/>
1358     <element ref="samlp:Extensions" minOccurs="0"/>
1359   </sequence>
1360   <attribute name="RequestID" type="ID" use="required"/>
1361   <attribute name="MajorVersion" type="integer" use="required"/>
1362   <attribute name="MinorVersion" type="integer" use="required"/>
1363   <attribute name="IssueInstant" type="dateTime" use="required"/>
1364   <attribute name="Consent" type="anyURI" use="optional"/>
1365 </complexType>
1366 <element name="Extensions" type="samlp:ExtensionsType"/>
1367 <complexType name="ExtensionsType">
1368   <sequence>
1369     <any namespace="##any" processContents="lax"
1370     maxOccurs="unbounded"/>
1371   </sequence>
1372 </complexType>
```

### 1373 **3.2.1.1 Element <RelayState>**

1374 SAML requests MAY contain a string-valued element containing state information that the requester  
1375 wishes the responder to include in the response. This is particularly useful with asynchronous bindings  
1376 of protocol messages, such as the encoding of messages in browser URLs. This data SHOULD be  
1377 integrity-protected by the requester and MAY have other protections placed on it by the requester, such  
1378 as confidentiality. The length of this value SHOULD be kept as short as possible because of limitations  
1379 of the bindings in which it may be needed.

1380 The following schema fragment defines the <RelayState> element:

```
1381 <element name="RelayState" type="string"/>
```

### 1382 **3.2.2 Complex Type StatusResponseType**

1383 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This  
1384 type defines common attributes and elements that are associated with all SAML responses:

1385 ResponseID [Required]

1386 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
1387 Section 1.2.3 for identifier uniqueness.

1388 InResponseTo [Optional]

1389 A reference to the identifier of the request to which the response corresponds, if any. If the response  
1390 is not generated in response to a request, or if the RequestID attribute value of a request cannot be

1391 determined (because the request is malformed), then this attribute MUST NOT be present.  
 1392 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
 1393 RequestID attribute value.

1394 MajorVersion [Required]  
 1395 The major version of this response. The identifier for the version of SAML defined in this  
 1396 specification is 2. SAML versioning is discussed in Section SAML Versioning.

1397 MinorVersion [Required]  
 1398 The minor version of this response. The identifier for the version of SAML defined in this  
 1399 specification is 0. SAML versioning is discussed in Section SAML Versioning.

1400 IssueInstant [Required]  
 1401 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
 1402 Time Values.

1403 Recipient [Optional]  
 1404 The intended recipient of this response. This is useful to prevent malicious forwarding of responses  
 1405 to unintended recipients, a protection that is required by some use profiles. It is set by the generator  
 1406 of the response to a URI reference that identifies the intended recipient. If present, the actual  
 1407 recipient MUST check that the URI reference identifies the recipient or a resource managed by the  
 1408 recipient. If it does not, the response MUST be discarded.

1409 <Issuer> [Optional]  
 1410 Identifies the entity that generated the response message.

1411 <ds:Signature> [Optional]  
 1412 An XML Signature that authenticates the response, as described in Section SAML and XML  
 1413 Signature Syntax and Processing.

1414 <RelayState> [Optional]  
 1415 This contains state information from the associated request being relayed back in the response. It  
 1416 MUST match the <RelayState> value in the associated request, if any.

1417 <Extensions> [Optional]  
 1418 This contains optional protocol message extension elements that are agreed upon between the  
 1419 communicating parties.

1420 <Status> [Required]  
 1421 A code representing the status of the corresponding request.

1422 The following schema fragment defines the **StatusResponseType** complex type:

```

1423 <complexType name="StatusResponseType">
1424   <sequence>
1425     <element ref="saml:Issuer" minOccurs="0"/>
1426     <element ref="ds:Signature" minOccurs="0"/>
1427     <element ref="samlp:RelayState" minOccurs="0"/>
1428     <element ref="samlp:Extensions" minOccurs="0"/>
1429     <element ref="samlp:Status"/>
1430   </sequence>
1431   <attribute name="ResponseID" type="ID" use="required"/>
1432   <attribute name="InResponseTo" type="NCName" use="optional"/>
1433   <attribute name="MajorVersion" type="integer" use="required"/>
1434   <attribute name="MinorVersion" type="integer" use="required"/>
1435   <attribute name="IssueInstant" type="dateTime" use="required"/>
1436   <attribute name="Recipient" type="anyURI" use="optional"/>

```

1437 | `</complexType>`

### 1438 | **3.2.2.1 Element <Status>**

1439 | The <Status> element contains the following elements:

1440 | <StatusCode> [Required]

1441 | A code representing the status of the corresponding request.

1442 | <StatusMessage> [Optional]

1443 | A message which MAY be returned to an operator.

1444 | <StatusDetail> [Optional]

1445 | Additional information concerning an error condition.

1446 | The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1447 | <element name="Status" type="samlp:StatusType"/>  
1448 | <complexType name="StatusType">  
1449 |   <sequence>  
1450 |     <element ref="samlp:StatusCode"/>  
1451 |     <element ref="samlp:StatusMessage" minOccurs="0"/>  
1452 |     <element ref="samlp:StatusDetail" minOccurs="0"/>  
1453 |   </sequence>  
1454 | </complexType>
```

### 1455 | **3.2.2.2 Element <StatusCode>**

1456 | The <StatusCode> element specifies one or more possibly nested, codes representing the status of the  
1457 | corresponding request. The <StatusCode> element has the following element and attribute:

1458 | Value [Required]

1459 | The status code value. This attribute contains an XML Schema QName; a namespace prefix **MUST**  
1460 | be provided. The value of the topmost <StatusCode> element **MUST** be from the top-level list  
1461 | provided in this section.

1462 | <StatusCode> [Optional]

1463 | A subordinate status code that provides more specific information on an error condition.

1464 | The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
1465 | local parts of these QNames are as follows:

1466 | Success

1467 | The request succeeded.

1468 | VersionMismatch

1469 | The SAML responder could not process the request because the version of the request message  
1470 | was incorrect.

1471 | Requester

1472 | The request could not be performed due to an error on the part of the requester.

1473 | Responder

1474 | The request could not be performed due to an error on the part of the SAML responder or SAML  
1475 | authority.

1476 | The following second-level status codes are referenced at various places in the specification. Additional  
1477 | second-level status codes MAY be defined in future versions of the SAML specification.

1478 | RequestVersionTooHigh

1479 | The SAML responder cannot process the request because the protocol version specified in the  
1480 | request message is a major upgrade from the highest protocol version supported by the responder.

1481 | RequestVersionTooLow

1482 | The SAML responder cannot process the request because the protocol version specified in the  
1483 | request message is too low.

1484 | RequestVersionDeprecated

1485 | The SAML responder can not process any requests with the protocol version specified in the  
1486 | request.

1487 | TooManyResponses

1488 | The response message would contain more elements than the SAML responder will return.

1489 | RequestDenied

1490 | The SAML responder or SAML authority is able to process the request but has chosen not to  
1491 | respond. This status code MAY be used when there is concern about the security context of the  
1492 | request message or the sequence of request messages received from a particular requester.

1493 | RequestUnsupported

1494 | The SAML responder or SAML authority does not support the request.

1495 | ResourceNotRecognized

1496 | The SAML authority does not wish to support resource-specific attribute queries, or the resource  
1497 | value provided in the request message is invalid or unrecognized.

1498 | FederationDoesNotExist

1499 | The responding provider does not recognize the federated <NameIdentifier> in the request.

1500 | UnknownPrincipal

1501 | The responding provider does not recognize the principal specified or implied by the request.

1502 | NoAuthnContext

1503 | The specified authentication context requirements cannot be met by the responder.

1504 | InvalidNameIDPolicy

1505 | The responding provider does not support the specified name identifier format for the requested  
1506 | subject.

1507 | NoPassive

1508 | Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1509 | ProxyCountExceeded

1510 | Indicates that an identity provider cannot authenticate the principal directly and is not permitted to  
1511 | proxy the request further.

1512 | NoAvailableIDP

1513 | Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in  
1514 | an <IDPList> can be resolved or that none of the supported identity providers are available.

1515 [NoSupportedIDP](#)

1516 [Used by an intermediary to indicate that none of the identity providers in an <IDPList> are](#)  
1517 [supported by the intermediary.](#)

1518 [SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT](#)  
1519 [define additional codes in the SAML assertion or protocol namespace.](#)

1520 [The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value](#)  
1521 [attribute and have the above semantics only in that context.](#)

1522 [The following schema fragment defines the <StatusCode> element and its \*\*StatusCodeType\*\* complex](#)  
1523 [type:](#)

```
1524 <element name="StatusCode" type="samlp:StatusCodeType"/>
1525 <complexType name="StatusCodeType">
1526   <sequence>
1527     <element ref="samlp:StatusCode" minOccurs="0"/>
1528   </sequence>
1529   <attribute name="Value" type="OName" use="required"/>
1530 </complexType>
```

### 1531 **3.2.2.3 Element <StatusMessage>**

1532 [The <StatusMessage> element specifies a message that MAY be returned to an operator:](#)

1533 [The following schema fragment defines the <StatusMessage> element:](#)

```
1534 <element name="StatusMessage" type="string"/>
```

### 1535 **3.2.2.4 Element <StatusDetail>**

1536 [The <StatusDetail> element MAY be used to specify additional information concerning an error](#)  
1537 [condition.](#)

1538 [The following schema fragment defines the <StatusDetail> element and its \*\*StatusDetailType\*\*](#)  
1539 [complex type:](#)

```
1540 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1541 <complexType name="StatusDetailType">
1542   <sequence>
1543     <any namespace="##any" processContents="lax" minOccurs="0"
1544     maxOccurs="unbounded"/>
1545   </sequence>
1546 </complexType>
```

## 1547 **3.3 Assertion Query and Request Protocol**

1548 [This section defines messages and processing rules for requesting existing assertions by reference or](#)  
1549 [querying for assertions by subject and statement type.](#)

### 1550 **3.3.1 Element <AssertionIDRequest>**

1551 [If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>](#)  
1552 [message can be used to request that the assertion\(s\) be returned in a <Response> message. The](#)  
1553 [<saml:AssertionIDReference> element is used to specify the assertion\(s\) to return. See Section](#)  
1554 [Element <AssertionIDReference> for more information on this element.](#)

1555 [The following schema fragment defines the <AssertionIDRequest> element:](#)

```
1556 | <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>  
1557 | <complexType name="AssertionIDRequestType">
```

### 1558 | 3.3.2 Element <Request>

1559 | The <Request> element specifies a SAML request. It provides either a query or a request for a specific  
1560 | assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has the complex  
1561 | type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following  
1562 | elements:

1563 | <Query>

1564 | An extension point that allows extension schemas to define new types of query.

1565 | <SubjectQuery>

1566 | An extension point that allows extension schemas to define new types of query that specify a single  
1567 | SAML subject.

1568 | <AuthenticationQuery>

1569 | Makes a query for authentication information.

1570 | <AttributeQuery>

1571 | Makes a query for attribute information.

1572 | <AuthorizationDecisionQuery>

1573 | Makes a query for an authorization decision.

1574 | <AssertionIDReference> [One or more]

1575 | Requests an assertion by reference to the value of its `AssertionID` attribute.

1576 | <AssertionArtifact> [One or more]

1577 | Requests assertions by supplying an assertion artifact that represents it.

1578 | The following schema fragment defines the <Request> element and its **RequestType** complex type:

```
1579 | <element name="Request" type="samlp:RequestType"/>  
1580 | <complexType name="RequestType">  
1581 | <complexContent>  
1582 |   <extension base="samlp:RequestAbstractType">  
1583 |     <sequencechoice>  
1584 |       <element ref="samlp:Query"/>  
1585 |       <element ref="samlp:SubjectQuery"/>  
1586 |       <element ref="samlp:AuthenticationQuery"/>  
1587 |       <element ref="samlp:AttributeQuery"/>  
1588 |       <element ref="samlp:AuthorizationDecisionQuery"/>  
1589 |       <element ref="saml:AssertionIDReference"  
1590 | maxOccurs="unbounded"/>  
1591 |     </sequence <element ref="samlp:AssertionArtifact"  
1592 | maxOccurs="unbounded"/>  
1593 |   </choice>  
1594 | </extension>  
1595 | </complexContent>  
1596 | </complexType>
```

### 1597 | **3.3.2.1 Requests for Assertions by Reference**

1598 | ~~In the context of a <Request> element, the <saml:AssertionIDReference> element is used to~~  
1599 | ~~request an assertion by means of its ID. See Section Element <AssertionIDReference> for more~~  
1600 | ~~information on this element.~~

### 1601 | **3.3.2.2 Element <AssertionArtifact>**

1602 | ~~The <AssertionArtifact> element is used to specify the assertion artifact that represents an~~  
1603 | ~~assertion being requested. Its use is governed by the specific profile of SAML that is being used; see the~~  
1604 | ~~SAML specification for bindings and profiles [SAMLBind] for more information on the use of assertion~~  
1605 | ~~artifacts in profiles.~~

1606 | ~~The following schema fragment defines the <AssertionArtifact> element:~~

```
1607 | <element name="AssertionArtifact" type="string"/>
```

### 1608 | **3.3.3 Queries**

1609 | ~~The following sections define the SAML query request messages constructs that contain query~~  
1610 | ~~information.~~

### 1611 | **3.3.4 Element <Query>**

1612 | ~~The <Query> element is an extension point that allows new SAML queries to be defined. Its~~  
1613 | ~~**QueryAbstractType** is abstract and is thus usable only as the base of a derived type.~~  
1614 | ~~**QueryAbstractType** is the base type from which all SAML query elements are derived.~~

1615 | ~~The following schema fragment defines the <Query> element and its **QueryAbstractType** complex type:~~

```
1616 | <element name="Query" type="saml:QueryAbstractType"/>  
1617 | <complexType name="QueryAbstractType" abstract="true"/>
```

