



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

Working Draft **1009**, **128** April 2004

Document identifier:

sstc-saml-core-2.0-draft-**1009**

Location:

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

Editors:

Scott Cantor, individual (cantor.2@osu.edu)
John Kemp, Nokia (john.kemp@nokia.com)
Eve Maler, Sun Microsystems (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
John Hughes, Entegriety
Carlisle Adams, Entrust
Tim Moses, Entrust
Nigel Edwards, Hewlett-Packard
Joe Pato, Hewlett-Packard
Bob Blakley, IBM
Marlena Erdos, IBM
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-
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 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-
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 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
79	<SubjectConfirmationData>.....	16
80	2.4.3.3 Element <Conditions>.....	16
81	2.4.3.3.1 Attributes NotBefore and NotOnOrAfter.....	18
82	2.4.3.3.2 Element <Condition>.....	18
83	2.4.3.3.3 Elements <AudienceRestrictionCondition> and <Audience>.....	18
84	2.4.3.3.4 Element <DoNotCacheCondition>.....	19
85	2.4.3.3.5 Element <ProxyRestrictionCondition>.....	19
86	2.4.3.4 Element <Advice>.....	20
87	2.5 Statements.....	21
88	2.5.1 Element <Statement>.....	21
89	2.5.1.1 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
90	<SubjectConfirmationData>.....	21
91	2.5.2 Element <AuthenticationStatement>.....	22
92	2.5.2.1 Element <SubjectLocality>.....	23
93	2.5.2.2 Element <AuthnContext>.....	23
94	2.5.3 Element <AttributeStatement>.....	24
95	2.5.3.1 Elements <AttributeDesignator> and <Attribute>.....	24
96	2.5.3.1.1 Element <AttributeValue>.....	26
97	2.5.4 Element <AuthorizationDecisionStatement>.....	26
98	2.5.4.1 Element <Action>.....	27
99	2.5.4.2 Element <Evidence>.....	28
100	3 SAML Protocols.....	29
101	3.1 Schema Header and Namespace Declarations.....	29
102	3.2 Requests and Responses.....	30
103	3.2.1 Complex Type RequestAbstractType.....	30
104	3.2.1.1 Element <RelayState>.....	31

105	3.2.2 Complex Type StatusResponseType.....	31
106	3.2.2.1 Element <Status>.....	32
107	3.2.2.2 Element <StatusCode>.....	33
108	3.2.2.3 Element <StatusMessage>.....	35
109	3.2.2.4 Element <StatusDetail>.....	35
110	3.3 Assertion Query and Request Protocol.....	35
111	3.3.1 Element <AssertionIDRequest>.....	35
112	3.3.2 Queries.....	36
113	3.3.2.1 Element <SubjectQuery>.....	36
114	3.3.2.2 Element <AuthenticationQuery>.....	36
115	3.3.2.3 Element <AttributeQuery>.....	37
116	3.3.2.4 Element <AuthorizationDecisionQuery>.....	38
117	3.3.3 Element <Response>.....	39
118	3.3.3.1 Processing Rules.....	39
119	3.4 Authentication Request Protocol.....	40
120	3.4.1 Element <AuthnRequest>.....	40
121	3.4.1.1 Element <NameIDPolicy>.....	42
122	3.4.1.2 Element <RequestAuthnContext>.....	43
123	3.4.1.3 Element <Scoping>.....	44
124	3.4.1.4 Element <IDPList>.....	45
125	3.4.1.5 Element <IDPEntry>.....	45
126	3.4.1.6 Processing Rules.....	46
127	3.4.1.7 Proxying.....	47
128	3.4.1.7.1 Processing Rules.....	47
129	3.5 Artifact Protocol.....	48
130	3.5.1 Element <ArtifactRequest>.....	48
131	3.5.2 Element <ArtifactResponse>.....	49
132	3.5.3 Processing Rules.....	50
133	3.6 Federated Name Identifier RegistrationManagement Protocol.....	50
134	3.6.1 Element <RegisterManageNameIdentifierRequest>.....	51
135	3.6.2 Element <RegisterManageNameIdentifierResponse>.....	52
136	3.6.3 Processing Rules.....	52
137	3.7 Federation Termination Protocol.....	53
138	3.7.1 Element <FederationTerminationNotification>.....	53
139	3.7.2 Element <FederationTerminationResponse>.....	54
140	3.7.3 Processing Rules.....	54
141	3.8 Single Logout Protocol.....	54
142	3.8.1 Element <LogoutRequest>.....	55
143	3.8.2 Element <LogoutResponse>.....	55
144	3.8.3 Processing Rules.....	56
145	3.8.3.1 Session Participant Rules.....	56
146	3.8.3.2 Session Authority Rules.....	56
147	3.9 Name Identifier Mapping Protocol.....	57
148	3.9.1 Element <NameIdentifierMappingRequest>.....	58
149	3.9.2 Element <NameIdentifierMappingResponse>.....	58
150	3.9.3 Processing Rules.....	59
151	4 SAML Versioning.....	60
152	4.1 SAML Specification Set Version.....	60
153	4.1.1 Schema Version.....	60
154	4.1.2 SAML Assertion Version.....	60
155	4.1.3 SAML Protocol Version.....	61
156	4.1.3.1 Request Version.....	61

157	4.1.4 Response Version.....	61
158	4.1.5 Permissible Version Combinations.....	62
159	4.2 SAML Namespace Version.....	62
160	4.2.1 Schema Evolution.....	62
161	5 SAML and XML Signature Syntax and Processing.....	63
162	5.1 Signing Assertions.....	64
163	5.2 Request/Response Signing.....	64
164	5.3 Signature Inheritance.....	64
165	5.4 XML Signature Profile.....	64
166	5.4.1 Signing Formats and Algorithms.....	64
167	5.4.2 References.....	64
168	5.4.3 Canonicalization Method.....	65
169	5.4.4 Transforms.....	65
170	5.4.5 KeyInfo.....	65
171	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	65
172	5.4.7 Example.....	65
173	6 SAML Extensions.....	68
174	6.1 Assertion Schema Extension.....	68
175	6.2 Protocol Schema Extension.....	68
176	7 SAML-Defined Identifiers.....	70
177	7.1 Authentication Method Identifiers.....	70
178	7.1.1 Password.....	70
179	7.1.2 Kerberos	70
180	7.1.3 Secure Remote Password (SRP).....	70
181	7.1.4 Hardware Token.....	71
182	7.1.5 SSL/TLS Certificate Based Client Authentication:.....	71
183	7.1.6 X.509 Public Key	71
184	7.1.7 PGP Public Key	71
185	7.1.8 SPKI Public Key	71
186	7.1.9 XKMS Public Key	71
187	7.1.10 XML Digital Signature	71
188	7.1.11 Authentication Context.....	72
189	7.1.12 Unspecified	72
190	7.2 Action Namespace Identifiers.....	72
191	7.2.1 Read/Write/Execute/Delete/Control.....	72
192	7.2.2 Read/Write/Execute/Delete/Control with Negation.....	72
193	7.2.3 Get/Head/Put/Post.....	73
194	7.2.4 UNIX File Permissions.....	73
195	7.3 NameIdentifier Format Identifiers.....	73
196	7.3.1 Unspecified.....	74
197	7.3.2 Email Address.....	74
198	7.3.3 X.509 Subject Name.....	74
199	7.3.4 Windows Domain Qualified Name.....	74
200	7.3.5 Kerberos Principal Name.....	74
201	7.3.6 Provider Identifier.....	74
202	7.3.7 Federated Identifier.....	75
203	7.3.8 Transient Identifier.....	75
204	7.4 Attribute NameFormat Identifiers.....	76
205	7.4.1 Unspecified.....	76
206	7.4.2 URI Reference.....	76

207	7.5 Attribute ValueType Identifiers.....	76
208	7.5.1 Unspecified Value Type.....	76
209	8 References.....	77
210	8.1 Normative References.....	77
211	8.2 Non-Normative References.....	77
212		

213

1 Introduction

214 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)
215 assertions and the protocols for requesting and returning them. SAML assertions, requests, and
216 responses are encoded in XML [XML]and use XML namespaces [XMLNS]. They are typically embedded
217 in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The
218 SAML specification for bindings [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

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 keywords "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 documents take precedence.

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

- 241 • The prefix `saml:` stands for the SAML assertion namespace,
242 `urn:oasis:names:tc:SAML:2.0:assertion`.
- 243 • The prefix `samlp:` stands for the SAML request-response protocol namespace,
244 `urn:oasis:names:tc:SAML:2.0:protocol`.
- 245 • The prefix `ds:` stands for the W3C XML Signature namespace,
246 <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 247 • The prefix `xenc:` stands for the W3C XML Encryption namespace,
248 <http://www.w3.org/2001/04/xmlenc#> [XMLEnc-XSD].
- 249 • The prefix `xsd:` stands for the W3C XML Schema namespace,
250 <http://www.w3.org/2001/XMLSchema> [Schema1], in example listings. In schema listings, this is
251 the default namespace and no prefix is shown.

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

254 1.2 Schema Organization and Namespaces

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

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

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

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

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

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

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

265 1.2.1 String and URI Values

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

272 1.2.2 Time Values

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

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

277 1.2.3 ID and ID Reference Values

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

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

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

290 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that
291 **xsd>IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML
292 reference identifier might actually be defined in a document separate from that in which the identifier

293 reference is used, which violates the **xsd:IDREF** requirement that its value match the value of an ID
294 attribute on some element in the same XML document.

295 **1.2.4 Comparing SAML Values**

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

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

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

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

2 SAML Assertions

316

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

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

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

2.1 Schema Header and Namespace Declarations

330

331 The following schema fragment defines the XML namespaces and other header information for the
332 assertion schema:

```
333 <schema
334     targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
335     xmlns="http://www.w3.org/2001/XMLSchema"
336     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
337     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
338     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
339     elementFormDefault="unqualified"
340     attributeFormDefault="unqualified"
341     blockDefault="substitution"
342     version="2.0">
343     <import namespace="http://www.w3.org/2000/09/xmldsig#"
344           schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
345 schema.xsd"/>
346     <import namespace="http://www.w3.org/2001/04/xmlenc#"
347           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
348 schema.xsd"/>
349     <annotation>
350         <documentation>
351             Document identifier: sstc-saml-schema-assertion-2.0
352             Location: http://www.oasis-
353 open.org/committees/documents.php?wg_abbrev=security
354         </documentation>
355     </annotation>
356     ...
357 </schema>
```

2.2 Simple Types

358

359 The following section defines the SAML assertion-related simple types.

360 2.2.1 Simple Type DecisionType

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

363 Permit

364 The specified action is permitted.

365 Deny

366 The specified action is denied.

367 Indeterminate

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

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

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

```
373 <simpleType name="DecisionType">  
374   <restriction base="string">  
375     <enumeration value="Permit"/>  
376     <enumeration value="Deny"/>  
377     <enumeration value="Indeterminate"/>  
378   </restriction>  
379 </simpleType>
```

380 2.3 Name Identifiers

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

383 2.3.1 Element `<BaseIdentifier>`

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

387 NameQualifier [Optional]

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

390 SPNameQualifier [Optional]

391 Further qualifies an **federated** identifier with the name of the service provider or affiliation of
392 providers which has federated the principal's identity.

393 The following schema fragment defines the `<BaseIdentifier>` element and its **BaseIdentifierType**
394 complex type:

```
395 <element name="BaseIdentifier" type="saml:BaseIdentifierAbstractType"/>  
396 <complexType name="BaseIdentifierAbstractType" abstract="true">  
397   <complexContent>  
398     <extension base="anyType">  
399       <attribute name="NameQualifier" type="string" use="optional"/>  
400       <attribute name="SPNameQualifier" type="string" use="optional"/>  
401     </extension>
```

```
402     </complexContent>
403 </complexType>
```

404 2.3.2 Element <NameIdentifier>

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

407 Format [Optional]

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

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

418 SPProvidedIdentifier [Optional]

419 The name identifier established by the service provider or affiliation of providers for the principal,
420 if different from the primary name identifier given in the content of the <NameIdentifier>
421 element.

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

```
424 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
425 <complexType name="NameIdentifierType" mixed="false">
426   <simpleContent>
427     <restriction base="saml:BaseIdentifierAbstractType">
428       <simpleType>
429         <restriction base="string"/>
430       </simpleType>
431       <attribute name="Format" type="anyURI" use="optional"/>
432       <attribute name="SPProvidedIdentifier" type="string"
433 use="optional"/>
434     </restriction>
435   </simpleContent>
436 </complexType>
```

437 2.3.3 Element <EncryptedIdentifier>

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

442 <xenc:EncryptedData> [Required]

443 The encrypted content and associated encryption details, as defined by [the XML Encryption](#)
444 [Syntax and Processing specification \[XMLEnc\]](#). The encrypted content MUST contain an
445 element that has a type that is derived from **BaseIdentifierAbstractType** or from
446 **AssertionType**.

447 <xenc:EncryptedKey> [Zero or more]
448 Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a
449 Recipient attribute that specifies the entity for whom the key has been encrypted.

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

453 The following schema fragment defines the <EncryptedIdentifier> element and its
454 **EncryptedIdentifierType** complex type:

```
455     <element name="EncryptedIdentifier" type="saml:EncryptedIdentifierType"/>  
456     <complexType name="EncryptedIdentifierType" mixed="false">  
457       <complexContent>  
458         <restriction base="saml:BaseIdentifierType">  
459           <sequence>  
460             <element ref="xenc:EncryptedData"/>  
461             <element ref="xenc:EncryptedKey" minOccurs="0"  
462 maxOccurs="unbounded"/>  
463           </sequence>  
464         </restriction>  
465       </complexContent>  
466     </complexType>
```

467 2.3.4 Element <Issuer>

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

471 The following schema fragment defines the <Issuer> element:

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

473 2.4 Assertions

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

475 2.4.1 Element <AssertionIDReference>

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

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

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

481 2.4.2 Element <AssertionURIReference>

482 The <AssertionURIReference> element makes a reference to a SAML assertion by its uniform
483 resource identifier (URI). Dereferencing the URI (in a fashion dictated by the URI) is intended to produce
484 the assertion. See the Bindings specification [SAMLBind] for information on how this element is used in a
485 protocol binding.

486 The following schema fragment defines the <AssertionURIReference> element:

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

488 2.4.3 Element <Assertion>

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

491 MajorVersion [Required]

492 The major version of this assertion. The identifier for the version of SAML defined in this
493 specification is 2. SAML versioning is discussed in Section SAML Versioning.

494 MinorVersion [Required]

495 The minor version of this assertion. The identifier for the version of SAML defined in this
496 specification is 0. SAML versioning is discussed in Section SAML Versioning.

497 AssertionID [Required]

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

500 IssueInstant [Required]

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

502 <Issuer> [Required]

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

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

510 <ds:Signature> [Optional]

511 An XML Signature that authenticates the assertion, as described in Section SAML and XML
512 Signature Syntax and Processing.

513 <Subject> [RequiredOptional]

514 The subject of the statement(s) in the assertion.

515 <Conditions> [Optional]

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

517 <Advice> [Optional]

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

520 **OneZero** or more of the following statement elements:

521 <Statement>

522 A statement defined in an extension schema.

523 <AuthenticationStatement>

524 An authentication statement.

525 <AuthorizationDecisionStatement>

526 An authorization decision statement.

527 <AttributeStatement>

528 An attribute statement.

529 ~~An assertion with no statements MUST contain a <Subject> element, because the semantic of a~~
530 ~~statement-less assertion containing no statements is to binds the name identifier in the subject to a~~
531 ~~principal. Otherwise <Subject>, if present, identifies the subject of all of the statements in the~~
532 ~~assertion. If omitted, then the statements in the assertion are assumed to identify (implicitly or~~
533 ~~explicitly) the subject(s) or subjects to which whom they apply in an application- or profile-specific~~
534 ~~manner.~~

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

```
537 <element name="Assertion" type="saml:AssertionType"/>
538 <complexType name="AssertionType">
539   <sequence>
540     <element ref="saml:Issuer"/>
541     <element ref="ds:Signature" minOccurs="0"/>
542     <element ref="saml:Subject" minOccurs="0"/>
543     <element ref="saml:Conditions" minOccurs="0"/>
544     <element ref="saml:Advice" minOccurs="0"/>
545     <choice minOccurs="0" maxOccurs="unbounded">
546       <element ref="saml:Statement"/>
547       <element ref="saml:AuthenticationStatement"/>
548       <element ref="saml:AuthorizationDecisionStatement"/>
549       <element ref="saml:AttributeStatement"/>
550     </choice>
551   </sequence>
552   <attribute name="MajorVersion" type="integer" use="required"/>
553   <attribute name="MinorVersion" type="integer" use="required"/>
554   <attribute name="AssertionID" type="ID" use="required"/>
555   <attribute name="IssueInstant" type="dateTime" use="required"/>
556 </complexType>
```

557 2.4.3.1 Element <Subject>

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

561 <NameIdentifier>, <EncryptedIdentifier>, Or <BaseIdentifier>

562 Identifies the subject.

563 <SubjectConfirmation>

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

567 If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML
568 authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it
569 can treat the entity presenting the assertion to the relying party as the entity that the SAML authority
570 associates with the name identifier for the purposes of processing the assertion. A <Subject> element
571 SHOULD NOT identify more than one principal.