### 1618 | **3.3.4.1 Element <SubjectQuery>**

1619 | ~~The <SubjectQuery> message element is an extension point that allows new SAML queries to be~~  
1620 | ~~defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and~~  
1621 | ~~is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>~~  
1622 | ~~element and an optional SessionIndex attribute to **RequestAbstractType** **element is an extension**~~  
1623 | ~~**point that allows new SAML queries that specify a single SAML subject. Its**~~  
1624 | ~~**SubjectQueryAbstractType** complex type is abstract and is thus usable only as the base of a~~  
1625 | ~~**derived type. SubjectQueryAbstractType** adds the <Subject> element.~~

1626 | SessionIndex [Optional]

1627 | ~~If present, specifies a filter for possible responses. Such a query asks the question "What assertions~~  
1628 | ~~containing subject statements do you have for this subject within the context of the supplied session~~  
1629 | ~~information?"~~

1630 | ~~If the SessionIndex attribute is present in any defined query, at least one element that extends~~  
1631 | ~~**StatementAbstractType** in the set of returned assertions MUST contain an SessionIndex attribute~~  
1632 | ~~that matches the SessionIndex attribute in the query. It is OPTIONAL for the complete set of all such~~  
1633 | ~~matching assertions to be returned in the response.~~

1634 | ~~The following schema fragment defines the <SubjectQuery> element and its~~  
1635 | ~~**SubjectQueryAbstractType** complex type:~~



```

1636 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>
1637 <complexType name="SubjectQueryAbstractType" abstract="true">
1638   <complexContent>
1639     <extension base="saml:RequestQueryAbstractType">
1640       <sequence>
1641         <element ref="saml:Subject"/>
1642       </sequence>
1643       <attribute name="SessionIndex" type="string"
1644         use="optional"/>
1645     </extension>
1646   </complexContent>
1647 </complexType>

```

### 1648 3.3.4.2 Element <AuthenticationQuery>

1649 The <AuthenticationQuery> message element is used to make the query “What assertions  
1650 containing authentication statements are available for this subject?” A successful <Response> will  
1651 contain one or moreelement is used to make the query “What assertions containing authentication  
1652 statements are available for this subject?” A successful response will be in the form of assertions  
1653 containing authentication statements.

1654 The <AuthenticationQuery> message element MUST NOT be used as a request for a new  
1655 authentication using credentials provided in the request. <AuthenticationQuery> is a request for  
1656 statements about authentication acts that have occurred in a previous interaction between the indicated  
1657 subject and the Authentication Authority.

1658 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the  
1659 addition of the following attribute:

1660 AuthenticationMethod [Optional]

1661 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1662 containing authentication statements do you have for this subject with the supplied authentication  
1663 method?”

1664 In response to an authentication query, a SAML authority returns assertions with authentication  
1665 statements as follows:

- 1666 • Rules given in Section Responses to for matching against the <Subject> element of the query  
1667 identify the assertions that may be returned.
- 1668 • If the AuthenticationMethod attribute is present in the query, at least one  
1669 <AuthenticationStatement> element in the set of returned assertions MUST contain an  
1670 AuthenticationMethod attribute that matches the AuthenticationMethod attribute in  
1671 the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in  
1672 the response.

1673 The following schema fragment defines the <AuthenticationQuery> element and its  
1674 **AuthenticationQueryType** complex type:

```

1675 <element name="AuthenticationQuery" type="saml:AuthenticationQueryType"/>
1676 <complexType name="AuthenticationQueryType">
1677   <complexContent>
1678     <extension base="saml:SubjectQueryAbstractType">
1679       <attribute name="AuthenticationMethod" type="anyURI"/>
1680     </extension>
1681   </complexContent>
1682 </complexType>

```

### 1683 3.3.4.3 Element <AttributeQuery>

1684 The <AttributeQuery> element is used to make the query "Return the requested attributes for this  
1685 subject." A successful response will be in the form of assertions containing attribute statements. This  
1686 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1687 the following element and attribute:

1688 Resource [Optional]

1689 If present, specifies that the attribute query is being made in order to evaluate a specific access  
1690 request relating to the resource. The SAML authority MAY use the resource attribute to establish the  
1691 scope of the request. It is permitted for this attribute to have the value of the empty URI reference  
1692 (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]  
1693 §4.2.

1694 If the resource attribute is specified and the SAML authority does not wish to support resource-  
1695 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the  
1696 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a  
1697 second-level <StatusCode> value of ResourceNotRecognized.

1698 <AttributeDesignator> [Any Number] ~~(see Section Elements <AttributeDesignator> and~~  
1699 ~~<Attribute>)~~

1700 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no  
1701 attributes are specified, it indicates that all attributes allowed by policy are requested.

1702 In response to an attribute query, a SAML authority returns assertions with attribute statements as  
1703 follows:

- 1704 • Rules given in Section Responses to for matching against the <Subject> element of the query  
1705 identify the assertions that may be returned.
- 1706 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute  
1707 values returned, as noted above.
- 1708 • The SAML authority MAY take the Resource attribute into account in further constraining the values  
1709 returned, as noted above.
- 1710 • The attribute values returned MAY be constrained by application-specific policy considerations.

1711 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**  
1712 complex type:

```
1713 <element name="AttributeQuery" type="saml:AttributeQueryType"/>  
1714 <complexType name="AttributeQueryType">  
1715   <complexContent>  
1716     <extension base="saml:SubjectQueryAbstractType">  
1717       <sequence>  
1718         <element ref="saml:AttributeDesignator"  
1719           minOccurs="0" maxOccurs="unbounded"/>  
1720       </sequence>  
1721       <attribute name="Resource" type="anyURI" use="optional"/>  
1722     </extension>  
1723   </complexContent>  
1724 </complexType>
```

### 1725 3.3.4.4 Element <AuthorizationDecisionQuery>

1726 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on  
1727 this resource be allowed for this subject, given this evidence?" A successful response will be in the form

1728 of assertions containing authorization decision statements. ~~This element is of type~~  
1729 ~~**AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the~~  
1730 ~~following elements and attribute:~~

1731 **Note:** The `<AuthorizationDecisionQuery>` feature has been frozen as of SAML  
1732 V2.0, with no future enhancements planned. Users who require additional functionality  
1733 may want to consider the eXtensible Access Control Markup Language [XACML], which  
1734 offers enhanced authorization decision features.

1735 This element is of type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType**  
1736 with the addition of the following elements and attribute:

1737 Resource [Required]

1738 A URI reference indicating the resource for which authorization is requested.

1739 `<Action>` [One or More]

1740 The actions for which authorization is requested.

1741 `<Evidence>` [Optional]

1742 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1743 In response to an authorization decision query, a SAML authority returns assertions with authorization  
1744 decision statements as follows:

- 1745 • Rules given in Section ~~3.3.4.1 Responses to~~ for matching against the `<Subject>` element of the  
1746 query identify the assertions that may be returned.

1747 The following schema fragment defines the `<AuthorizationDecisionQuery>` element and its  
1748 **AuthorizationDecisionQueryType** complex type:

```
1749 <element name="AuthorizationDecisionQuery"  
1750 type="saml:AuthorizationDecisionQueryType"/>  
1751 <complexType name="AuthorizationDecisionQueryType">  
1752   <complexContent>  
1753     <extension base="saml:SubjectQueryAbstractType">  
1754       <sequence>  
1755         <element ref="saml:Action" maxOccurs="unbounded"/>  
1756         <element ref="saml:Evidence" minOccurs="0"/>  
1757       </sequence>  
1758       <attribute name="Resource" type="anyURI" use="required"/>  
1759     </extension>  
1760   </complexContent>  
1761 </complexType>
```

## 1762 3.4 Responses

1763 The following sections define the SAML constructs that contain response information.

### 1764 3.4.1 Complex-Type ResponseAbstractType

1765 All SAML responses are of types that are derived from the abstract **ResponseAbstractType** complex  
1766 type. This type defines common attributes and elements that are associated with all SAML responses:

1767 ResponseID [Required]

1768 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
1769 Section 1.2.3 for identifier uniqueness.

1770 **InResponseTo** [Optional]  
 1771 A reference to the identifier of the request to which the response corresponds, if any. If the response  
 1772 is not generated in response to a request, or if the RequestID attribute value of a request cannot be  
 1773 determined (because the request is malformed), then this attribute MUST NOT be present.  
 1774 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
 1775 RequestID attribute value.

1776 **MajorVersion** [Required]  
 1777 The major version of this response. The identifier for the version of SAML defined in this  
 1778 specification is 1. SAML versioning is discussed in Section SAML Versioning.

1779 **MinorVersion** [Required]  
 1780 The minor version of this response. The identifier for the version of SAML defined in this  
 1781 specification is 1. SAML versioning is discussed in Section SAML Versioning.

1782 **IssueInstant** [Required]  
 1783 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
 1784 Time Values.

1785 **Recipient** [Optional]  
 1786 The intended recipient of this response. This is useful to prevent malicious forwarding of responses  
 1787 to unintended recipients, a protection that is required by some use profiles. It is set by the generator  
 1788 of the response to a URI reference that identifies the intended recipient. If present, the actual  
 1789 recipient MUST check that the URI reference identifies the recipient or a resource managed by the  
 1790 recipient. If it does not, the response MUST be discarded.

1791 **<ds:Signature>** [Optional]  
 1792 An XML Signature that authenticates the response, as described in Section SAML and XML  
 1793 Signature Syntax and Processing.

1794 The following schema fragment defines the **ResponseAbstractType** complex type:

```

1795 <complexType name="ResponseAbstractType" abstract="true">
1796   <sequence>
1797     <element ref="ds:Signature" minOccurs="0"/>
1798   </sequence>
1799   <attribute name="ResponseID" type="ID" use="required"/>
1800   <attribute name="InResponseTo" type="NCName" use="optional"/>
1801   <attribute name="MajorVersion" type="integer" use="required"/>
1802   <attribute name="MinorVersion" type="integer" use="required"/>
1803   <attribute name="IssueInstant" type="dateTime" use="required"/>
1804   <attribute name="Recipient" type="anyURI" use="optional"/>
1805 </complexType>

```

### 1806 3.4.2 Element <Response>

1807 The <Response> message element is used when a response consists of a list of zero or more  
 1808 assertions that answer the request. It has the complex type **ResponseType**, which extends  
 1809 **StatusResponseType** by adding the following elementelement specifies the status of the corresponding  
 1810 SAML request and a list of zero or more assertions that answer the request. It has the complex type  
 1811 **ResponseType**, which extends **ResponseAbstractType** by adding the following elements in order:

1812 **<Status>** [Required]

1813 A code representing the status of the corresponding request.

1814 | <Assertion> [Any Number]

1815 | Specifies an assertion by value. (See Section Element <Assertion> for more information.)