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

```
573 <element name="Subject" type="saml:SubjectType"/>
```

```

574 <complexType name="SubjectType">
575   <choice>
576     <sequence>
577       <choice>
578         <element ref="saml:BaseIdentifier"/>
579         <element ref="saml:NameIdentifier"/>
580         <element ref="saml:EncryptedIdentifier"/>
581       </choice>
582       <element ref="saml:SubjectConfirmation" minOccurs="0"
583 maxOccurs="unbounded"/>
584     </sequence>
585     <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
586   </choice>
587 </complexType>

```

588 **2.4.3.2 Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 589 **<SubjectConfirmationData>**

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

592 <ConfirmationMethod> [Required]

593 A URI reference that identifies a protocol to be used to authenticate the subject. URI references
594 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the
595 SAML profiles specification [SAMLProf]. Additional methods may be added by defining new URIs and
596 profiles or by private agreement.

597 <SubjectConfirmationData> [Optional]

598 Additional authentication information to be used by a specific authentication protocol. For example,
599 typical content of this element might be a <ds:KeyInfo> element as defined in the XML Signature
600 Syntax and Processing specification [XMLSig], which identifies a cryptographic key.

601 <ds:KeyInfo> [Optional]

602 An XML Signature [XMLSig] element that identifies a cryptographic key.

603 The following schema fragment defines the <SubjectConfirmation> element and its
604 SubjectConfirmationType complex type, along with the <SubjectConfirmationData> element and
605 the <ConfirmationMethod> element:

```

606 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
607 <complexType name="SubjectConfirmationType">
608   <sequence>
609     <element ref="saml:ConfirmationMethod"/>
610     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
611     <element ref="ds:KeyInfo" minOccurs="0"/>
612   </sequence>
613 </complexType>
614 <element name="SubjectConfirmationData" type="anyType"/>
615 <element name="ConfirmationMethod" type="anyURI"/>

```

616 **2.4.3.3 Element <Conditions>**

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

618 NotBefore [Optional]

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

621 NotOnOrAfter [Optional]

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

624 <Condition> [Any Number]

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

626 <AudienceRestrictionCondition> [Any Number]

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

628 <DoNotCacheCondition> [Any Number]

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

631 <ProxyRestrictionCondition> [Any Number]

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

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

```
636 <element name="Conditions" type="saml:ConditionsType"/>  
637 <complexType name="ConditionsType">  
638   <choice minOccurs="0" maxOccurs="unbounded">  
639     <element ref="saml:AudienceRestrictionCondition"/>  
640     <element ref="saml:DoNotCacheCondition">  
641     <element ref="saml:ProxyRestrictionCondition"/>  
642     <element ref="saml:Condition"/>  
643   </choice>  
644   <attribute name="NotBefore" type="dateTime" use="optional"/>  
645   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
646 </complexType>
```

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

- 651 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is
652 considered to be **Valid**.
- 653 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the
654 assertion is **Invalid**.
- 655 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity
656 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 657 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then
658 the assertion is considered to be **Valid**.

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

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

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

668 **2.4.3.3.1 Attributes NotBefore and NotOnOrAfter**

669 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within
670 the context of its profile of use. They do not guarantee that the statements in the assertion will be valid
671 throughout the validity period.

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

674 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
675 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the
676 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the
677 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),
678 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither
679 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is
680 valid at any time.

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

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

685 **2.4.3.3.2 Element <Condition>**

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

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

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

692 **2.4.3.3.3 Elements <AudienceRestrictionCondition> and <Audience>**

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

698 `<Audience>`

699 A URI reference that identifies an intended audience. The URI reference MAY identify a document
700 that describes the terms and conditions of audience membership. It MAY also contain the unique
701 identifier of a SAML system entity, as described by the <NameIdentifier> Format URI of
702 urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

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

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

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

```
713 <element name="AudienceRestrictionCondition"  
714 type="saml:AudienceRestrictionConditionType"/>  
715 <complexType name="AudienceRestrictionConditionType">  
716 <complexContent>  
717 <extension base="saml:ConditionAbstractType">  
718 <sequence>  
719 <element ref="saml:Audience" maxOccurs="unbounded"/>  
720 </sequence>  
721 </extension>  
722 </complexContent>  
723 </complexType>  
724 <element name="Audience" type="anyURI"/>
```

725 2.4.3.3.4 Element `<DoNotCacheCondition>`

726 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be
727 retained for future use. Note that no relying party is required to perform caching. However, any that do so
728 MUST observe this condition. This condition conveys one-time-use semantics, and is independent from
729 the `NotBefore` and `NotOnOrAfter` condition information.

730 A SAML authority SHOULD NOT include more than one `<DoNotCacheCondition>` element within a
731 `<Conditions>` element of an assertion. If multiple `<DoNotCacheCondition>` elements appear within
732 a `<Conditions>` element, a Relying Party MUST treat the multiple elements as though a single
733 `<DoNotCacheCondition>` element was specified.

734 For the purposes of determining the validity of the `<Conditions>` element, the
735 `<DoNotCacheCondition>` is considered to always be valid.

736 The following schema fragment defines the `<DoNotCacheCondition>` element and its
737 **DoNotCacheConditionType** complex type:

```
738 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType"/>  
739 <complexType name="DoNotCacheConditionType">  
740 <complexContent>  
741 <extension base="saml:ConditionAbstractType"/> </complexContent>  
742 </complexType>
```

743 2.4.3.3.5 Element `<ProxyRestrictionCondition>`

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

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

749 Count [Optional]

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

752 <Audience> [Zero or More]

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

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

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

763 A SAML authority SHOULD NOT include more than one <ProxyRestrictionCondition> element
764 within a <Conditions> element of an assertion. If multiple <ProxyRestrictionCondition>
765 elements appear within a <Conditions> element, a relying party MUST treat the multiple elements as
766 though a single <ProxyRestrictionCondition> element was specified, with a Count value equal to
767 the lowest of any specified, and the set of <Audience> elements consisting of the union of the elements
768 specified.

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

771 The following schema fragment defines the <ProxyRestrictionCondition> element and its
772 **ProxyRestrictionConditionType** complex type:

```
773 <element name="ProxyRestrictionCondition"  
774 type="saml:ProxyRestrictionConditionType"/>  
775 <complexType name="ProxyRestrictionConditionType">  
776   <complexContent>  
777     <extension base="saml:ConditionAbstractType">  
778       <sequence>  
779         <element ref="saml:Audience" minOccurs="0"  
780 maxOccurs="unbounded"/>  
781       </sequence>  
782       <attribute name="Count" type="nonNegativeInteger"  
783 use="optional"/>  
784     </extension>  
785   </complexContent>  
786 </complexType>
```

787 2.4.3.4 Element <Advice>

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

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

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

- 795 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating
796 the claims) or indirectly (by reference to the supporting assertions).
- 797 • State a proof of the assertion claims.

- 798 • Specify the timing and distribution points for updates to the assertion.

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

```
800 <element name="Advice" type="saml:AdviceType"/>
801 <complexType name="AdviceType">
802   <choice minOccurs="0" maxOccurs="unbounded">
803     <element ref="saml:AssertionIDReference"/>
804     <element ref="saml:AssertionURIReference"/>
805     <element ref="saml:Assertion"/>
806     <any namespace="##other" processContents="lax"/>
807   </choice>
808 </complexType>
```

809 2.5 Statements

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

811 2.5.1 Element <Statement>

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

815 **SessionIndex** [Optional]

816 Indexes a particular session between the subject and the authority issuing this statement. The value
817 of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST
818 NOT be a unique value identifying a principal's session at the authority.

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

```
821 <element name="Statement" type="saml:StatementAbstractType"/>
822 <complexType name="StatementAbstractType" abstract="true">
823   <attribute name="SessionIndex" type="string" use="optional"/>
824 </complexType>
```

825 ~~2.5.1.1 Elements <SubjectConfirmation>, <ConfirmationMethod>, and~~ 826 ~~<SubjectConfirmationData>~~

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

829 ~~<ConfirmationMethod> [Required]~~

830 ~~A URI reference that identifies a protocol to be used to authenticate the subject. URI references~~
831 ~~identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the~~
832 ~~SAML profiles specification [SAMLProf]. Additional methods may be added by defining new URIs and~~
833 ~~profiles or by private agreement.~~

834 ~~<SubjectConfirmationData> [Optional]~~

835 ~~Additional authentication information to be used by a specific authentication protocol.~~

836 ~~<ds:KeyInfo> [Optional]~~

837 ~~An XML Signature [XMLSig] element that identifies a cryptographic key.~~

838 The following schema fragment defines the ~~<SubjectConfirmation>~~ element and its
839 **SubjectConfirmationType** complex type, along with the ~~<SubjectConfirmationData>~~ element and
840 the ~~<ConfirmationMethod>~~ element:

```
841 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
842 <complexType name="SubjectConfirmationType">
843   <sequence>
844     <element ref="saml:ConfirmationMethod"/>
845     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
846     <element ref="ds:KeyInfo" minOccurs="0"/>
847   </sequence>
848 </complexType>
849 <element name="SubjectConfirmationData" type="anyType"/>
850 <element name="ConfirmationMethod" type="anyURI"/>
```

851 2.5.2 Element <AuthenticationStatement>

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

856 AuthenticationMethod [Required]

857 A URI reference that specifies the type of authentication that took place. URI references identifying
858 common authentication protocols are listed in Section Authentication Method Identifiers. A value of
859 urn:oasis:names:tc:SAML:2.0:am:authncontext indicates that an <AuthnContext>
860 element is included in the statement that describes further details of the authentication.