1816 | The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1817 | <element name="Response" type="samlp:ResponseType"/>
1818 | <complexType name="ResponseType">
1819 |   <complexContent>
1820 |     <extension base="samlp:StatusResponseResponseAbstractType">
1821 |       <sequence>
1822 |         <del><element ref="samlp:Status"/></del>
1823 |         <element ref="saml:Assertion" minOccurs="0"
1824 | maxOccurs="unbounded"/>
1825 |       </sequence>
1826 |     </extension>
1827 |   </complexContent>
1828 | </complexType>
```

### 1829 | 3.4.2.1 Processing Rules

1830 | In response to a query message, every assertion returned by a SAML authority MUST contain a  
1831 | <Subject> element that strongly matches the <Subject> element found in the query.

1832 | A <Subject> element S1 strongly matches S2 if and only if the following two conditions both apply:

- 1833 | • If S2 includes an identifier element (any element whose type is derived from  
1834 | BaseIdentifierAbstractType), then S1 must include an identical identifier element.
- 1835 | • If S2 includes one or more <SubjectConfirmation> elements, then S1 must include at least one  
1836 | <SubjectConfirmation> element such that the assertion's subject can be confirmed in the  
1837 | manner described by at least one element in the requested set.

1838 | If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
1839 | expressed by a query, the <Response> element MUST NOT contain an <Assertion> element and  
1840 | MUST include a <StatusCode> element with value Success. It MAY return a <StatusMessage>  
1841 | element with additional information.

## 1842 | 3.5 Authentication Request Protocol

1843 | When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing  
1844 | authentication statements to establish a security context at one or more relying parties, it can use the  
1845 | authentication request protocol to send an <AuthnRequest> message to a SAML authority and request  
1846 | that it return a <Response> message containing one or more such assertions. A SAML authority that  
1847 | supports this protocol is also termed an identity provider. Such assertions MAY contain additional  
1848 | statements of any type, but at least one assertion MUST contain at least one authentication statement.

1849 | Apart from this requirement, the specific contents of the returned assertions depend on the profile or  
1850 | context of use. Also, the exact means by which the principal or agent authenticates to the identity  
1851 | provider are not specified, though the means of authentication MAY impact the content of the response.  
1852 | Other issues related to the validation of authentication credentials by the identity provider or any  
1853 | communication between the identity provider and any other entities involved in the authentication  
1854 | process are also out of scope of this protocol.

1855 | The descriptions and processing rules in the following sections reference the following actors, many of  
1856 | whom might be the same entity in a particular profile of use:

1857 | Request Issuer

1858 | The entity who creates the authentication request and to whom the response is to be returned.

- 1859 | Presenter
- 1860 | The entity who presents the request to the authority and either authenticates itself during the  
1861 | sending of the message, or relies on an existing security context to establish its identity. If not  
1862 | the request issuer, the sender acts as an intermediary between the request issuer and the  
1863 | responding identity provider.
- 1864 | Requested Subject
- 1865 | The entity about whom one or more assertions are being requested.
- 1866 | Confirming Subject
- 1867 | The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>  
1868 | elements of the resulting assertion(s).
- 1869 | Relying Party
- 1870 | The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1871 | the profile or context of use, generally to establish a security context.

### 1872 | **3.5.1 Element <AuthnRequest>**

- 1873 | To request that an identity provider issue an authentication assertion, an entity authenticates to it (or  
1874 | relies on an existing security context) and sends it an <AuthnRequest> message that describes the  
1875 | properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may  
1876 | be information that relates to the content of the assertion and/or information that relates to how the  
1877 | resulting <Response> message should be delivered to the request issuer.
- 1878 | The request issuer might not be the same as the presenter of the request, if for example the request  
1879 | issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the  
1880 | requested subject to provide a service.
- 1881 | The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected  
1882 | by the protocol binding used to deliver the message.
- 1883 | This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
1884 | adds the following elements and attributes, all of which are optional in general, but may be required by  
1885 | specific profiles:
- 1886 | <Subject> [Optional]
- 1887 | Specifies the requested subject of the resulting assertion(s). This may include one or more  
1888 | <SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions' can  
1889 | be confirmed.
- 1890 | If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
1891 | requested subject. If no <SubjectConfirmation> elements are included, then the presenter is  
1892 | presumed to be the only confirming entity required and the method is implied by the profile of use  
1893 | and/or the policies of the identity provider.
- 1894 | <NameIDPolicy> [Optional]
- 1895 | Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,  
1896 | then any type of identifier supported by the identity provider for the requested subject can be used,  
1897 | constrained by any relevant deployment-specific policies, with respect to privacy, for example.
- 1898 |
- 1899 | <Conditions> [Optional]
- 1900 | Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the

1901 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary.

1902 <RequestAuthnContext> [Optional]

1903 Specifies the requirements, if any, that the request issuer places on the authentication context that  
 1904 applies to the responding provider's authentication of the presenter.

1905 <Scoping> [Optional]

1906 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as  
 1907 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity  
 1908 providers by the responder.

1909 IsPassive [Optional]

1910 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of  
 1911 the user interface from the request issuer and interact with the presenter in a noticeable fashion. If a  
 1912 value is not provided, the default is "true".

1913 ForceAuthn [Optional]

1914 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
 1915 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
 1916 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the  
 1917 presenter unless the constraints of IsPassive can be met.

1918 ProtocolBinding [Optional]

1919 A URI that identifies a SAML protocol binding to be used when returning the <Response> message.

1920 AssertionConsumerServiceID [Optional]

1921 References one of a set of <AssertionConsumerService> elements in the request issuer's  
 1922 metadata as the one to which the <Response> should be returned. It applies only to profiles that  
 1923 specify use of this metadata element, in which the request issuer is different than the presenter. If  
 1924 omitted, the metadata element labeled with the isDefault attribute MUST be used with such  
 1925 profiles.

1926 AssertionConsumerServiceURL [Optional]

1927 If the would-be presenter of an <AuthnRequest> recognizes that the issuer's request cannot be  
 1928 satisfied for some reason, this attribute specifies where a <Response> message generated by that  
 1929 would-be presenter MUST be returned. This attribute can be required by certain profiles.

1930 ProviderName [Optional]

1931 Specifies the human-readable name of the request issuer for use by the presenter's user agent or  
 1932 the identity provider.

1933 See Section 3.4.1.8 for general processing rules regarding this message.

1934 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**  
 1935 complex type:

```

1936 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
1937 <complexType name="AuthnRequestType">
1938   <complexContent>
1939     <extension base="samlp:RequestAbstractType">
1940       <sequence>
1941         <element ref="saml:Subject" minOccurs="0"/>
1942         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
1943         <element
1944         ref="saml:Conditions" minOccurs="0"/>
1945         <element ref="samlp:RequestAuthnContext"
1946         minOccurs="0"/>

```

```

1947         <element ref="samlp:Scoping" minOccurs="0"/>
1948     </sequence>
1949     <attribute name="IsPassive" type="boolean"
1950 use="optional"/>
1951     <attribute name="ForceAuthn" type="boolean"
1952 use="optional"/>
1953     <attribute name="ProtocolBinding" type="anyURI"
1954 use="optional"/>
1955     <attribute name="AssertionConsumerServiceID" type="string"
1956 use="optional"/>
1957     <attribute name="AssertionConsumerServiceURL"
1958 type="anyURI" use="optional"/>
1959     <attribute name="ProviderName" type="string"
1960 use="optional"/>
1961 </extension>
1962 </complexContent>
1963 </complexType>

```

### 1964 **3.5.1.1 Element <NameIDPolicy>**

1965 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
1966 <AuthnRequest>. Its NameIDPolicyType complex type defines the following attributes:

#### 1967 Format [Required]

1968 Specifies the URI of a name identifier format defined in this or another specification (see Section 7.3  
1969 for examples).

#### 1970 SPNameQualifier [Optional]

1971 Used with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated or  
1972 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, it optionally specifies that a  
1973 federated identifier be returned (or created) in the namespace of a service provider other than the  
1974 issuing service provider, or an affiliation group.

1975 When this element is used, if the content is not understood by or acceptable to the identity provider,  
1976 then a <Response> MUST be returned with a <Status> containing a second-level <StatusCode> of  
1977 samlp:InvalidNameIDPolicy.

1978 A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated expresses the  
1979 request issuer's willingness, at the discretion of the requested subject, to establish an identity federation  
1980 for the subject with the identity provider, if one does not already exist. But note that when  
1981 <NameIDPolicy> is omitted, the identity provider MAY, at its (and the subject's) discretion, also  
1982 establish such an identity federation with the understanding that the issuing service provider might  
1983 ignore the federated and persistent aspect of the identifier.

1984 A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates that the  
1985 resulting assertion(s) MUST contain <EncryptedIdentifier> elements instead of plaintext. The  
1986 underlying name identifier's unencrypted form can be of any type supported by the identity provider for  
1987 the requested subject.

1988 Any Format value (or the omission of this element) MAY result in an <EncryptedIdentifier> in the  
1989 resulting assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1990 The following schema fragment defines the <NameIDPolicy> element and its NameIDPolicyType  
1991 complex type:

```

1992     <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
1993     <complexType name="NameIDPolicyType">
1994         <sequence/>
1995         <attribute name="Format" type="anyURI" use="required"/>
1996         <attribute name="SPNameQualifier" type="string" use="optional"/>

```



1997 | `</complexType>`

### 1998 | **3.5.1.2 Element <RequestAuthnContext>**

1999 | The <RequestAuthnContext> element specifies the authentication context requirements of the  
2000 | request issuer with respect to the authentication of the presenter. Its RequestAuthnContextType  
2001 | complex type defines the following elements and attributes:

2002 | <AuthnContextClassRef> or <AuthnContextStatementRef> [One or More]

2003 | Specifies one or more URIs identifying authentication context classes or statements.

2004 | Comparison [Optional]

2005 | Specifies the comparison method used to evaluate the requested context classes or statements, one  
2006 | of "exact", "minimum", "maximum", or "better". The default is "exact".

2007 | If <RequestAuthnContext> is specified in an <AuthnRequest> message, the authentication  
2008 | statement in the resulting assertion MUST contain an authentication context that conforms to the  
2009 | requested context as described below.

2010 | Either a set of class references or statement references can be used. Additionally, the set of supplied  
2011 | references MUST be evaluated as an ordered set, where the first element is the most preferred  
2012 | authentication context class or statement. If none of the specified classes or statements can be satisfied  
2013 | in accordance with the rules below, then the identity provider MUST return a <Response> message with  
2014 | a second-level <StatusCode> of saml:NoAuthnContext.

2015 | If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
2016 | statement MUST be the exact match of at least one of the authentication contexts specified.

2017 | If Comparison is set to "minimum", then the resulting authentication context in the authentication  
2018 | statement MUST be at least as strong (as deemed by the identity provider) as one of the authentication  
2019 | contexts specified.

2020 | If Comparison is set to "better", then the resulting authentication context in the authentication statement  
2021 | MUST be stronger (as deemed by the identity provider) than any one of the authentication contexts  
2022 | specified.

2023 | If Comparison is set to "maximum", then the resulting authentication context in the authentication  
2024 | statement MUST be as strong as possible (as deemed by the identity provider) without exceeding the  
2025 | strength of at least one of the authentication contexts specified.

2026 | The following schema fragment defines the <RequestAuthnContext> element and its  
2027 | RequestAuthnContextType complex type:

```
2028 | <element name="RequestAuthnContext" type="saml:RequestAuthnContextType"/>  
2029 | <complexType name="RequestAuthnContextType">  
2030 |   <choice>  
2031 |     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>  
2032 |     <element ref="saml:AuthnContextStatementRef"  
2033 | maxOccurs="unbounded"/>  
2034 |   </choice>  
2035 |   <attribute name="Comparison" type="saml:AuthnContextComparisonType"  
2036 | use="optional"/>  
2037 | </complexType>  
2038 | <simpleType name="AuthnContextComparisonType">  
2039 |   <restriction base="string">  
2040 |     <enumeration value="exact"/>  
2041 |     <enumeration value="minimum"/>  
2042 |     <enumeration value="maximum"/>  
2043 |     <enumeration value="better"/>  
2044 | </restriction>
```

2045 | `</restriction>`  
2046 | `</simpleType>`

### 2047 | **3.5.1.3 Element <Scoping>**

2048 | The <Scoping> element specifies the identity providers trusted by the request issuer to authenticate the  
2049 | presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to  
2050 | subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
2051 | elements and attribute:

2052 | <IDPList> [Optional]

2053 | An advisory list of identity providers and associated information that the request issuer deems  
2054 | acceptable to respond to the request.

2055 | <RequesterID> [Zero or More]

2056 | Identifies the set requesting entities on whose behalf the request issuer is acting. Used to  
2057 | communicate the chain of request issuers when proxying occurs, as described in section 3.4.1.9.

2058 | ProxyCount [Optional]

2059 | Specifies the number of proxying indirections permissible between the identity provider that receives  
2060 | this <AuthnRequest> and the identity provider who ultimately authenticates the principal. A count  
2061 | of zero permits no proxying, while omitting this attribute expresses no such restriction.

2062 | In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
2063 | <Response> message with a second-level <StatusCode> of samlp:NoAvailableIDP or  
2064 | samlp:NoSupportedIDP if it cannot contact or does not support any of the specified identity providers.

2065 | The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

2066 | `<element name="Scoping" type="samlp:ScopingType"/>`  
2067 | `<complexType name="ScopingType">`

### 2068 | **3.5.2 Element <Status>**

2069 | The <Status> element contains the following elements:

2070 | <StatusCode> [Required]

2071 | A code representing the status of the corresponding request.

2072 | <StatusMessage> [Optional]

2073 | A message which MAY be returned to an operator.

2074 | <StatusDetail> [Optional]

2075 | Additional information concerning an error condition.

2076 | The following schema fragment defines the <Status> element and its **StatusType** complex type:

2077 | `<element name="Status" type="samlp:StatusType"/>`  
2078 | `<complexType name="StatusType">`

2079 | `<sequence>`

2080 | `<element ref="samlp:IDPList" minOccurs="0" status="code" />`

2081 | `<element ref="samlp:RequesterID" minOccurs="0"`

2082 | `maxOccurs="unbounded" status="message" minOccurs="0" />`

2083 | `<element ref="samlp:StatusDetail" minOccurs="0" />`

2084 | `</sequence>`

2085 | `<attribute name="ProxyCount" type="nonNegativeInteger" use="optional" />`

```
2086 | </complexType>  
2087 | <element name="RequesterID" type="anyURI"/><StatusCode>
```

### 2088 | **3.5.2.1 Element <IDPList>**

2089 | The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the  
2090 | presenter. Its IDPListType complex type defines the following elements:

2091 | <IDPEntry> [One or More]

2092 | Information about a single identity provider

2093 | <GetComplete> [Optional]

2094 | If the <IDPList> is not complete, this element may specify a URI that resolves to the complete list.

2095 | The following schema fragment defines the <IDPList> element and its IDPListType complex type:

```
2096 | <element name="IDPList" type="samlp:IDPListType"/>  
2097 | <complexType name="IDPListType">
```

2098 | The <StatusCode> element specifies one or more possibly-nested, codes representing the status of the  
2099 | corresponding request. The <StatusCode> element has the following element and attribute:

2100 | Value [Required]

2101 | The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST  
2102 | be provided. The value of the topmost <StatusCode> element MUST be from the top-level list  
2103 | provided in this section.

2104 | <StatusCode> [Optional]

2105 | A subordinate status code that provides more specific information on an error condition.

2106 | The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
2107 | local parts of these QNames are as follows:

2108 | Success

2109 | The request succeeded.

2110 | VersionMismatch

2111 | The SAML responder could not process the request because the version of the request message  
2112 | was incorrect.

2113 | Requester

2114 | The request could not be performed due to an error on the part of the requester.

2115 | Responder

2116 | The request could not be performed due to an error on the part of the SAML responder or SAML  
2117 | authority.

2118 | The following second-level status codes are referenced at various places in the specification. Additional  
2119 | second-level status codes MAY be defined in future versions of the SAML specification.

2120 | RequestVersionTooHigh

2121 | The SAML responder cannot process the request because the protocol version specified in the  
2122 | request message is a major upgrade from the highest protocol version supported by the responder.

2123 | RequestVersionTooLow

2124 | The SAML responder cannot process the request because the protocol version specified in the

2125 | ~~request message is too low.~~

2126 | ~~RequestVersionDeprecated~~

2127 | ~~The SAML responder can not process any requests with the protocol version specified in the~~  
2128 | ~~request.~~

2129 | ~~TooManyResponses~~

2130 | ~~The response message would contain more elements than the SAML responder will return.~~

2131 | ~~RequestDenied~~

2132 | ~~The SAML responder or SAML authority is able to process the request but has chosen not to~~  
2133 | ~~respond. This status code MAY be used when there is concern about the security context of the~~  
2134 | ~~request message or the sequence of request messages received from a particular requester.~~

2135 | ~~ResourceNotRecognized~~

2136 | ~~The SAML authority does not wish to support resource specific attribute queries, or the resource~~  
2137 | ~~value provided in the request message is invalid or unrecognized.~~

2138 | ~~SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT~~  
2139 | ~~define additional codes in the SAML assertion or protocol namespace.~~

2140 | ~~The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value~~  
2141 | ~~attribute and have the above semantics only in that context.~~

2142 | ~~The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex~~  
2143 | ~~type:~~

```
2144 | <element name="StatusCode" type="samlp:StatusCodeType"/>  
2145 | <complexType name="StatusCodeType">  
2146 |   <sequence>  
2147 |     <element ref="samlp:IDPEntry" maxOccurs="unbounded" status="deprecated" minOccurs="0"/>  
2148 |   </sequence>  
2149 | </complexType>  
2150 | <attribute name="Value" type="QName" use="required"/>  
2151 | <element name="GetComplete" type="anyURI"/>  
2152 |
```

### 2153 | **3.5.2.2 Element <IDPEntry>**

2154 | ~~The <IDPEntry> element specifies a single identity provider trusted by the request issuer to~~  
2155 | ~~authenticate the presenter. Its **IDPEntryType** complex type defines the following elements:~~

2156 | ~~<ID> [Required]~~

2157 | ~~The unique identifier of the identity provider~~

2158 | ~~<Name> [Optional]~~

2159 | ~~A human readable name for the identity provider~~

2160 | ~~<Loc> [Optional]~~

2161 | ~~The location of a profile-specific endpoint supporting the authentication request protocol. The~~  
2162 | ~~binding to be used must be understood from the profile of use.~~

2163 | ~~The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:~~

```
2164 | <element name="IDPEntry" type="samlp:IDPEntryType"/>  
2165 | <complexType name="IDPEntryType">  
2166 |   <sequence/>  
2167 |   <attribute name="ID" type="anyURI" use="required"/>  
2168 |   <attribute name="Name" type="string" use="optional"/>
```

2169 | `<attribute name="Loc" type="anyURI" use="optional"/>`  
2170 | `</complexType>`

### 2171 | **3.5.2.3 Processing Rules**

2172 | The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
2173 | therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
2174 | kinds of input and output are required or prohibited. The following processing rules apply as invariant  
2175 | behavior across any profile of this protocol exchange.

2176 | The recipient MUST validate any signature present on the request or response message. To be  
2177 | considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2178 | message.

2179 | The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing  
2180 | one or more assertions that meet the specifications defined by the request, or a <Status> describing  
2181 | the error that occurred. The responder MAY conduct additional message exchanges with the request  
2182 | sender as needed to initiate or complete the authentication process, subject to the nature of the protocol  
2183 | binding and the authentication mechanism. As described in the next section, this includes proxying the  
2184 | request by directing the presenter to another identity provider by issuing its own <AuthnRequest>  
2185 | message, so that the resulting assertion can be used to authenticate the presenter to the original  
2186 | responder.

2187 | If the responder is unable to authenticate the presenter or does not recognize the requested subject, it  
2188 | MUST return a <Response> with a <Status> containing a second-level <StatusCode> of  
2189 | samlp:UnknownPrincipal.

2190 | If the <Subject> element in the request is present, then the resulting assertions' <Subject> MUST  
2191 | strongly match the request <Subject>, as described in section 3.3.4.1, except that the identifier MAY  
2192 | be in a different form if specified by <NameIDPolicy>.

2193 | All of the content defined specifically within <AuthnRequest> is optional, although some may be  
2194 | required by certain profiles. In the absence of any specific content at all, the following behavior is  
2195 | assumed:

- 2196 | • The assertion(s) returned MUST contain a <Subject> element that represents the presenter.  
2197 | The identifier type and format are determined by the identity provider. At least one statement  
2198 | MUST be an <AuthenticationStatement> that describes the authentication performed by the  
2199 | responder or authentication service associated with it.
- 2200 | • The request presenter should, to the extent possible, be the only entity able to satisfy the  
2201 | <SubjectConfirmation> of the assertion(s). In the case of weaker confirmation methods,  
2202 | binding-specific or other mechanisms will be used to help satisfy this requirement.
- 2203 | • The resulting assertion(s) MUST contain an <AudienceRestrictionCondition> element  
2204 | referencing the request issuer as an acceptable relying party. Other audiences MAY be included  
2205 | as deemed appropriate by the identity provider.

2206 |

### 2207 | **3.5.2.4 Proxying**

2208 | If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or  
2209 | cannot directly authenticate him/her, but believes that the presenter has already authenticated to another  
2210 | identity provider, it may respond to the request by issuing a new <AuthnRequest> on its own behalf to

2211 be presented to the other identity provider. The original identity provider is termed the proxying identity  
2212 provider.

2213 Upon the successful return of a <Response> to the proxying provider, the enclosed assertion MAY be  
2214 used to authenticate the presenter so that the proxying provider can issue an assertion of its own in  
2215 response to the original <AuthnRequest>, completing the overall message exchange. Both the  
2216 proxying and authenticating identity providers MAY include constraints on proxying activity in the  
2217 messages and assertions they issue, as described in previous sections, and below.

2218

2219 The request issuer can influence proxy behavior by including a <Scoping> element where the provider  
2220 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be  
2221 proxied by including an ordered <IDPList> of preferred providers.

2222 An identity provider can control secondary use of its assertions by proxying identity providers using a  
2223 <ProxyRestrictionCondition> element in the assertions it issues.

#### 2224 **3.5.2.4.1 Processing Rules**

2225 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is  
2226 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY  
2227 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

2228 An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. The identity  
2229 provider MUST return an error containing a second-level <samlp:StatusCode> value of  
2230 samlp:ProxyCountExceeded, unless it can directly authenticate the presenter.

2231 If it chooses to proxy, when creating the new <AuthnRequest>, an identity provider MUST include  
2232 equivalent or stricter forms of all the information included in the original request (such as authentication  
2233 context policy). Note however that the proxying provider is free to specify whatever <NameIDPolicy> it  
2234 wishes to maximize the chances of a successful response.

2235 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST  
2236 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for  
2237 example) will be honored by the authenticating provider.

2238 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less  
2239 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new  
2240 request SHOULD contain a <ProxyCount> attribute.

2241 If an <IDPList> was specified in the original request, the new request MUST also contain an  
2242 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the  
2243 <IDPList>, but MUST NOT remove any from the list.

2244 The authentication request and response are processed in normal fashion, in accordance with the rules  
2245 given in Section 3.4.1.8 and the profile of use. Once the presenter has authenticated to the proxying  
2246 identity provider (by delivering a <Response>), the following steps are followed:

- 2247 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
2248 relevant information from the original assertion. The original assertion will be restricted by  
2249 <AudienceRestrictionCondition> to (at least) the proxying identity provider, while the new  
2250 assertion's condition will reference (at least) the original request issuer.
- 2251 • The new assertion's <Subject> should contain an identifier that satisfies the original request  
2252 issuer's preferences, as defined by its <NameIDPolicy> element.

- 2253 | • The <AuthenticationStatement> in the new assertion MUST include an <AuthnContext>  
2254 | element containing an <ac:AuthenticatingAuthority> element referencing the identity  
2255 | provider to which the proxying identity provider referred the presenter. If the original assertion  
2256 | contains <AuthnContext> information that includes one or more  
2257 | <ac:AuthenticatingAuthority> elements, those elements SHOULD be included in the new  
2258 | assertion, with the new element placed after them.
- 2259 | • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
2260 | MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
2261 | consistent over time across different requests. The value MUST not conflict with values used or  
2262 | generated by other SAML providers.
- 2263 | • Any other <AuthnContext> information MAY be copied, translated, or omitted in accordance  
2264 | with the policies of the proxying identity provider, provided that the original requirements dictated  
2265 | by the request issuer are met.
- 2266 | If, in the future, the identity provider is asked to authenticate the same presenter for a second request  
2267 | issuer, and this request is equally or less strict than the original request, the identity provider MAY skip  
2268 | the creation of a new <AuthnRequest> to the authenticating identity provider and immediately issue  
2269 | another assertion (assuming the original assertion it received is still valid). The concrete definition of  
2270 | "equally or less strict" is up to the proxying identity provider.

## 2271 | **3.6 Artifact Protocol**

2272 | The artifact protocol provides a mechanism by which SAML protocol messages can be transported in a  
2273 | SAML binding by reference instead of by value. Both requests and responses can be obtained by  
2274 | reference using this specialized protocol. A message sender, instead of binding a message to a transport  
2275 | protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of  
2276 | forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes,  
2277 | it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
2278 | protocol to dereference the artifact into the original protocol message. The most common use for this  
2279 | mechanism is with bindings that cannot easily carry a message because of size constraints.

2280 | Depending on the characteristics of the underlying message being passed by reference, the artifact  
2281 | protocol MAY require protections such as mutual authentication, integrity protection, confidentiality, etc.  
2282 | from the protocol binding used to dereference the artifact. In all cases, the artifact MUST exhibit a single-  
2283 | use semantic such that once it has been successfully dereferenced, it can no longer be used by any  
2284 | party.

2285 | Regardless of the protocol message obtained, the result of dereferencing an artifact MUST be treated  
2286 | exactly as if the message so obtained had been sent originally in place of the artifact.

### 2287 | **3.6.1 Element <ArtifactRequest>**

2288 | The <ArtifactRequest> message is used to request that a protocol message be returned in an  
2289 | <ArtifactResponse> message by specifying an artifact that represents the protocol message. The  
2290 | original transmission of the artifact is governed by the specific binding or profile of SAML that is being  
2291 | used; see the SAML specifications for bindings [SAMLBind] and profiles [SAMLProf] for more information  
2292 | on the use of artifacts in bindings and profiles.

2293 | The <ArtifactRequest> message SHOULD be signed or otherwise authenticated and integrity  
2294 | protected by the protocol binding used to deliver the message.

2295 | The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
2296 | Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2297 This message has the complex type **ArtifactRequestType**, which extends **RequestAbstractType** and  
2298 adds the following element:

2299 <Artifact> [Required]

2300 The artifact value that the requester received and now wishes to translate into the protocol message  
2301 it represents. See [SAMLBind] for specific artifact format information.

2302 The following schema fragment defines the <ArtifactRequest> element and its  
2303 **ArtifactRequestType** complex type:

```
2304 <element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>
2305 <complexType name="ArtifactRequestType">
2306   <complexContent>
2307     <extension base="samlp:RequestAbstractType">
2308       <sequence>
2309         <element ref="samlp:Artifact"/>
2310       </sequence>
2311     </extension>
2312   </complexContent>
2313 </complexType>
2314 <element name="Artifact" type="string"/>
```

### 2315 **3.6.2 Element <ArtifactResponse>**

2316 The recipient of an <ArtifactRequest> message MUST respond with an <ArtifactResponse>  
2317 message, which is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a  
2318 single optional wildcard element corresponding to the protocol message being returned. This wrapped  
2319 message element can be a request or a response.

2320 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity  
2321 protected by the protocol binding used to deliver the message.

2322 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2323 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2324 The following schema fragment defines the <ArtifactResponse> element and its  
2325 **ArtifactResponseType** complex type:

```
2326 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2327 <complexType name="ArtifactResponseType">
2328   <complexContent>
2329     <extension base="samlp:StatusResponseType">
2330       <sequence>
2331         <any namespace="#any" processContents="lax"
2332           minOccurs="0"/>
2333       </sequence>
2334     </extension>
2335   </complexContent>
2336 </complexType>
```

### 2337 **3.6.3 Processing Rules**

2338 The recipient MUST validate any signature present on the request or response message. To be  
2339 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2340 message.

2341 If the responder recognizes the artifact as valid, then it responds with the associated protocol message  
2342 in an <ArtifactResponse> message. Otherwise, it responds with an <ArtifactResponse>  
2343 message with no embedded message. In both cases, the <Status> element MUST include a



2344 <StatusCode> element with the code value Success. A response message with no embedded  
2345 message inside it is termed an empty response in the remainder of this section.

2346 The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent  
2347 request with the same artifact by any requester results in an empty response as described above.

2348 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles,  
2349 MAY be intended for consumption by any party that receives it and can respond appropriately. In most  
2350 other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued  
2351 MUST be associated with the intended recipient of the message that the artifact represents. If the artifact  
2352 issuer receives an <ArtifactRequest> from a requester that cannot authenticate itself as the original  
2353 intended recipient, then the artifact issuer MUST return an empty response.

2354 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such  
2355 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and  
2356 return it in an <ArtifactRequest> to the issuer.

2357 Note that the <ArtifactResponse>'s InResponseTo attribute MUST contain the value of the  
2358 corresponding <AssertionRequest>'s RequestID attribute, but the embedded protocol message will  
2359 contain its own message identifier, and in the case of an embedded response, may contain a different  
2360 InResponseTo value that corresponds to the original request message to which the embedded  
2361 message is responding.

## 2362 **3.7 Federated Name Registration Protocol**

2363 When an identity provider and service provider first federate a principal's identity using a  
2364 <NameIdentifier> element with a Format of urn:oasis:names:tc:SAML:2.0:nameid-  
2365 format:federated, the identity provider generates an opaque value that serves as the initial name  
2366 identifier that both the service provider and the identity provider use in referring to the principal when  
2367 communicating with each other.

2368 Subsequent to federation, the service provider MAY register a different opaque value with the identity  
2369 provider. This opaque value is an attribute termed the SPProvidedIdentifier. Until the service provider  
2370 registers a different name, this attribute is omitted from <NameIdentifier> elements referring to the  
2371 principal.

2372 Either the service provider or the identity provider MAY register a new name identifier for a principal with  
2373 each other at any time following federation. The name identifiers specified by providers SHOULD be  
2374 unique across the identity providers with which the principal's identity is federated and SHOULD be  
2375 unique within the group of name identifiers that have been registered with the identity provider by this  
2376 service provider.

2377 Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-  
2378 format:federated) can be replaced and set with this protocol: non-federated, encrypted, or transient  
2379 identifiers MUST NOT be used.

### 2380 **3.7.1 Element <RegisterNameIdentifierRequest>**

2381 To register an SPProvidedIdentifier attribute with an identity provider, the service provider sends a  
2382 <RegisterNameIdentifierRequest> message. The same message may be sent by an identity  
2383 provider, seeking to change the <NameIdentifier> value stored by the service provider.

2384 The <RegisterNameIdentifierRequest> message SHOULD be signed or otherwise authenticated  
2385 and integrity protected by the protocol binding used to deliver the message

2386 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
2387 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2388 This message has the complex type RegisterNameIdentifierRequestType, which extends  
2389 RequestAbstractType and adds the following elements:

2390 <NameIdentifier> [Required]

2391 The federated name identifier and associated attributes that specify the principal as currently  
2392 recognized by the identity and service providers prior to this request.

2393 <NewIdentifier> [Required]

2394 The new federated identifier value to be used when communicating with the requesting provider  
2395 concerning this principal. If the requester is the service provider, the new identifier will appear in  
2396 subsequent <NameIdentifier> elements in the SPProvidedIdentifier attribute. If the  
2397 requester is the identity provider, the new value will appear in subsequent <NameIdentifier>  
2398 elements as the element's value.

2399 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
2400 RegisterNameIdentifierRequestType complex type:

```
2401 <element name="NewIdentifier" type="string">  
2402 <element name="RegisterNameIdentifierRequest"  
2403 type="samlp:RegisterNameIdentifierRequestType"/>  
2404 <complexType name="RegisterNameIdentifierRequestType">  
2405 <complexContent>  
2406 <extension base="samlp:RequestAbstractType">  
2407 <sequence>  
2408 <element ref="saml:NameIdentifier"/>  
2409 <element ref="samlp:NewIdentifier"/>  
2410 </sequence>  
2411 </extension>  
2412 </complexContent>  
2413 </complexType>
```

### 2414 **3.7.2 Element <RegisterNameIdentifierResponse>**

2415 The recipient of a <RegisterNameIdentifierRequest> message MUST respond with a  
2416 <RegisterNameIdentifierResponse> message, which is of type StatusResponseType with no  
2417 additional content.

2418 The <RegisterNameIdentifierResponse> message SHOULD be signed or otherwise authenticated  
2419 and integrity protected by the protocol binding used to deliver the message.

2420 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2421 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2422 The following schema fragment defines the <RegisterNameIdentifierResponse> element:

```
2423 <element name="RegisterNameIdentifierResponse"  
2424 type="samlp:StatusResponseType"/>
```

### 2425 **3.7.3 Processing Rules**

2426 The recipient MUST validate any signature present on the request or response message. To be  
2427 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2428 message.

2429 If the request includes a <NameIdentifier> for which no federation exists between the service  
2430 provider and the identity provider, the responding provider MUST respond with a <Status> containing a  
2431 second-level <StatusCode> of samlp:FederationDoesNotExist.

2432 If the service provider requests that its identifier be changed, the identity provider MUST include the  
2433 <NewIdentifier> element's value as the SPProvidedIdentifier when subsequently  
2434 communicating to the service provider regarding this principal.

2435 If the identity provider requests that its identifier be changed, the service provider MUST use the  
2436 <NewIdentifier> element's value as the <NameIdentifier> element value when subsequently  
2437 communicating with the identity provider regarding this principal.

2438 In either case, the <NameIdentifier> value in the request and its associated  
2439 SPProvidedIdentifier attribute MUST contain the most recent name identifier information  
2440 established between the providers for the principal. The NameQualifier attribute MUST contain the  
2441 unique identifier of the identity provider. If the principal's identity federation is between the identity  
2442 provider and an affiliation group of which the service provider is a member, then the SPNameQualifier  
2443 attribute MUST contain the unique identifier of the affiliation group. Otherwise, it MUST contain the  
2444 unique identifier of the service provider.

2445 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
2446 systems at both the requester and the responder. Implementations might wish to allow each party to  
2447 accept either identifier for some period of time following the successful completion of a name identifier  
2448 change. Not doing so could result in the inability of the principal to access resources.

2449 All other processing rules associated with the underlying request and response messages MUST be  
2450 observed.

## 2451 **3.8 Federation Termination Protocol**

2452 When a principal (or an appropriate agent acting on his or her behalf) terminates an identity federation  
2453 between a service provider and an identity provider through an interaction with the service provider, the  
2454 service provider MUST send a <FederationTerminationNotification> message to the identity  
2455 provider. The service provider is stating that it will no longer accept authentication assertions from the  
2456 identity provider for the specified principal.

2457 Likewise, when a principal terminates an identity federation through an interaction with the identity  
2458 provider, the identity provider MUST send a <FederationTerminationNotification> message to  
2459 the service provider. In this case, the identity provider is stating that it will no longer provide  
2460 authentication assertions to the service provider for the specified principal.

2461 Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-  
2462 format:federated) can be replaced and set with this protocol; non-federated, encrypted, or transient  
2463 identifiers MUST NOT be used.

### 2464 **3.8.1 Element <FederationTerminationNotification>**

2465 A provider sends a <FederationTerminationNotification> to the provider with which it is  
2466 terminating a federation. The <FederationTerminationNotification> message SHOULD be  
2467 signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the  
2468 message.

2469 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
2470 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2471 This message has the complex type **FederationTerminationNotificationType**, which extends  
2472 **RequestAbstractType** and adds the following elements:

2473 <NameIdentifier> [Required]

2474 The federated name identifier and associated attributes that specify the principal as currently  
2475 recognized by the identity and service providers prior to this request. Format MUST be  
2476 urn:oasis:names:tc:SAML:2.0:nameid-format:federated.

2477 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
2478 **RegisterNameIdentifierRequestType** complex type:

```
2479 <element name="FederationTerminationNotification"  
2480 type="saml:FederationTerminationNotificationType"/>  
2481 <complexType name="FederationTerminationNotificationType">  
2482 <complexContent>  
2483 <extension base="saml:RequestAbstractType">  
2484 <sequence>  
2485 <element ref="saml:NameIdentifier"/>  
2486 </sequence>  
2487 </extension>  
2488 </complexContent>  
2489 </complexType>
```

### 2490 **3.8.2 Element <FederationTerminationResponse>**

2491 The recipient of a <FederationTerminationNotification> message MUST respond with a  
2492 <FederationTerminationResponse> message, which is of type **StatusResponseType** with no  
2493 additional content.

2494 The <FederationTerminationResponse> message SHOULD be signed or otherwise authenticated  
2495 and integrity protected by the protocol binding used to deliver the message.

2496 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2497 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2498 The following schema fragment defines the <FederationTerminationResponse> element:

```
2499 <element name="FederationTerminationResponse"  
2500 type="saml:StatusResponseType"/>
```

### 2501 **3.8.3 Processing Rules**

2502 The recipient MUST validate any signature present on the request or response message. To be  
2503 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2504 message.

2505 If the request includes a <NameIdentifier> for which no federation exists between the service  
2506 provider and the identity provider, the responding provider MUST respond with a <saml:Status>  
2507 containing a second-level <saml:StatusCode> of saml:FederationDoesNotExist.

2508 Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been  
2509 terminated. A provider MAY choose to invalidate the session of a user for whom federation has been  
2510 terminated.

2511 All other processing rules associated with the underlying request and response messages MUST be  
2512 observed.

## 2513 | **3.9 Single Logout Protocol**

2514 | The single logout protocol provides a message exchange protocol by which all sessions provided by a  
2515 | particular session authority are near-simultaneously terminated. The single logout protocol is used either  
2516 | when a principal logs out at a session participant or when the principal logs out directly at the  
2517 | session authority. This protocol may also be used to logout a principal due to a timeout. The reason for  
2518 | the logout event may be indicated through the `reason` attribute.

2519 | The principal may have established authenticated sessions both with the session authority, and  
2520 | individual session participants, based on authentication assertions supplied by the session authority.

2521 | When the principal invokes the single logout process at a session participant, the session participant  
2522 | MUST send a `<LogoutRequest>` message to the session authority that provided the authentication  
2523 | service related to that session at the session participant.

2524 | When either the principal invokes a logout at the session authority, or a session participant sends a  
2525 | logout request to the session authority specifying that principal, the session authority MUST send a  
2526 | `<LogoutRequest>` message to each session participant to which it provided authentication assertions  
2527 | under its current session with the principal, with the exception of the session participant that sent the  
2528 | `<LogoutRequest>` message to the session authority.

### 2533 | **3.9.1 Element `<LogoutRequest>`**

2534 | A session participant or session authority sends a `<LogoutRequest>` message to indicate that a  
2535 | session has been terminated.

2536 | The `<LogoutRequest>` message SHOULD be signed or otherwise authenticated and integrity  
2537 | protected by the protocol binding used to deliver the message.

2538 | This message has the complex type `LogoutRequestType`, which extends `RequestAbstractType`, and  
2539 | adds the following elements and attributes:

2540 | `<NameIdentifier>` [Required]

2541 | The name identifier and associated attributes that specify the principal as currently recognized by  
2542 | the identity and service providers prior to this request.

2543 | `<SessionIndex>` [Optional]

2544 | The identifier that indexes this session at the message recipient.

2545 | `NotOnOrAfter` [Optional]

2546 | The time at which the request expires.

2547 | `Reason` [Optional]

2548 | An indication of the reason for the logout.

2549 | The following schema fragment defines the `<LogoutRequest>` element and associated  
2550 | `LogoutRequestType` complex type:

```

2551 | <element name="LogoutRequest" type="samlp:LogoutRequestType"/>
2552 | <complexType name="LogoutRequestType">
2553 | <complexContent>
2554 | <extension base="samlp:RequestAbstractType">
2555 | <sequence>
2556 | <element ref="saml:NameIdentifier"/>
2557 | <element name="SessionIndex" type="string" minOccurs="0"
2558 | maxOccurs="unbounded"/>
2559 | </sequence>
2560 | <attribute name="Reason" type="string" minOccurs="0"/>
2561 | <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
2562 | </extension>
2563 | </complexContent>
2564 | </complexType>

```

## 2565 | **3.9.2 Element <LogoutResponse>**

2566 | The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message,  
2567 | of type **StatusResponseType**, with no additional content specified.

2568 | The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity  
2569 | protected by the protocol binding used to deliver the message.

2570 | The following schema fragment defines the <LogoutResponse> element:

```

2571 | <element name="LogoutResponse" type="samlp:StatusResponseType"/>

```

## 2572 | **3.9.3 Processing Rules**

2573 | The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting or  
2574 | responding provider, with a Format value of urn:oasis:names:tc:SAML:2.0:nameid-  
2575 | format:provider.

2576 | Message recipients MUST validate any signature present on the messages specified in this protocol. To  
2577 | be considered valid, the signature provided must be the signature of the <Issuer> contained in the  
2578 | message.

2579 | The message sender MAY use the Reason attribute to indicate the reason for sending the  
2580 | <LogoutRequest>. Other values MAY be agreed upon between participants, but the following values  
2581 | are defined directly by this specification for use by all message senders:

2582 | urn:oasis:names:tc:SAML:2.0:logout:user

2583 | Specifies that the message is being sent because the principal wishes to terminate the indicated  
2584 | session.

2585 | urn:oasis:names:tc:SAML:2.0:logout:admin

2586 | Specifies that the message is being sent because an administrator wishes to terminate the indicated  
2587 | session for that principal.

2588 | All other processing rules associated with the underlying request and response messages MUST be  
2589 | observed.

### 2590 | **3.9.3.1 Session Participant Rules**

2591 | When a session participant receives a <LogoutRequest>, the session participant MUST authenticate  
2592 | the message.. If the sender is the authority that provided an assertion linked to the principal's current  
2593 | session, the session participant MUST invalidate the principal's session(s) referred to by the  
2594 | <NameIdentifier> element, and any <SessionIndex> elements supplied in the message.

2595 |  
2596 | The session participant MUST apply the logout request message to any assertion that meets the  
2597 | following conditions, even if the assertion arrives after the logout request:  
2598 | • The <SessionIndex> of the assertion's statements matches one specified in the logout request.  
2599 | • The assertion would otherwise be valid  
2600 | • The logout request has not yet expired (determined by examining the NotOnOrAfter attribute on  
2601 | the message).

### 2602 | **3.9.3.2 Session Authority Rules**

2603 | When a session authority receives a <LogoutRequest>, the session authority MUST authenticate the  
2604 | sender. If the sender is a session participant to which the session authority provided an assertion for the  
2605 | current session, then the session authority SHOULD do the following:

- 2606 | • Send a <LogoutRequest> message to each session participant for which the session authority  
2607 | provided assertions in the current session, other than the originator of a current  
2608 | <LogoutRequest>.
- 2609 | • Send a <LogoutRequest> message to any session authority on behalf of whom the session  
2610 | authority proxied the user's authentication, unless the second authority is the originator of the  
2611 | <LogoutRequest>.
- 2612 | • Terminate the principal's current session as specified by the <NameIdentifier> element, and  
2613 | any <SessionIndex> elements present in the logout request message.

2614 | It should be noted that a session authority MAY initiate a logout for reasons other than having received a  
2615 | <LogoutRequest> from a session participant – these include, but are not limited to:

- 2616 | • If some timeout period was agreed out-of-band with an individual session participant, the session  
2617 | authority MAY send a <LogoutRequest> to that individual participant alone.
- 2618 | • An agreed global timeout period has been exceeded.
- 2619 | • The principal, or some other trusted entity has requested logout of the principal, directly at the  
2620 | session authority.
- 2621 | • The session authority has determined that the principal's credentials may have been compromised.

2622 | When constructing a logout request message, the session authority MUST set the value of the  
2623 | NotOnOrAfter attribute of the message to a time value, indicating an expiration time for the message.

2624 | In addition to the values specified in section 3.6.3 for the Reason attribute, the following values are also  
2625 | available for use by the session authority only:

2626 | urn:oasis:names:tc:SAML:2.0:logout:global-timeout

2627 | Specifies that the message is being sent because of the global session timeout interval period  
2628 | being exceeded.

2629 | urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2630 | Specifies that the message is being sent because a timeout interval period agreed between a  
2631 | participant and the authority has been exceeded.

2632 | If an error occurs during this further processing of the logout (for example, relying session participants  
2633 | may not all implement the particular single logout protocol binding used by the requesting session  
2634 | participant), then the session authority MUST respond to the original requester with a  
2635 | <LogoutResponse> message, indicating the status of the logout request. The value

2636 samlp:UnsupportedBinding is provided for a second-level <samlp:StatusCode>, indicating that a  
2637 session participant should retry the <LogoutRequest> using a different protocol binding.

## 2638 **3.10 Name Identifier Mapping Protocol**

2639 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
2640 identifier for the same principal in a particular format or federation namespace, it can send a request to  
2641 the identity provider using this protocol.

2642 For example, a service provider that wishes to communicate with another service provider with whom it  
2643 does not share an identity federation for the principal can use an identity provider that shares an identity  
2644 federation for the principal with both service providers to map from its own federated identifier to a new  
2645 identifier, generally encrypted, with which it can communicate with the second service provider.

2646 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into an  
2647 <EncryptedIdentifier> element unless a specific deployment dictates such protection is  
2648 unnecessary.

### 2649 **3.10.1 Element <NameIdentifierMappingRequest>**

2650 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
2651 <NameIdentifierMappingRequest> message. This message has the complex type  
2652 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2653 element:

2654 <BaseIdentifier> or <NameIdentifier> or <EncryptedIdentifier> [Required]

2655 The identifier and associated attributes that specify the principal as currently recognized by the  
2656 requester and the responder.

2657 <NameIDPolicy>

2658 The format and optional name qualifier that describes the requirements for the identifier to be  
2659 returned.

2660 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2661 binding used to deliver the message.

2662 The following schema fragment defines the <NameIdentifierMappingRequest> element and its  
2663 **NameIdentifierMappingRequestType** complex type:

```
2664 <element name="NameIdentifierMappingRequest"  
2665 type="samlp:NameIdentifierMappingRequestType"/>  
2666 <complexType name="NameIdentifierMappingRequestType">  
2667   <complexContent>  
2668     <extension base="samlp:RequestAbstractType">  
2669       <sequence>  
2670         <choice>  
2671           <element ref="saml:BaseIdentifier"/>  
2672           <element ref="saml:NameIdentifier"/>  
2673           <element ref="saml:EncryptedIdentifier"/>  
2674         </choice>  
2675         <element ref="samlp:NameIDPolicy"/>  
2676       </sequence>  
2677     </extension>  
2678   </complexContent>  
2679 </complexType>
```



### 2680 **3.10.2 Element <NameIdentifierMappingResponseStatusMessage>**

2681 The recipient of a <NameIdentifierMappingRequest> message MUST respond with a  
2682 <NameIdentifierMappingResponse> message. This message has the complex type  
2683 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2684 element <StatusMessage> element specifies a message that MAY be returned to an operator:

2685 <NameIdentifier> or <EncryptedIdentifier> [Required]

2686 The identifier and associated attributes that specify the principal in the manner requested, usually in  
2687 encrypted form.

2688 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2689 binding used to deliver the message.

2690 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2691 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2692 The following schema fragment defines the <NameIdentifierMappingResponse> element and its  
2693 **NameIdentifierMappingResponseType** complex type:

```
2694 <element name="NameIdentifierMappingResponse"  
2695 type="samlp:NameIdentifierMappingResponseType"/>  
2696 <complexType name="NameIdentifierMappingResponseType">  
2697 <complexContent>  
2698 <extension base="samlp:StatusResponseType">  
2699 <choice>  
2700 <element ref="saml:NameIdentifier">  
2701 <element ref="saml:EncryptedIdentifier">  
2702 </choice>  
2703 </extension>  
2704 </complexContent>
```

2705 The following schema fragment defines the <StatusMessage> element and its **StatusMessageType**  
2706 complex type:

```
2707 <element name="StatusMessage" type="string"/>
```

#### 2708 **3.10.2.1 Element <StatusDetail>**

2709 The <StatusDetail> element MAY be used to specify additional information concerning an error  
2710 condition.

2711 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
2712 complex type:

```
2713 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
2714 <complexType name="StatusDetailType">  
2715 <sequence>  
2716 <any namespace="##any" processContents="lax" minOccurs="0"  
2717 maxOccurs="unbounded"/>  
2718 </sequence>  
2719 </complexType>
```

### 2720 **3.10.3 Processing RulResponses to Queries**

2721 The recipient MUST validate any signature present on the request or response message. To be  
2722 considered valid, the signature provided MUST be the signature of the <Issuer> contained in the  
2723 message.

2724 | If the responder does not recognize the principal identified in the request, it MUST respond with a  
2725 | <Status> containing a second-level <StatusCode> of samlp:UnknownPrincipal.

2726 | At the responder's discretion, the samlp:InvalidNameIDPolicy status code MAY be returned to  
2727 | indicate an inability or unwillingness to supply an identifier in the requested format. Likewise, the  
2728 | samlp:FederationDoesNotExist status code MAY be used to indicate that a requested federated  
2729 | identifier cannot be returned.

2730 | All other processing rules associated with the underlying request and response messages MUST be  
2731 | observed.

2732 | In response to a query, every assertion returned by a SAML authority MUST contain at least one  
2733 | statement whose <saml:Subject> element **strongly matches** the <saml:Subject> element found in  
2734 | the query.

2735 | A <saml:Subject> element S1 strongly matches S2 if and only if the following two conditions both  
2736 | apply:

- 2737 | • If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical  
2738 | <saml:NameIdentifier> element.
- 2739 | • If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an identical  
2740 | <saml:SubjectConfirmation> element.

2741 | If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
2742 | expressed by a query, the <Response> element MUST NOT contain an <Assertion> element and  
2743 | MUST include a <StatusCode> element with value Success. It MAY return a <StatusMessage>  
2744 | element with additional information.

2745

## 4 SAML Versioning

2746 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
2747 sections, along with processing rules for detecting and handling version differences, when applicable.  
2748 Also included are guidelines on when and why specific version information is expected to change in  
2749 future revisions of the specification.

2750 When version information is expressed as both a Major and Minor version, it may be expressed  
2751 discretely, or in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version  
2752 number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

2753  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

### 2754 4.1 SAML Specification Set Version

2755 Each release of the SAML specification set will contain a major and minor version designation describing  
2756 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
2757 content and filenames of published materials, including the specification set document(s), and XML  
2758 schema instance(s). There are no normative processing rules surrounding specification set versioning,  
2759 since it merely encompasses the collective release of normative specification documents which  
2760 themselves contain processing rules.

2761 The overall size and scope of changes to the specification set document(s) will informally dictate whether  
2762 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
2763 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics  
2764 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a  
2765 major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0,  
2766 described in Section .

#### 2767 4.1.1 Schema Version

2768 As a non-normative documentation mechanism, any XML schema instances published as part of the  
2769 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the  
2770 specification set version in which it has been published. Validating implementations MAY use the  
2771 attribute as a means of distinguishing which version of a schema is being used to validate messages, or  
2772 to support a multiplicity of versions of the same logical schema.

#### 2773 4.1.2 SAML Assertion Version

2774 The SAML <Assertion> element contains attributes for expressing the major and minor version of the  
2775 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to  
2776 document the syntax, semantics, and processing rules of the assertions of the same version. That is,  
2777 specification set version 1.0 describes assertion version 1.0, and so on.

2778 There is explicitly NO relationship between the assertion version and the SAML assertion XML  
2779 namespace that contains the schema definitions for that assertion version.

2780 The following processing rules apply:

- 2781 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported  
2782 by the authority.
- 2783 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
2784 supported by the relying party.

- 2785 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
2786 number is higher than the minor assertion version number supported by the relying party. However,  
2787 all assertions that share a major assertion version number MUST share the same general processing  
2788 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1  
2789 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0  
2790 assertion without ill effect.

### 2791 4.1.3 SAML Protocol Version

2792 The SAML protocol <Request> and <Response> elements contain attributes for expressing the major  
2793 and minor version of the request or response message using a pair of integers. Each version of the  
2794 SAML specification set will be construed so as to document the syntax, semantics, and processing rules  
2795 of the protocol messages of the same version. That is, specification set version 1.0 describes request  
2796 and response version V1.0, and so on.

2797 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace  
2798 that contains the schema definitions for protocol messages for that protocol version.

2799 The version numbers used in SAML protocol <Request> and <Response> elements will be the same  
2800 for any particular revision of the SAML specification set.

#### 2801 4.1.3.1 Request Version

2802 The following processing rules apply to requests:

- 2803 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
2804 SAML requester and the SAML responder.
- 2805 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume  
2806 that it supports requests with the highest request version supported by the requester.
- 2807 • A SAML requester MUST NOT issue a request message with a request version number matching a  
2808 response version number that the requester does not support.
- 2809 • A SAML responder MUST reject any request with a major request version number not supported by  
2810 the responder.
- 2811 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
2812 higher than the highest supported request version that it supports. However, all requests that share a  
2813 major request version number MUST share the same general processing rules and semantics, and  
2814 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax  
2815 of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

#### 2816 4.1.4 Response Version

2817 The following processing rules apply to responses:

- 2818 • A SAML responder MUST NOT issue a response message with a response version number higher  
2819 than the request version number of the corresponding request message.
- 2820 • A SAML responder MUST NOT issue a response message with a major response version number  
2821 lower than the major request version number of the corresponding request message except to report  
2822 the error `RequestVersionTooHigh`.

2823 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-  
2824 level <StatusCode> value of `VersionMismatch`, and MAY result in reporting one of the following

2825 second-level values: RequestVersionTooHigh, RequestVersionTooLow, or  
2826 RequestVersionDeprecated.

## 2827 4.1.5 Permissible Version Combinations

2828 In general, assertions of a particular major version may appear in response messages of the same major  
2829 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol  
2830 schema. Future versions of this specification are expected to explicitly describe the permitted  
2831 combinations across major versions.

2832 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to  
2833 appear in a V1.1 response message.

## 2834 4.2 SAML Namespace Version

2835 XML schema instances and "qualified names" (QNames) published as part of the specification set  
2836 contain one or more target namespaces into which the type, element, and attribute definitions are  
2837 placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and  
2838 syntactical definitions that make up that part of the specification.

2839 The namespace URIs defined by the specification set will generally contain version information of the  
2840 form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to  
2841 the major and minor version of the specification set in which the namespace is first introduced and  
2842 defined. This information is not typically consumed by an XML processor, which treats the namespace  
2843 opaquely, but is intended to communicate the relationship between the specification set and the  
2844 namespaces it defines.

2845 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
2846 defined by a major revision of the specification set to remain valid and stable across minor revisions of  
2847 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,  
2848 but this is expected to be rare. In such cases, the older namespaces and their associated definitions  
2849 should be expected to remain valid until a major specification set revision.

### 2850 4.2.1 Schema Evolution

2851 In general, maintaining namespace stability while adding or changing the content of a schema are  
2852 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
2853 older implementations will react to any given change, making forward compatibility difficult to achieve.  
2854 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of  
2855 namespace stability. Except in special circumstances (for example to correct major deficiencies or fix  
2856 errors), implementations should expect forward compatible schema changes in minor revisions, allowing  
2857 new messages to validate against older schemas.

2858 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
2859 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
2860 that leverage the extension facilities described in Section SAML Extensions. Older implementations  
2861 SHOULD reject such extensions gracefully when they are encountered in contexts that dictate mandatory  
2862 semantics. Examples include new query, statement, or condition types.

2863

## 5 SAML and XML Signature Syntax and Processing

2864 SAML assertions and SAML protocol request and response messages may be signed, with the following  
2865 benefits:

- 2866 • An assertion signed by the SAML authority supports:
  - 2867 – Assertion integrity.
  - 2868 – Authentication of the SAML authority to a SAML relying party.
  - 2869 – If the signature is based on the SAML authority's public-private key pair, then it also provides for  
2870 non-repudiation of origin.
- 2871 • A SAML protocol request or response message signed by the message originator supports:
  - 2872 – Message integrity.
  - 2873 – Authentication of message origin to a destination.
  - 2874 – If the signature is based on the originator's public-private key pair, then it also provides for non-  
2875 repudiation of origin.

2876 A digital signature is not always required in SAML. For example, it may not be required in the following  
2877 situations:

- 2878 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains  
2879 protection from a signature on the containing protocol response message. "Inherited" signatures  
2880 should be used with care when the contained object (such as the assertion) is intended to have a  
2881 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,  
2882 exposing the XML content and adding potentially unnecessary overhead.
- 2883 • The SAML relying party or SAML requester may have obtained an assertion or protocol message  
2884 from the SAML authority or SAML responder directly (with no intermediaries) through a secure  
2885 channel, with the SAML authority or SAML responder having authenticated to the relying party or  
2886 SAML responder by some means other than a digital signature.

2887 Many different techniques are available for "direct" authentication and secure channel establishment  
2888 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
2889 the applicable security requirements depend on the communicating applications and the nature of the  
2890 assertion or message transported.

2891 It is recommended that, in all other contexts, digital signatures be used for assertions and request and  
2892 response messages. Specifically:

- 2893 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority  
2894 SHOULD be signed by the SAML authority.
- 2895 • A SAML protocol message arriving at a destination from an entity other than the originating site  
2896 SHOULD be signed by the origin site.

2897 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that  
2898 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures  
2899 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure  
2900 compatibility with profiles that may require other mechanisms.

2901 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST  
2902 be used if signing.

## 2903 **5.1 Signing Assertions**

2904 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema  
2905 as described in Section Assertions.

## 2906 **5.2 Request/Response Signing**

2907 All SAML protocol request and response messages MAY be signed using the XML Signature. This is  
2908 reflected in the schema as described in Sections Requests and Responses and Responses.

## 2909 **5.3 Signature Inheritance**

2910 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
2911 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a  
2912 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
2913 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,  
2914 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
2915 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
2916 and signature options.

2917 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
2918 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may  
2919 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
2920 information from the surrounding context, but no such inheritance should be inferred unless specifically  
2921 identified by the profile.

## 2922 **5.4 XML Signature Profile**

2923 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
2924 and many choices. This section details the constraints on these facilities so that SAML processors do not  
2925 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
2926 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply: the  
2927 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the  
2928 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section  
2929 as the identifier attributes.

### 2930 **5.4.1 Signing Formats and Algorithms**

2931 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
2932 detached.

2933 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
2934 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
2935 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2936 **5.4.2 References**

2937 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the  
2938 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element  
2939 may or may not be the root element of the actual XML document containing the signed assertion or  
2940 message.

2941 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier  
2942 attribute value of the root element of the message being signed. For example, if the attribute value is  
2943 "foo", then the URI attribute in the `<ds:Reference>` element MUST be "#foo".

### 2944 **5.4.3 Canonicalization Method**

2945 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
2946 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`  
2947 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages  
2948 embedded in an XML context can be verified independent of that context.

### 2949 **5.4.4 Transforms**

2950 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
2951 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
2952 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2953 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2954 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2955 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This  
2956 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2957 applying the transforms manually to the content and reverifying the result as consisting of the same  
2958 SAML message.

### 2959 **5.4.5 KeyInfo**

2960 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
2961 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
2962 be absent.

### 2963 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2964 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section  
2965 SAML Assertions.

### 2966 **5.4.7 Interoperability with SAML V1.0**

2967 ~~The use of XML Signature [XMLSig] described above is incompatible with the usage described in the~~  
2968 ~~SAML V1.0 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to~~  
2969 ~~ensure interoperability. It was constrained by the inability to use URI references to identify the SAML~~  
2970 ~~content to be signed. With this limitation removed by the addition of SAML identifier attributes, a decision~~  
2971 ~~has been made to forgo backwards compatibility with the older specification in this respect.~~

### 2972 **5.4.8 Example**

2973 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2974 added for readability; the signatures are not valid and cannot be successfully verified.