861 AuthenticationInstant [Required]

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

864 <SubjectLocality> [Optional]

865 Specifies the DNS domain name and IP address for the system from which the subject was
866 apparently authenticated.

867 <AuthnContext> [Optional]

868 The context used by the identity provider in the authentication event that yielded this statement.
869 Contains a [reference to an authentication context class](#), an authentication context statement or
870 [statement reference, or both a reference to one](#). ~~Optionally contains a reference to an authentication~~
871 ~~context class~~. See the Authentication Context specification [SAMLAuthnCxt] for a full description of
872 authentication context [class](#) information.

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

875 <AuthenticationStatement> elements MUST contain a SessionIndex value, conforming to the
876 rules specified in section 2.5.1.

877 [Assertions containing <AuthenticationStatement> elements MUST contain a <Subject> element.](#)

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

```
880 <element name="AuthenticationStatement"
881         type="saml:AuthenticationStatementType"/>
882 <complexType name="AuthenticationStatementType">
```

```

883     <complexContent>
884         <extension base="saml:StatementAbstractType">
885             <sequence>
886                 <element ref="saml:SubjectLocality" minOccurs="0"/>
887                 <element ref="saml:AuthnContext" minOccurs="0"/>
888             </sequence>
889             <attribute name="AuthenticationMethod" type="anyURI"
890 use="required"/>
891             <attribute name="AuthenticationInstant" type="dateTime"
892 use="required"/>
893         </extension>
894     </complexContent>
895 </complexType>

```

896 2.5.2.1 Element <SubjectLocality>

897 The <SubjectLocality> element specifies the DNS domain name and IP address for the system
898 from which the subject was authenticated. It has the following attributes:

899 IPAddress [Optional]

900 The IP address of the system from which the subject was authenticated.

901 DNSAddress [Optional]

902 The DNS address of the system from which the subject was authenticated.

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

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

```

907 <element name="SubjectLocality"
908         type="saml:SubjectLocalityType"/>
909 <complexType name="SubjectLocalityType">
910     <attribute name="IPAddress" type="string" use="optional"/>
911     <attribute name="DNSAddress" type="string" use="optional"/>
912 </complexType>

```

913 2.5.2.2 Element <AuthnContext>

914 The <AuthnContext> element specifies the context of an authentication event with an **optional** context
915 class **referenceURI**, followed by an **authentication** context statement or statement reference, **or both**. It's
916 complex **AuthnContextType** has the following elements:

917 <AuthnContextClassRef> [Optional]

918 A URI identifying an authentication context class that describes the authentication context statement
919 that follows.

920 <AuthnContextStatement> or <AuthnContextStatementRef> [**RequiredOptional**]

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

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

```

925 <element name="AuthnContext" type="saml:AuthnContextType"/>
926 <complexType name="AuthnContextType">
927     <sequence>
928         <element ref="saml:AuthnContextClassRef" minOccurs="0"/>

```



```

929 |         <choice minOccurs="0">
930 |             <element ref="saml:AuthnContextStatement"/>
931 |             <element ref="saml:AuthnContextStatementRef"/>
932 |         </choice>
933 |     </sequence>
934 | </complexType>
935 | <element name="AuthnContextClassRef" type="anyURI"/>
936 | <element name="AuthnContextStatementRef" type="anyURI"/>
937 | <element name="AuthnContextStatement" type="anyType"/>

```

938 **2.5.3 Element <AttributeStatement>**

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

942 <Attribute> [One or More]

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

944 Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

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

```

947 | <element name="AttributeStatement" type="saml:AttributeStatementType"/>
948 | <complexType name="AttributeStatementType">
949 |     <complexContent>
950 |         <extension base="saml:StatementAbstractType">
951 |             <sequence>
952 |                 <element ref="saml:Attribute"
953 | maxOccurs="unbounded"/>
954 |             </sequence>
955 |         </extension>
956 |     </complexContent>
957 | </complexType>

```

958 **2.5.3.1 Elements <AttributeDesignator> and <Attribute>**

959 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It
960 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute
961 values within a specific namespace be returned (see Section Element <AttributeQuery> for more
962 information). The <AttributeDesignator> element contains the following XML attributes:

963 Name [Required]

964 The name of the attribute.

965 NameFormat [Optional]

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

971 ValueType [Optional]

972 A URI reference representing the datatype of the desired or supplied attribute. If no ValueType
973 value is provided, the identifier urn:oasis:names:tc:saml:2.0:valuetype-format:unspecified (see
974 Section 7.x) is in effect. Note that datatypes specified on the <AttributeValue> element

975 using `xsi:type` have no SAML-defined relationship with `ValueType`. The `ValueType` setting
976 (default or explicit) in an attribute query using the `<AttributeDesignator>` element MUST be
977 exactly matched (in addition to other exact matches as described in Section x) in order for an
978 attribute to be returned.

979 Arbitrary attributes

980 This complex type uses an `<xsd:anyAttribute>` extension point to allow for arbitrary XML
981 attributes to be added to `<AttributeDesignator>` constructs without the need for an explicit
982 schema extension. This allows additional fields to be added as needed to supply additional
983 parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-
984 namespace-qualified) XML attributes to the `AttributeType` complex type or to any element bound
985 to this type or a derivation of it; such attributes are reserved for future maintenance and
986 enhancement of SAML itself.

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

```
989 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
990 <complexType name="AttributeDesignatorType">  
991   <attribute name="Name" type="string" use="required"/>  
992   <attribute name="NameFormat" type="anyURI" use="optional"/>  
993   <attribute name="ValueType" type="anyURI" use="optional"/>  
994   <anyAttribute/>  
995 </complexType>
```

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

999 `<AttributeValue>` [Any Number]

1000 The value of the attribute. If an attribute contains more than one discrete value, it is
1001 RECOMMENDED that each value appear in its own `<AttributeValue>` element. If the attribute
1002 exists but has no value, then the `<AttributeValue>` element MUST be omitted. If more than one
1003 `<AttributeValue>` element is supplied for an attribute, and any of the elements have a datatype
1004 assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have the identical
1005 datatype assigned.

1006 Arbitrary attributes

1007 This complex type inherits from **AttributeDesignatorType** the ability to add arbitrary XML
1008 attributes to `<Attribute>` constructs without the need for an explicit schema extension. This
1009 allows additional fields to be added as needed to supply the context in which the attribute should
1010 be understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML
1011 attributes to the `AttributeType` complex type or to any element bound to this type or a derivation
1012 of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

1013 The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
1014 <element name="Attribute" type="saml:AttributeType"/>  
1015 <complexType name="AttributeType">  
1016   <complexContent>  
1017     <extension base="saml:AttributeDesignatorType">  
1018       <sequence>  
1019         <element ref="saml:AttributeValue" minOccurs="0"  
1020         maxOccurs="unbounded"/>  
1021       </sequence>  
1022     </extension>  
1023   </complexContent>  
1024 </complexType>
```

1025 **2.5.3.1.1 Element <AttributeValue>**

1026 The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType** type,
1027 which allows any well-formed XML to appear as the content of the element.

1028 If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer**
1029 or **xsd:string**), the data type MAY be declared explicitly by means of an `xsi:type` declaration in the
1030 <AttributeValue> element. If the attribute value contains structured data, the necessary data
1031 elements MAY be defined in an extension schema.

1032 **Note:** Specifying a datatype on <AttributeValue> using `xsi:type` will require the
1033 presence of the extension schema that defines the datatype in order for schema
1034 processing to proceed.

1035 The following schema fragment defines the <AttributeValue> element:

```
1036 <element name="AttributeValue" type="anyType"/>
```

1037 **2.5.4 Element <AuthorizationDecisionStatement>**

1038 **Note:** The <AuthorizationDecisionStatement> feature has been frozen as of
1039 SAML V2.0, with no future enhancements planned. Users who require additional
1040 functionality may want to consider the eXtensible Access Control Markup Language
1041 [XACML], which offers enhanced authorization decision features.

1042 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority
1043 asserting that a request for access by the statement's subject to the specified resource has resulted in
1044 the specified authorization decision on the basis of some optionally specified evidence.

1045 The resource is identified by means of a URI reference. In order for the assertion to be interpreted
1046 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference
1047 in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different
1048 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing
1049 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1050 In general, the rules for equivalence and definition of a normal form, if any, are scheme
1051 dependent. When a scheme uses elements of the common syntax, it will also use the common
1052 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL
1053 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where
1054 the port is elided.

1055 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the
1056 URI normalized form wherever possible as follows:

- 1057 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1058 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1059 Inconsistent URI reference interpretation can also result from differences between the URI reference
1060 syntax and the semantics of an underlying file system. Particular care is required if URI references are
1061 employed to specify an access control policy language. The following security conditions should be
1062 satisfied by the system which employs SAML assertions:

- 1063 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,
1064 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a
1065 part of the resource URI reference.

- 1066 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users
1067 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to
1068 gain access to a denied resource by creating such an equivalence.

1069 The `<AuthorizationDecisionStatement>` element is of type
1070 **AuthorizationDecisionStatementType**, which extends **StatementAbstractType** with the addition of the
1071 following elements (in order) and attributes:

1072 Resource [Required]

1073 A URI reference identifying the resource to which access authorization is sought. It is permitted for
1074 this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the
1075 start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1076 Decision [Required]

1077 The decision rendered by the SAML authority with respect to the specified resource. The value is of
1078 the **DecisionType** simple type.

1079 `<Action>` [One or more]

1080 The set of actions authorized to be performed on the specified resource.

1081 `<Evidence>` [Optional]

1082 A set of assertions that the SAML authority relied on in making the decision.

1083 Assertions containing `<AuthorizationDecisionStatement>` elements MUST contain a `<Subject>`
1084 element.

1085 The following schema fragment defines the `<AuthorizationDecisionStatement>` element and its
1086 **AuthorizationDecisionStatementType** complex type:

```
1087 <element name="AuthorizationDecisionStatement"  
1088 type="saml:AuthorizationDecisionStatementType"/>  
1089 <complexType name="AuthorizationDecisionStatementType">  
1090   <complexContent>  
1091     <extension base="saml:StatementAbstractType">  
1092       <sequence>  
1093         <element ref="saml:Action" maxOccurs="unbounded"/>  
1094         <element ref="saml:Evidence" minOccurs="0"/>  
1095       </sequence>  
1096       <attribute name="Resource" type="anyURI" use="required"/>  
1097       <attribute name="Decision" type="saml:DecisionType"  
1098 use="required"/>  
1099     </extension>  
1100   </complexContent>  
1101 </complexType>
```

1102 2.5.4.1 Element `<Action>`

1103 The `<Action>` element specifies an action on the specified resource for which permission is sought. It
1104 has the following attribute and string-data content:

1105 Namespace [Optional]

1106 A URI reference representing the namespace in which the name of the specified action is to be
1107 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwdc-
1108 negotiation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

1109 *string data* [Required]

1110 An action sought to be performed on the specified resource.

1111 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
1112 <element name="Action" type="saml:ActionType"/>
1113 <complexType name="ActionType">
1114   <simpleContent>
1115     <extension base="string">
1116       <attribute name="Namespace" type="anyURI"/>
1117     </extension>
1118   </simpleContent>
1119 </complexType>
```

1120 2.5.4.2 Element <Evidence>

1121 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on
1122 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one
1123 or more of the following elements:

1124 <AssertionIDReference> [Any number]

1125 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1126 <AssertionURIReference> [Any number]

1127 Specifies an assertion by reference to a URI.

1128 <Assertion> [Any number]

1129 Specifies an assertion by value.

1130 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party
1131 and the SAML authority making the authorization decision. For example, in the case that the SAML
1132 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use
1133 that assertion as evidence in making its authorization decision without endorsing the <Evidence>
1134 element's assertion as valid either to the relying party or any other third party.

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

```
1136 <element name="Evidence" type="saml:EvidenceType"/>
1137 <complexType name="EvidenceType">
1138   <choice maxOccurs="unbounded">
1139     <element ref="saml:AssertionIDReference"/>
1140     <element ref="saml:AssertionURIReference"/>
1141     <element ref="saml:Assertion"/>
1142   </choice>
1143 </complexType>
```

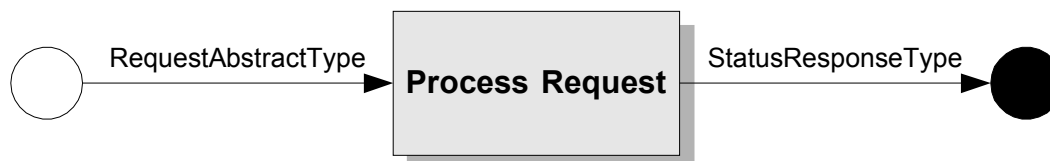
3 SAML Protocols

1144

1145 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of
1146 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting
1147 queries, assertions, and other messages using existing widely deployed transport protocols.

1148 Specific SAML request and response messages derive from common types. The requester sends an
1149 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an
1150 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1151



1153

Figure 1: SAML Request-Response Protocol

1154 The protocols defined by SAML achieve the following actions:

- 1155 • Returning one or more requested assertions (includes a direct request of the desired assertions, as
1156 well as querying for assertions that meet particular criteria)
- 1157 • Performing authentication on request and returning the corresponding assertion
- 1158 • Registering a **federated**-name **identifier** or terminating a **federated**-name registration on request
- 1159 • Retrieve a protocol message that has been requested by means of an artifact
- 1160 • Performing a near-simultaneous logout of a collection of related sessions ("single logout") on
1161 request
- 1162 • Providing a name identifier mapping on request

3.1 Schema Header and Namespace Declarations

1163

1164 The following schema fragment defines the XML namespaces and other header information for the
1165 protocol schema:

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
    schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
  <annotation>
    <documentation>
      Document identifier: sstc-saml-schema-protocol-2.0
```

```
1184         Location: http://www.oasis-
1185 open.org/committees/documents.php?wg_abbrev=security
1186         </documentation>
1187     </annotation>
1188     ...
1189 </schema>
```

1190 3.2 Requests and Responses

1191 The following sections define the SAML constructs that underlie request and response messages.

1192 3.2.1 Complex Type RequestAbstractType

1193 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.
1194 This type defines common attributes and elements that are associated with all SAML requests:

1195 **RequestID** [Required]

1196 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in
1197 Section 1.2.3 for identifier uniqueness. The values of the **RequestID** attribute in a request and the
1198 **InResponseTo** attribute in the corresponding response MUST match.

1199 **MajorVersion** [Required]

1200 The major version of this request. The identifier for the version of SAML defined in this specification
1201 is 2. SAML versioning is discussed in Section SAML Versioning.

1202 **MinorVersion** [Required]

1203 The minor version of this request. The identifier for the version of SAML defined in this specification
1204 is 0. SAML versioning is discussed in Section SAML Versioning.

1205 **IssueInstant** [Required]

1206 The time instant of issue of the request. The time value is encoded in UTC as described in Section
1207 Time Values.

1208 **Consent** [Optional]

1209 Indicates whether or not consent has been obtained from a user in the sending this request.

1210 **<Issuer>** [Optional]

1211 Identifies the entity that generated the request message.

1212 **<ds:Signature>** [Optional]

1213 An XML Signature that authenticates the request, as described in Section SAML and XML Signature
1214 Syntax and Processing.

1215 **<RelayState>** [Optional]

1216 This contains state information that MUST be relayed back in the associated response.

1217 **<Extensions>** [Optional]

1218 This contains optional protocol message extension elements that are agreed upon between the
1219 communicating parties.

1220 **Note:** The **<RespondWith>** element has been removed from **<Request>** for V2.0 of
1221 SAML.

1222 The following schema fragment defines the **RequestAbstractType** complex type:

```

1223 <complexType name="RequestAbstractType" abstract="true">
1224   <sequence>
1225     <element ref="saml:Issuer" minOccurs="0"/>
1226     <element ref="ds:Signature" minOccurs="0"/>
1227     <element ref="samlp:RelayState" minOccurs="0"/>
1228     <element ref="samlp:Extensions" minOccurs="0"/>
1229   </sequence>
1230   <attribute name="RequestID" type="ID" use="required"/>
1231   <attribute name="MajorVersion" type="integer" use="required"/>
1232   <attribute name="MinorVersion" type="integer" use="required"/>
1233   <attribute name="IssueInstant" type="dateTime" use="required"/>
1234   <attribute name="Consent" type="anyURI" use="optional"/>
1235 </complexType>
1236 <element name="Extensions" type="samlp:ExtensionsType"/>
1237 <complexType name="ExtensionsType">
1238   <sequence>
1239     <any namespace="##other" processContents="lax"
1240     maxOccurs="unbounded"/>
1241   </sequence>
1242 </complexType>

```

1243 3.2.1.1 Element ~~<RelayState>~~

1244 ~~SAML requests MAY contain a string-valued element containing state information that the requester~~
1245 ~~wishes the responder to include in the response. This is particularly useful with asynchronous bindings~~
1246 ~~of protocol messages, such as the encoding of messages in browser URLs. This data SHOULD be~~
1247 ~~integrity-protected by the requester and MAY have other protections placed on it by the requester, such~~
1248 ~~as confidentiality. The length of this value SHOULD be kept as short as possible because of limitations~~
1249 ~~of the bindings in which it may be needed.~~

1250 ~~The following schema fragment defines the <RelayState> element:~~

```

1251 <element name="RelayState" type="string"/>

```

1252 3.2.2 Complex Type StatusResponseType

1253 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This
1254 type defines common attributes and elements that are associated with all SAML responses:

1255 ResponseID [Required]

1256 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in
1257 Section 1.2.3 for identifier uniqueness.

1258 InResponseTo [Optional]

1259 A reference to the identifier of the request to which the response corresponds, if any. If the response
1260 is not generated in response to a request, or if the `RequestID` attribute value of a request cannot be
1261 determined (because the request is malformed), then this attribute MUST NOT be present.
1262 Otherwise, it MUST be present and its value MUST match the value of the corresponding
1263 `RequestID` attribute value.

1264 MajorVersion [Required]

1265 The major version of this response. The identifier for the version of SAML defined in this
1266 specification is 2. SAML versioning is discussed in Section SAML Versioning.

1267 MinorVersion [Required]

1268 The minor version of this response. The identifier for the version of SAML defined in this
1269 specification is 0. SAML versioning is discussed in Section SAML Versioning.

- 1270 IssueInstant [Required]
 1271 The time instant of issue of the response. The time value is encoded in UTC as described in Section
 1272 Time Values.
- 1273 Recipient [Optional]
 1274 The intended recipient of this response. This is useful to prevent malicious forwarding of responses
 1275 to unintended recipients, a protection that is required by some use profiles. It is set by the generator
 1276 of the response to a URI reference that identifies the intended recipient. If present, the actual
 1277 recipient MUST check that the URI reference identifies the recipient or a resource managed by the
 1278 recipient. If it does not, the response MUST be discarded.
- 1279 <Issuer> [Optional]
 1280 Identifies the entity that generated the response message.
- 1281 <ds:Signature> [Optional]
 1282 An XML Signature that authenticates the response, as described in Section SAML and XML
 1283 Signature Syntax and Processing.
- 1284 ~~<RelayState> [Optional]~~
 1285 ~~This contains state information from the associated request being relayed back in the response. It~~
 1286 ~~MUST match the <RelayState> value in the associated request, if any.~~
- 1287 <Extensions> [Optional]
 1288 This contains optional protocol message extension elements that are agreed upon between the
 1289 communicating parties.
- 1290 <Status> [Required]
 1291 A code representing the status of the corresponding request.
- 1292 The following schema fragment defines the **StatusResponseType** complex type:

```

1293 <complexType name="StatusResponseType">
1294   <sequence>
1295     <element ref="saml:Issuer" minOccurs="0"/>
1296     <element ref="ds:Signature" minOccurs="0"/>
1297     <element ref="samlp:RelayState" minOccurs="0"/>
1298     <element ref="samlp:Extensions" minOccurs="0"/>
1299     <element ref="samlp:Status"/>
1300   </sequence>
1301   <attribute name="ResponseID" type="ID" use="required"/>
1302   <attribute name="InResponseTo" type="NCName" use="optional"/>
1303   <attribute name="MajorVersion" type="integer" use="required"/>
1304   <attribute name="MinorVersion" type="integer" use="required"/>
1305   <attribute name="IssueInstant" type="dateTime" use="required"/>
1306   <attribute name="Recipient" type="anyURI" use="optional"/>
1307 </complexType>

```

1308 3.2.2.1 Element <Status>

1309 The <Status> element contains the following elements:

- 1310 <StatusCode> [Required]
 1311 A code representing the status of the corresponding request.
- 1312 <StatusMessage> [Optional]
 1313 A message which MAY be returned to an operator.

1314 <StatusDetail> [Optional]

1315 Additional information concerning an error condition.

1316 The following schema fragment defines the <Status> element and its **StatusType** complex type:

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

1325 3.2.2.2 Element <StatusCode>

1326 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the
1327 corresponding request. The <StatusCode> element has the following element and attribute:

1328 Value [Required]

1329 The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST
1330 be provided. The value of the topmost <StatusCode> element MUST be from the top-level list
1331 provided in this section.

1332 <StatusCode> [Optional]

1333 A subordinate status code that provides more specific information on an error condition.

1334 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The
1335 local parts of these QNames are as follows:

1336 Success

1337 The request succeeded.

1338 Requester

1339 The request could not be performed due to an error on the part of the requester.

1340 Responder

1341 The request could not be performed due to an error on the part of the SAML responder or SAML
1342 authority.

1343 VersionMismatch

1344 The SAML responder could not process the request because the version of the request message
1345 was incorrect.

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

1348 | ~~FederationDoesNotExist~~

1349 | ~~The responding provider does not recognize the federated <NameIdentifier> in the request.~~

1350 | InvalidNameIDPolicy

1351 The responding provider does not support the specified name identifier format for the requested
1352 subject.

1353 NoAuthnContext

1354 The specified authentication context requirements cannot be met by the responder.

1355 NoAvailableIDP
1356 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in
1357 an <IDPList> can be resolved or that none of the supported identity providers are available.

1358 NoPassive
1359 Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1360 NoSupportedIDP
1361 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are
1362 supported by the intermediary.

1363 ProxyCountExceeded
1364 Indicates that an identity provider cannot authenticate the principal directly and is not permitted to
1365 proxy the request further.

1366 RequestDenied
1367 The SAML responder or SAML authority is able to process the request but has chosen not to
1368 respond. This status code MAY be used when there is concern about the security context of the
1369 request message or the sequence of request messages received from a particular requester.

1370 RequestUnsupported
1371 The SAML responder or SAML authority does not support the request.

1372 RequestVersionDeprecated
1373 The SAML responder can not process any requests with the protocol version specified in the
1374 request.

1375 RequestVersionTooHigh
1376 The SAML responder cannot process the request because the protocol version specified in the
1377 request message is a major upgrade from the highest protocol version supported by the responder.

1378 RequestVersionTooLow
1379 The SAML responder cannot process the request because the protocol version specified in the
1380 request message is too low.

1381 ResourceNotRecognized
1382 The SAML authority does not wish to support resource-specific attribute queries, or the resource
1383 value provided in the request message is invalid or unrecognized.

1384 TooManyResponses
1385 The response message would contain more elements than the SAML responder will return.

1386 UnknownPrincipal
1387 The responding provider does not recognize the principal specified or implied by the request.

1388 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT
1389 define additional codes in the SAML assertion or protocol namespace.

1390 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value
1391 attribute and have the above semantics only in that context.

1392 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex
1393 type:

1394 <element name="StatusCode" type="samlp:StatusCodeType"/>

```

1395 <complexType name="StatusCodeType">
1396   <sequence>
1397     <element ref="samlp:StatusCode" minOccurs="0"/>
1398   </sequence>
1399   <attribute name="Value" type="QName" use="required"/>
1400 </complexType>

```

1401 3.2.2.3 Element <StatusMessage>

1402 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1403 The following schema fragment defines the <StatusMessage> element:

```

1404 <element name="StatusMessage" type="string"/>

```

1405 3.2.2.4 Element <StatusDetail>

1406 The <StatusDetail> element MAY be used to specify additional information concerning an error condition.

1408 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType** complex type:

```

1410 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1411 <complexType name="StatusDetailType">
1412   <sequence>
1413     <any namespace="##any" processContents="lax" minOccurs="0"
1414     maxOccurs="unbounded"/>
1415   </sequence>
1416 </complexType>

```

1417 3.3 Assertion Query and Request Protocol

1418 This section defines messages and processing rules for requesting existing assertions by reference or querying for assertions by subject and statement type.

1420 3.3.1 Element <AssertionIDRequest>

1421 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest> message can be used to request that the assertion(s) be returned in a <Response> message. The <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section Element <AssertionIDReference> for more information on this element.

1425 The following schema fragment defines the <AssertionIDRequest> element:

```

1426 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
1427 <complexType name="AssertionIDRequestType">
1428   <complexContent>
1429     <extension base="samlp:RequestAbstractType">
1430       <sequence>
1431         <element ref="saml:AssertionIDReference"
1432         maxOccurs="unbounded"/>
1433       </sequence>
1434     </extension>
1435   </complexContent>
1436 </complexType>

```

1437 3.3.2 Queries

1438 The following sections define the SAML query request messages.

1439 3.3.2.1 Element <SubjectQuery>

1440 The <SubjectQuery> message element is an extension point that allows new SAML queries to be
1441 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and
1442 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>
1443 element and an optional *SessionIndex* attribute to **RequestAbstractType**.

1444 *SessionIndex* [Optional]

1445 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1446 containing subject statements do you have for this subject within the context of the supplied session
1447 information?”

1448 If the *SessionIndex* attribute is present in any defined query, at least one element that extends
1449 **StatementAbstractType** in the set of returned assertions **MUST** contain an *SessionIndex* attribute
1450 that matches the *SessionIndex* attribute in the query. It is **OPTIONAL** for the complete set of all such
1451 matching assertions to be returned in the response.

1452 The following schema fragment defines the <SubjectQuery> element and its
1453 **SubjectQueryAbstractType** complex type:

```
1454 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>  
1455 <complexType name="SubjectQueryAbstractType" abstract="true">  
1456   <complexContent>  
1457     <extension base="saml:RequestAbstractType">  
1458       <sequence>  
1459         <element ref="saml:Subject"/>  
1460       </sequence>  
1461       <attribute name="SessionIndex" type="string"  
1462         use="optional"/>  
1463     </extension>  
1464   </complexContent>  
1465 </complexType>
```

1466 3.3.2.2 Element <AuthenticationQuery>

1467 The <AuthenticationQuery> message element is used to make the query “What assertions
1468 containing authentication statements are available for this subject?” A successful <Response> will
1469 contain one or more assertions containing authentication statements.

1470 The <AuthenticationQuery> message **MUST NOT** be used as a request for a new authentication
1471 using credentials provided in the request. <AuthenticationQuery> is a request for statements about
1472 authentication acts that have occurred in a previous interaction between the indicated subject and the
1473 Authentication Authority.

1474 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the
1475 addition of the following attribute:

1476 *AuthenticationMethod* [Optional]

1477 If present, specifies a filter for possible responses. Such a query asks the question “What assertions
1478 containing authentication statements do you have for this subject with the supplied authentication
1479 method?”

1480 In response to an authentication query, a SAML authority returns assertions with authentication
1481 statements as follows:

- 1482 • Rules given in Section for matching against the <Subject> element of the query identify the
1483 assertions that may be returned.
 - 1484 • If the AuthenticationMethod attribute is present in the query, at least one
1485 <AuthenticationStatement> element in the set of returned assertions MUST contain an
1486 AuthenticationMethod attribute that matches the AuthenticationMethod attribute in
1487 the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in
1488 the response.
- 1489 TODO: add <AuthnContext> into message in some fashion. Maybe reuse <RequestAuthnContext> from
1490 <AuthnRequest>?

1491 The following schema fragment defines the <AuthenticationQuery> element and its
1492 **AuthenticationQueryType** complex type:

```
1493 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1494 <complexType name="AuthenticationQueryType">
1495   <complexContent>
1496     <extension base="samlp:SubjectQueryAbstractType">
1497       <attribute name="AuthenticationMethod" type="anyURI"/>
1498     </extension>
1499   </complexContent>
1500 </complexType>
```

1501 3.3.2.3 Element <AttributeQuery>

1502 The <AttributeQuery> element is used to make the query "Return the requested attributes for this
1503 subject." A successful response will be in the form of assertions containing attribute statements. This
1504 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of
1505 the following element and attribute:

1506 Resource [Optional]

1507 If present, specifies that the attribute query is being made in order to evaluate a specific access
1508 request relating to the resource. The SAML authority MAY use the resource attribute to establish the
1509 scope of the request. It is permitted for this attribute to have the value of the empty URI reference
1510 (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]
1511 §4.2.

1512 If the resource attribute is specified and the SAML authority does not wish to support resource-
1513 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the
1514 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a
1515 second-level <StatusCode> value of ResourceNotRecognized.

1516 <AttributeDesignator> [Any Number]

1517 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no
1518 attributes are specified, it indicates that all attributes allowed by policy are requested.

1519 In response to an attribute query, a SAML authority returns assertions with attribute statements as
1520 follows:

- 1521 • Rules given in Section for matching against the <Subject> element of the query identify the
1522 assertions that may be returned.
- 1523 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute
1524 values returned, as noted above.
- 1525 • The SAML authority MAY take the Resource attribute into account in further constraining the values
1526 returned, as noted above.

- 1527 • The attribute values returned MAY be constrained by application-specific policy considerations.

1528 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**
1529 complex type:

```
1530 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1531 <complexType name="AttributeQueryType">
1532   <complexContent>
1533     <extension base="samlp:SubjectQueryAbstractType">
1534       <sequence>
1535         <element ref="saml:AttributeDesignator"
1536           minOccurs="0" maxOccurs="unbounded"/>
1537       </sequence>
1538       <attribute name="Resource" type="anyURI" use="optional"/>
1539     </extension>
1540   </complexContent>
1541 </complexType>
```

1542 3.3.2.4 Element <AuthorizationDecisionQuery>

1543 The <AuthorizationDecisionQuery> element is used to make the query “Should these actions on
1544 this resource be allowed for this subject, given this evidence?” A successful response will be in the form
1545 of assertions containing authorization decision statements.

1546 **Note:** The <AuthorizationDecisionQuery> feature has been frozen as of SAML
1547 V2.0, with no future enhancements planned. Users who require additional functionality
1548 may want to consider the eXtensible Access Control Markup Language [XACML], which
1549 offers enhanced authorization decision features.

1550 This element is of type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType**
1551 with the addition of the following elements and attribute:

1552 Resource [Required]

1553 A URI reference indicating the resource for which authorization is requested.

1554 <Action> [One or More]

1555 The actions for which authorization is requested.

1556 <Evidence> [Optional]

1557 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1558 In response to an authorization decision query, a SAML authority returns assertions with authorization
1559 decision statements as follows:

- 1560 • Rules given in Section 3.3.4.1 for matching against the <Subject> element of the query identify the
1561 assertions that may be returned.

1562 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
1563 **AuthorizationDecisionQueryType** complex type:

```
1564 <element name="AuthorizationDecisionQuery"
1565   type="samlp:AuthorizationDecisionQueryType"/>
1566 <complexType name="AuthorizationDecisionQueryType">
1567   <complexContent>
1568     <extension base="samlp:SubjectQueryAbstractType">
1569       <sequence>
1570         <element ref="saml:Action" maxOccurs="unbounded"/>
1571         <element ref="saml:Evidence" minOccurs="0"/>
1572       </sequence>
1573       <attribute name="Resource" type="anyURI" use="required"/>
1574     </extension>
1575   </complexContent>
1576 </complexType>
```

1574
1575
1576

```
        </extension>  
    </complexContent>  
</complexType>
```

1577 3.3.3 Element <Response>

1578 The <Response> message element is used when a response consists of a list of zero or more
1579 assertions that answer the request. It has the complex type **ResponseType**, which extends
1580 **StatusResponseType** by adding the following element:

1581 <Assertion> [Any Number]

1582 Specifies an assertion by value. (See Section Element <Assertion> for more information.)

1583 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1584 <element name="Response" type="samlp:ResponseType"/>  
1585 <complexType name="ResponseType">  
1586   <complexContent>  
1587     <extension base="samlp:StatusResponseType">  
1588       <sequence>  
1589         <element ref="saml:Assertion" minOccurs="0"  
1590 maxOccurs="unbounded"/>  
1591       </sequence>  
1592     </extension>  
1593   </complexContent>  
1594 </complexType>
```

1595 3.3.3.1 Processing Rules

1596 In response to a query message, every assertion returned by a SAML authority **MUST** contain a
1597 <Subject> element that **strongly matches** the <Subject> element found in the query.

1598 A <Subject> element S1 strongly matches S2 if and only if the following two conditions both apply:

- 1599 • If S2 includes an identifier element (any element whose type is derived from
1600 **BaseIdentifierAbstractType**), then S1 must include an identical identifier element.
- 1601 • If S2 includes one or more <SubjectConfirmation> elements, then S1 must include at least one
1602 <SubjectConfirmation> element such that the assertion's subject can be confirmed in the
1603 manner described by at least one element in the requested set.

1604 If the SAML authority cannot provide an assertion with any statements satisfying the constraints
1605 expressed by a query, the <Response> element **MUST NOT** contain an <Assertion> element and
1606 **MUST** include a <StatusCode> element with value *Success*. It **MAY** return a <StatusMessage>
1607 element with additional information.

1608 3.4 Authentication Request Protocol

1609 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing
1610 authentication statements to establish a security context at one or more relying parties, it can use the
1611 authentication request protocol to send an <AuthnRequest> message to a SAML authority and request
1612 that it return a <Response> message containing one or more such assertions. ~~A SAML authority that
1613 supports this protocol is also termed an identity provider.~~ Such assertions **MAY** contain additional
1614 statements of any type, but at least one assertion **MUST** contain at least one authentication statement.
1615 A SAML authority that supports this protocol is also termed an identity provider.