```
2975 <Response  
2976   IssueInstant="2003-04-17T00:46:02Z"  
2977   MajorVersion="1"  
2978   MinorVersion="1"  
2979   Recipient="www.opensaml.org"
```







3113

## 6 SAML Extensions

3114 The SAML schemas support extensibility. An example of an application that extends SAML assertions is  
3115 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use  
3116 the extensibility features in SAML to create extension schemas.

3117 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML  
3118 elements can serve as the head element of a substitution group. However, SAML types are not defined  
3119 as final, so that all SAML types MAY be extended and restricted. The following sections discuss only  
3120 elements and types not blocked from substitution, so that all SAML elements MAY serve as the head  
3121 element of a substitution group. Also, types are not defined as final, so that all SAML types MAY be  
3122 extended and restricted. The following sections discuss only elements that have been specifically  
3123 designed to support extensibility.

3124

### 6.1 Assertion Schema Extension

3125 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
3126 statements it contains, if the extension mechanism is used for either part.

3127 The following elements are intended specifically for use as extension points in an extension schema;  
3128 their types are set to `abstract`, and are thus usable only as the base of a derived type:

3129 • `<Condition>`

3130 • `<Statement>`

3131 • ~~`<SubjectStatement>`~~

3132 • The following elements that are directly usable as part of SAML MAY be extended:

3133 • `<AuthenticationStatement>`

3134 • `<AuthorizationDecisionStatement>`

3135 • `<AttributeStatement>`

3136 • `<AudienceRestrictionCondition>`

3137 The following elements are defined to allow elements from arbitrary namespaces within them, which  
3138 serves as a built-in extension point without requiring an extension schema:

3139 • `<BaseIdentifier>`

3140 • `<SubjectConfirmationData>`

3141 • `<AttributeValue>`

3142 • `<Advice>`

3143 • `<AuthnContext>`

3144

### 6.2 Protocol Schema Extension

3145 The following SAML protocol elements are intended specifically for use as extension points in an  
3146 extension schema; their types are set to `abstract`, and are thus usable only as the base of a derived  
3147 type:

- 3148 • <Query>
- 3149 • <SubjectQuery>

3150 The following elements that are directly usable as part of SAML MAY be extended:

- 3151 • <Request>
- 3152 • <AuthenticationQuery>
- 3153 • <AuthorizationDecisionQuery>
- 3154 • <AttributeQuery>
- 3155 • <Response>

### 3156 **6.3 Use of Type Derivation and Substitution Groups**

3157 ~~W3C XML Schema provides two principal mechanisms for specifying an element of an extended type:~~  
 3158 ~~type derivation and substitution groups.~~

3159 ~~For example, a <Statement> element can be assigned the type **NewStatementType** by means of the~~  
 3160 ~~`xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived~~  
 3161 ~~from **StatementType**. The following example of a SAML assertion assumes that the extension schema~~  
 3162 ~~(represented by the `new:` prefix) has defined this new type:~~

```
3163 <saml:Assertion ...>  

  3164 <saml:Statement xsi:type="new:NewStatementType">  

  3165 ...  

  3166 </saml:Statement>  

  3167 </saml:Assertion>
```

3168 ~~Alternatively, the extension schema can define a <NewStatement> element that is a member of a~~  
 3169 ~~substitution group that has <Statement> as a head element. For the substituted element to be schema-~~  
 3170 ~~valid, it needs to have a type that matches or is derived from the head element's type. The following is an~~  
 3171 ~~example of an extension schema fragment that defines this new element:~~

```
3172 <xsd:element "NewStatement" type="new:NewStatementType"  

  3173 substitutionGroup="saml:Statement"/>
```

3174 ~~The substitution group declaration allows the <NewStatement> element to be used anywhere the SAML~~  
 3175 ~~<Statement> element can be used. The following is an example of a SAML assertion that uses the~~  
 3176 ~~extension element:~~

```
3177 <saml:Assertion ...>  

  3178 <new:NewStatement>  

  3179 ...  

  3180 </new:NewStatement>  

  3181 </saml:Assertion>
```

3182 ~~The choice of extension method has no effect on the semantics of the XML document but does have~~  
 3183 ~~implications for interoperability.~~

3184 ~~The advantages of type derivation are as follows:~~

- 3185 • ~~A document can be more fully interpreted by a parser that does not have access to the extension~~  
 3186 ~~schema because a "native" SAML element is available.~~
- 3187 • ~~At the time of this writing, some W3C XML Schema validators do not support substitution groups,~~  
 3188 ~~whereas the `xsi:type` attribute is widely supported.~~

3189 ~~The advantage of substitution groups is that a document can be explained without the need to explain~~  
 3190 ~~the functioning of the `xsi:type` attribute.~~

3191

## 7 SAML-Defined Identifiers

3192 The following sections define URI-based identifiers for common authentication methods, resource access  
3193 actions, and subject name identifier formats.

3194 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of  
3195 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
3196 have one of the following stems:

```
3197 urn:oasis:names:tc:SAML:1.0:  
3198 urn:oasis:names:tc:SAML:1.1:
```

### 7.1 Authentication Method Identifiers

3200 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the  
3201 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although  
3202 both can refer to the same underlying mechanisms. An authentication statement with an  
3203 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The  
3204 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the  
3205 authentication statement does not provide the means to perform that authentication, such as a password,  
3206 key, or certificate.

3207 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,  
3208 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML  
3209 relying party to confirm that the request or message came from a system entity that corresponds to the  
3210 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method  
3211 that the relying party can use to do this in the future. This may or may not have any relationship to an  
3212 authentication that was performed previously. Unlike the authentication method, the subject confirmation  
3213 method may be accompanied by some piece of information, such as a certificate or key, that will allow  
3214 the relying party to perform the necessary check.

3215 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML  
3216 ~~bindings and~~ profiles specification [SAMLProf] for more information. Additional methods may be added  
3217 by defining new profiles or by private agreement.

3218 The following identifiers refer to SAML-specified authentication methods.

#### 7.1.1 Password

3220 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

3221 The authentication was performed by means of a password.

#### 7.1.2 Kerberos

3223 **URI:** urn:ietf:rfc:1510

3224 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the  
3225 Needham-Schroeder symmetric key authentication mechanism [Needham78].

#### 7.1.3 Secure Remote Password (SRP)

3227 **URI:** urn:ietf:rfc:2945

3228 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC  
3229 2945].

#### 3230 **7.1.4 Hardware Token**

3231 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

3232 The authentication was performed using some (unspecified) hardware token.

#### 3233 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

3234 **URI:** urn:ietf:rfc:2246

3235 The authentication was performed using either the SSL or TLS protocol with certificate-based client  
3236 authentication. TLS is described in [RFC 2246].

#### 3237 **7.1.6 X.509 Public Key**

3238 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

3239 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
3240 of an X.509 PKI [X.500][PKIX]. It may have been one of the mechanisms for which a more specific  
3241 identifier has been defined below.

#### 3242 **7.1.7 PGP Public Key**

3243 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

3244 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
3245 of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific  
3246 identifier has been defined below.

#### 3247 **7.1.8 SPKI Public Key**

3248 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

3249 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
3250 of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has  
3251 been defined below.

#### 3252 **7.1.9 XKMS Public Key**

3253 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

3254 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
3255 of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific  
3256 identifier has been defined below.

#### 3257 **7.1.10 XML Digital Signature**

3258 **URI:** urn:ietf:rfc:3075

3259 The authentication was performed by means of an XML digital signature [RFC 3075].

3260 | **7.1.11 Authentication Context**

3261 | **URI:** <urn:oasis:names:tc:SAML:2.0:am:authncontext>

3262 | The authentication method is described by the proximal <AuthnContext> element.

3263 | **7.1.12 Unspecified**

3264 | **URI:** <urn:oasis:names:tc:SAML:1.0:am:unspecified>

3265 | The authentication was performed by an unspecified means.

3266 | **7.2 Action Namespace Identifiers**

3267 | The following identifiers MAY be used in the `Namespace` attribute of the <Action> element (see  
3268 | Section Element <Action>) to refer to common sets of actions to perform on resources.

3269 | **7.2.1 Read/Write/Execute/Delete/Control**

3270 | **URI:** <urn:oasis:names:tc:SAML:1.0:action:rwedc>

3271 | Defined actions:

3272 |     Read Write Execute Delete Control

3273 | These actions are interpreted as follows:

3274 | Read

3275 |     The subject may read the resource.

3276 | Write

3277 |     The subject may modify the resource.

3278 | Execute

3279 |     The subject may execute the resource.

3280 | Delete

3281 |     The subject may delete the resource.

3282 | Control

3283 |     The subject may specify the access control policy for the resource.

3284 | **7.2.2 Read/Write/Execute/Delete/Control with Negation**

3285 | **URI:** <urn:oasis:names:tc:SAML:1.0:action:rwedc-negation>

3286 | Defined actions:

3287 |     Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

3288 | The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner  
3289 | described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively  
3290 | specify that the stated permission is denied. Thus a subject described as being authorized to perform the  
3291 | action ~Read is affirmatively denied read permission.

3292 A SAML authority MUST NOT authorize both an action and its negated form.

### 3293 **7.2.3 Get/Head/Put/Post**

3294 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3295 Defined actions:

3296 GET HEAD PUT POST

3297 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
3298 the GET action on a resource is authorized to retrieve it.

3299 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and  
3300 POST actions to the write permission. The correspondence is not exact however since an HTTP GET  
3301 operation may cause data to be modified and a POST operation may cause modification to a resource  
3302 other than the one specified in the request. For this reason a separate Action URI reference specifier is  
3303 provided.

### 3304 **7.2.4 UNIX File Permissions**

3305 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3306 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
3307 notation.

3308 The action string is a four-digit numeric code:

3309 *extended user group world*

3310 Where the *extended* access permission has the value

3311 +2 if sgid is set

3312 +4 if suid is set

3313 The *user group* and *world* access permissions have the value

3314 +1 if execute permission is granted

3315 +2 if write permission is granted

3316 +4 if read permission is granted

3317 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read  
3318 and execute; and world read.

## 3319 **7.3 NameIdentifier Format Identifiers**

3320 The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see  
3321 Section Element `<NameIdentifier>`) to refer to common formats for the content of the  
3322 `<NameIdentifier>` element and the associated processing rules, if any.

3323 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of  
3324 SAML.



### 3325 **7.3.1 Unspecified**

3326 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

3327 The interpretation of the content of the element is left to individual implementations.

### 3328 **7.3.2 Email Address**

3329 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

3330 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as  
3331 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that  
3332 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in  
3333 parentheses) after it, and is not surrounded by "<" and ">".

### 3334 **7.3.3 X.509 Subject Name**

3335 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3336 Indicates that the content of the element is in the form specified for the contents of the  
3337 <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors  
3338 should note that the XML Signature specification specifies encoding rules for X.509 subject names that  
3339 differ from the rules given in IETF RFC 2253 [RFC 2253].

### 3340 **7.3.4 Windows Domain Qualified Name**

3341 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3342 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
3343 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"  
3344 separator MAY be omitted.

### 3345 **7.3.5 Provider Identifier**

3346 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:provider

3347 Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a  
3348 SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser  
3349 profiles). Such an identifier can be used to make assertions about system entities that can issue SAML  
3350 requests, responses, and assertions.

### 3351 **7.3.6 Federated Identifier**

3352 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:federated

3353 Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity  
3354 federation between an identity provider and a service provider (or affiliation of service providers).  
3355 Federated name identifiers generated by identity providers MUST be constructed using pseudo-random  
3356 values that have no discernible correspondence with the subject's actual identifier (for example,  
3357 username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's  
3358 identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

3359 The element's content MUST contain the most recent identifier of the subject set by the identity provider.

3360 The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider  
3361 participating in the identity federation. It MAY be omitted if the value can be derived from the context of  
3362 the message containing the element, such as the issuer of an assertion.

3363 The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider  
3364 or affiliation of providers participating in the identity federation. It MAY be omitted if the element is  
3365 contained in a message intended only for consumption directly by the service provider, and the value  
3366 would be the name of that service provider.

3367 The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject  
3368 most recently set by the service provider or affiliation, if any. If no such identifier has been established,  
3369 than the attribute MUST be omitted.

3370 Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear  
3371 text with providers other than the providers that have established the identity federation. Furthermore,  
3372 they MUST NOT appear in log files or similar locations without appropriate controls and protections.  
3373 Deployments without such requirements are free to use other kinds of identifiers in their SAML  
3374 exchanges.

3375 Note also that while federated identifiers are typically used to reflect an account linking relationship  
3376 between a pair of providers, a service provider is not obligated to recognize or make use of the long term  
3377 nature of the persistent identifier or establish such a link. Such a "one-sided" identity federation is not  
3378 discernibly different and does not affect the behavior of the identity provider or any processing rules  
3379 specific to federated identifiers in the protocols defined in this specification.

### 3380 **7.3.7 Transient Identifier**

3381 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

3382 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated  
3383 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated  
3384 in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of  
3385 256 characters.

3386 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier  
3387 represents a transient and temporary identity federation, as described in Section Federated Identifier. In  
3388 such a case, they MAY be omitted in accordance with the rules specified in that section.

## 3389 **7.4 Attribute NameFormat Identifiers**

3390 The following identifiers MAY be used in the `NameFormat` attribute defined on the  
3391 **AttributeDesignatorType** complex type (see Section x) to refer to the classification of the attribute name  
3392 for purposes of interpreting the name.

### 3393 **7.4.1 Unspecified**

3394 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:unspecified

3395 The interpretation of the attribute name is left to individual implementations.

### 3396 **7.4.2 URI Reference**

3397 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:uri

3398 | The attribute name follows the convention for URI references [RFC 2396], for example as used in  
3399 | XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is  
3400 | application-specific.

## 3401 | **7.5 Attribute ValueType Identifiers**

3402 | The following identifier MAY be used in the `ValueType` attribute defined on the  
3403 | **AttributeDesignatorType** complex type (see Section x) to refer to the URI-based datatype of the  
3404 | desired or supplied attribute.

### 3405 | **7.5.1 Application-Specific Value Type**

3406 | **URI:** urn:oasis:names:tc:SAML:2.0:valuetype-format:appSpecific

3407 | Indicates that the datatype of the desired or supplied attribute is application-specific. Note that any  
3408 | `ValueType` setting (default or explicit) in an attribute query, including this setting, needs to be exactly  
3409 | matched (in addition to other exact matches) in order for an attribute to be returned.

3410 |

## 8 References

The following works are cited in the body of this specification.

### 8.1 Normative References

- [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/>.
- [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- [Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- [XML] T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- [XMLEnc] D. Eastlake et al., XML Encryption Syntax and Processing, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web Consortium. Note that this specification normatively references [XMLEnc-XSD], listed below.
- [XMLEnc-XSD] XML Encryption Schema. World Wide Web Consortium. <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- [XMLNS] T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January 1999. <http://www.w3.org/TR/REC-xml-names>.
- [XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*. World Wide Web Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this specification normatively references [XMLSig-XSD], listed below.
- [XMLSig-XSD] XML Signature Schema. World Wide Web Consortium. <http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd>.

### 8.2 Non-Normative References

- [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](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- [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.
- [PGP] Atkins, D., Stallings, W. and P. Zimmermann. *PGP Message Exchange Formats*. IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- [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>.
- [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- [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>.

3454 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January  
3455 1999. <http://www.ietf.org/rfc/rfc2246.txt>.

3456 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*  
3457 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.  
3458 <http://www.ietf.org/rfc/rfc2253.txt>.

3459 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF  
3460 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3461 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.  
3462 <http://www.ietf.org/rfc/rfc2630.txt>.

3463 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.  
3464 <http://www.ietf.org/rfc/rfc2822.txt>.

3465 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,  
3466 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3467 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF  
3468 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3469 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*  
3470 *(SAML)*. OASIS, September 2003. Document ID [oasis-sstc-saml-bindings-](http://www.oasis-open.org/committees/security/)  
3471 [2.0and Profiles for the OASIS Security Assertion Markup Language \(SAML\)](http://www.oasis-open.org/committees/security/).  
3472 OASIS, September 2003. Document ID [oasis-sste-saml-bindings-1.1](http://www.oasis-open.org/committees/security/).  
3473 <http://www.oasis-open.org/committees/security/>.

3474 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*  
3475 *(SAML)*. OASIS, September 2003. Document ID [oasis-sstc-saml-profiles-2.0](http://www.oasis-open.org/committees/security/).  
3476 <http://www.oasis-open.org/committees/security/>.

3477 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*  
3478 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3479 [oasis-sstc-saml-conform-1.1](http://www.oasis-open.org/committees/security/). HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>3480 open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3481 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*  
3482 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-  
open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-<br/>3483 open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3484 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*  
3485 *(SAML)*. OASIS, September 2003. Document ID [oasis-sstc-saml-glossary-1.1](http://www.oasis-<br/>3486 open.org/committees/security/).  
3487 HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>open.org/committees/security/)"[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>open.org/committees/security/).

3488 [SAMLXSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID  
3489 [oasis-sstc-saml-schema-protocol-1.1](http://www.oasis-<br/>3490 open.org/committees/security/). HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3491 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*  
3492 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3493 [oasis-sstc-saml-sec-consider-1.1](http://www.oasis-<br/>3494 open.org/committees/security/). HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3495 [SAMLXSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID  
3496 [oasis-sstc-saml-schema-assertion-1.1](http://www.oasis-<br/>3497 open.org/committees/security/). HYPERLINK "[http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3498 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
3499 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.

3500 [Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
3501 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.

3502	<b>[SPKI]</b>	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999. <a href="http://www.ietf.org/rfc/rfc2693.txt">http://www.ietf.org/rfc/rfc2693.txt</a> .
3503		
3504		
3505	<b>[UNICODE-C]</b>	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a> .
3506		
3507	<b>[W3C-CHAR]</b>	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a> .
3508		
3509	<b>[W3C-CharMod]</b>	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. <a href="http://www.w3.org/TR/charmod/">http://www.w3.org/TR/charmod/</a> .
3510		
3511	<b>[X.500]</b>	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
3512		
3513	<del><b>[XACML]</b></del>	<del>eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml</a>.</del>
3514		
3515		
3516	<b>[XKMS]</b>	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. <a href="http://www.w3.org/TR/xkms/">http://www.w3.org/TR/xkms/</a> .
3517		
3518		
3519	<del><b>[XML]</b></del>	<del>T. Bray, et al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i>. World Wide Web Consortium, October 2000. <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a>.</del>
3520		
3521	<del><b>[XMLEnc]</b></del>	<del>D. Eastlake et al., <i>XML Encryption Syntax and Processing</i>, <a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/</a>, World Wide Web Consortium.</del>
3522		
3523		
3524	<del><b>[XMLEnc-XSD]</b></del>	<del>XML Encryption Schema. World Wide Web Consortium. <a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd</a>.</del>
3525		
3526	<del><b>[XMLSig]</b></del>	<del>D. Eastlake et al., <i>XML Signature Syntax and Processing</i>, World Wide Web Consortium, February 2002. <a href="http://www.w3.org/TR/xmldsig-core/">http://www.w3.org/TR/xmldsig-core/</a>.</del>
3527		
3528	<del><b>[XMLSig-XSD]</b></del>	<del>XML Signature Schema. World Wide Web Consortium. <a href="http://www.w3.org/TR/2000/GR-xmldsig-core-20001031/xmldsig-core-schema.xsd">http://www.w3.org/TR/2000/GR-xmldsig-core-20001031/xmldsig-core-schema.xsd</a>.</del>
3529		
3530		

3531

---

## Appendix A. Acknowledgments

3532

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

3533

3534

- @@

## Appendix B. Revision History

Rev	Date	By Whom	What
<a href="#">01</a>	<a href="#">20 Oct 2003</a>	<a href="#">Eve Maler</a>	Initial draft. Converted to OpenOffice. <b>CORE-1</b> through <b>CORE-4</b> . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed.
<a href="http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf">http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf</a>			
<a href="#">02</a>	<a href="#">4 Jan 2004</a>	<a href="#">Eve Maler</a>	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal ( <a href="http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587">http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587</a> ) for work item <b>W-2</b> , Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet).  Fixed <b>CORE-10</b> (the description of subelement occurrence in the <code>&lt;Evidence&gt;</code> element).
<a href="http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf">http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf</a>			
<a href="#">03</a>	<a href="#">24 Jan 2004</a>	<a href="#">Scott Cantor</a>	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars.
<a href="http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf">http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf</a> <a href="http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf">http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf</a>			
<a href="#">04</a>	<a href="#">1 Feb 2004</a>	<a href="#">Eve Maler</a>	Made minor edits to new and existing material: changed new <code>&lt;AssertionRequest&gt;</code> element name to <code>&lt;AssertionIDRequest&gt;</code> ; changed new <code>&lt;AssertionArtifact&gt;</code> and <code>&lt;NewIdentifier&gt;</code> element declarations from local to global; made distinction between normative and non-normative references; implemented the blocking of element substitution. The bulk of work item <b>W-2</b> , Identity Federation, is now reflected here. What remains is the federation termination protocol, plus a few other pieces that are covered under other work items.
<a href="http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf">http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf</a>			
<a href="#">05</a>	<a href="#">17 Feb 2004</a>	<a href="#">Scott Cantor, John Kemp, Eve Maler</a>	Added FedTerm protocol ( <b>W-2</b> ), removed NameID date attributes, clarified Name Reg processing rules, added Extensions facility and Consent attribute. Also moved Signature on assertions to a location consistent with Request and Response. Added session protocol material ( <b>W-1</b> ); still unfinished.
<a href="http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf">http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf</a>			
<a href="#">06</a>	<a href="#">20 Feb 2004</a>	<a href="#">Scott Cantor, John Kemp, Eve Maler</a>	Added AssertionURIReference ( <b>W-19</b> ), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response ( <b>W-1</b> ). Implemented the freezing of authZ decision statement functionality ( <b>W-28b</b> ).
<a href="http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf">http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf</a>			



Rev	Date	By Whom	What
<u>07</u>	<u>7 Mar 2004</u>	<u>Scott Cantor, Eve Maler</u>	<p>Implemented new arrangement for subject information and decision on KeyInfo description, as agreed at 2 Mar 2004 telecon.</p> <p>Adjusted normative language around subject "matching" rules based on subject changes.</p> <p>Revised AuthnRequest proposal based on those changes and feedback from list and focus calls.</p> <p>Incorporated additional schema and processing rules related to ECP and proxying use cases from ID-FF.</p> <p>Added AuthnContext to AuthenticationStatement.</p> <p>Added NameIdentifierMapping protocol (<b>W-2</b>).</p>
			<u>00</u>
<u>08</u>	<u>15 Mar 2004</u>	<u>Scott Cantor, Eve Maler</u>	<p>Added ArtifactRequest/Response pair as a new protocol.</p> <p>Implemented proposed W-28a attribute changes (rev 03 of the proposal, reflecting focus group input).</p>
Rev	Date	By Whom	What
<u>01</u>	<u>20-Oct-2003</u>	<u>Eve Maler</u>	Initial draft. Converted to OpenOffice. <b>CORE-1</b> through <b>CORE-4</b> . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed.
<u>02</u>	<u>4-Jan-2004</u>	<u>Eve Maler</u>	<p>Implemented Scott Cantor's draft sstc-nameid-07 solution proposal (<a href="http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587">http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587</a>) for work item <b>W-2</b>, Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet).</p> <p>Fixed <b>CORE-10</b> (the description of subelement occurrence in the &lt;Evidence&gt; element).</p>

3536

3537

---

## Appendix C. Notices

3538 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
3539 might be claimed to pertain to the implementation or use of the technology described in this document or  
3540 the extent to which any license under such rights might or might not be available; neither does it  
3541 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
3542 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
3543 made available for publication and any assurances of licenses to be made available, or the result of an  
3544 attempt made to obtain a general license or permission for the use of such proprietary rights by  
3545 implementors or users of this specification, can be obtained from the OASIS Executive Director.

3546 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
3547 or other proprietary rights which may cover technology that may be required to implement this  
3548 specification. Please address the information to the OASIS Executive Director.

3549 **Copyright © OASIS Open 2004. All Rights Reserved.**

3550 This document and translations of it may be copied and furnished to others, and derivative works that  
3551 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
3552 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
3553 notice and this paragraph are included on all such copies and derivative works. However, this document  
3554 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
3555 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
3556 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required  
3557 to translate it into languages other than English.

3558 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
3559 or assigns.

3560 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3561 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
3562 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS  
3563 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR  
3564 PURPOSE.