1616 Apart from this requirement, the specific contents of the returned assertions depend on the profile or
1617 context of use. Also, the exact means by which the principal or agent authenticates to the identity
1618 provider are not specified, though the means of authentication MAY impact the content of the response.
1619 Other issues related to the validation of authentication credentials by the identity provider or any
1620 communication between the identity provider and any other entities involved in the authentication
1621 process are also out of scope of this protocol.

1622 The descriptions and processing rules in the following sections reference the following actors, many of
1623 whom might be the same entity in a particular profile of use:

1624 Request Issuer

1625 The entity who creates the authentication request and to whom the response is to be returned.

1626 Presenter

1627 The entity who presents the request to the authority and either authenticates itself during the
1628 sending of the message, or relies on an existing security context to establish its identity. If not
1629 the request issuer, the sender acts as an intermediary between the request issuer and the
1630 responding identity provider.

1631 Requested Subject

1632 The entity about whom one or more assertions are being requested.

1633 Confirming Subject

1634 The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>
1635 elements of the resulting assertion(s).

1636 Relying Party

1637 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by
1638 the profile or context of use, generally to establish a security context.

1639 **3.4.1 Element <AuthnRequest>**

1640 To request that an identity provider issue an authentication assertion, an entity authenticates to it (or
1641 relies on an existing security context) and sends it an <AuthnRequest> message that describes the
1642 properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may
1643 be information that relates to the content of the assertion and/or information that relates to how the
1644 resulting <Response> message should be delivered to the request issuer.

1645 The request issuer might not be the same as the presenter of the request, if for example the request
1646 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the
1647 requested subject to provide a service.

1648 The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected
1649 by the protocol binding used to deliver the message.

1650 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and
1651 adds the following elements and attributes, all of which are optional in general, but may be required by
1652 specific profiles:

1653 <Subject> [Optional]

1654 Specifies the requested subject of the resulting assertion(s). This may include one or more
1655 <SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions' can
1656 be confirmed.

1657 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the

1658 requested subject. If no <SubjectConfirmation> elements are included, then the presenter is
1659 presumed to be the only confirming entity required and the method is implied by the profile of use
1660 and/or the policies of the identity provider.

1661 <NameIDPolicy> [Optional]
1662 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,
1663 then any type of identifier supported by the identity provider for the requested subject can be used,
1664 constrained by any relevant deployment-specific policies, with respect to privacy, for example.
1665

1666 <Conditions> [Optional]
1667 Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the
1668 resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary.

1669 <RequestAuthnContext> [Optional]
1670 Specifies the requirements, if any, that the request issuer places on the authentication context that
1671 applies to the responding provider's authentication of the presenter.

1672 <Scoping> [Optional]
1673 Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as
1674 limitations and context related to proxying of the <AuthnRequest> message to subsequent identity
1675 providers by the responder.

1676 IsPassive [Optional]
1677 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of
1678 the user interface from the request issuer and interact with the presenter in a noticeable fashion. If a
1679 value is not provided, the default is "true".

1680 ForceAuthn [Optional]
1681 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than
1682 rely on a previous security context. If a value is not provided, the default is "false". However, if both
1683 ForceAuthn and IsPassive are "true", the identity provider MUST NOT freshly authenticate the
1684 presenter unless the constraints of IsPassive can be met.

1685 ProtocolBinding [Optional]
1686 A URI that identifies a SAML protocol binding to be used when returning the <Response> message.

1687 AssertionConsumerServiceID [Optional]
1688 References one of a set of <AssertionConsumerService> elements in the request issuer's
1689 metadata as the one to which the <Response> should be returned. It applies only to profiles that
1690 specify use of this metadata element, in which the request issuer is different than the presenter. If
1691 omitted, the metadata element labeled with the isDefault attribute MUST be used with such
1692 profiles.

1693 AssertionConsumerServiceURL [Optional]
1694 Specifies by value the location to which the presenter of an <AuthnRequest>
1695 recognizes that the issuer's request cannot be satisfied for some reason, this attribute specifies
1696 where a <Response> message generated by that would-be presenter MUST be returned. This
1697 attribute can be required by certain profiles. The responder MUST insure by some means (such as
1698 metadata) that the value specified is in fact associated with the request issuer.

1699 **ProviderName [Optional]**
1700 Specifies the human-readable name of the request issuer for use by the presenter's user agent or
1701 the identity provider.

1702 See Section 3.4.1.8 for general processing rules regarding this message.

1703 The following schema fragment defines the `<AuthnRequest>` element and its **AuthnRequestType**
1704 complex type:

```
1705 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
1706 <complexType name="AuthnRequestType">
1707   <complexContent>
1708     <extension base="samlp:RequestAbstractType">
1709       <sequence>
1710         <element ref="saml:Subject" minOccurs="0"/>
1711         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
1712         <element ref="saml:Conditions" minOccurs="0"/>
1713         <element ref="samlp:RequestAuthnContext"
1714           minOccurs="0"/>
1715         <element ref="samlp:Scoping" minOccurs="0"/>
1716       </sequence>
1717       <attribute name="IsPassive" type="boolean"
1718         use="optional"/>
1719       <attribute name="ForceAuthn" type="boolean"
1720         use="optional"/>
1721       <attribute name="ProtocolBinding" type="anyURI"
1722         use="optional"/>
1723       <attribute name="AssertionConsumerServiceID" type="string"
1724         use="optional"/>
1725       <attribute name="AssertionConsumerServiceURL"
1726         type="anyURI" use="optional"/>
1727       <attribute name="ProviderName" type="string"
1728         use="optional"/>
1729     </extension>
1730   </complexContent>
1731 </complexType>
1732
```

1733 3.4.1.1 Element `<NameIDPolicy>`

1734 The `<NameIDPolicy>` element tailors the name identifier in the subjects of assertions resulting from an
1735 `<AuthnRequest>`. Its **NameIDPolicyType** complex type defines the following attributes:

1736 **Format [Required]**

1737 Specifies the URI of a name identifier format defined in this or another specification (see Section 7.3
1738 for examples).

1739 **SPNameQualifier [Optional]**

1740 Used with a **Format** of `urn:oasis:names:tc:SAML:2.0:nameid-format:federated` or
1741 `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`, it optionally specifies that a
1742 federated identifier be returned (or created) in the namespace of a service provider other than the
1743 issuing service provider, or an affiliation group.

1744 **AllowCreate [Optional]**

1745 Used to indicate whether the identity provider is allowed MAY, in the course of fulfilling the request, to
1746 create a new identifier to represent the principal. Defaults to "true". When "false", the request issuer
1747 constrains the identity provider to only issue an assertion to it if an acceptable identifier for the
1748 principal has already been established between them.

1749 | When this element is used, if the content is not understood by or acceptable to the identity provider,
1750 | then a <Response> MUST be returned with a <Status> containing a second-level <StatusCode> of
1751 | samlp:InvalidNameIDPolicy.

1752 | ~~A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated expresses the~~
1753 | ~~request issuer's willingness, at the discretion of the requested subject, to establish an identity federation~~
1754 | ~~for the subject with the identity provider, if one does not already exist. But note that when~~
1755 | ~~<NameIDPolicy> is omitted, the identity provider MAY, at its (and the subject's) discretion, also~~
1756 | ~~establish such an identity federation with the understanding that the issuing service provider might~~
1757 | ~~ignore the federated and persistent aspect of the identifier.~~

1758 | A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates that the
1759 | resulting assertion(s) MUST contain <EncryptedIdentifier> elements instead of plaintext. The
1760 | underlying name identifier's unencrypted form can be of any type supported by the identity provider for
1761 | the requested subject.

1762 | Any Format value (or the omission of this element) MAY result in an <EncryptedIdentifier> in the
1763 | resulting assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1764 | The following schema fragment defines the <NameIDPolicy> element and its **NameIDPolicyType**
1765 | complex type:

```
1766 | <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>  
1767 | <complexType name="NameIDPolicyType">  
1768 |   <sequence/>  
1769 |   <attribute name="Format" type="anyURI" use="required"/>  
1770 |   <attribute name="SPNameQualifier" type="string" use="optional"/>  
1771 |   <attribute name="AllowCreate" type="boolean" use="optional"/>  
1772 | </complexType>
```

1773 | 3.4.1.2 Element <RequestAuthnContext>

1774 | The <RequestAuthnContext> element specifies the authentication context requirements of the
1775 | request issuer with respect to the authentication of the presenter. Its **RequestAuthnContextType**
1776 | complex type defines the following elements and attributes:

1777 | <AuthnContextClassRef> or <AuthnContextStatementRef> [One or More]

1778 | Specifies one or more URIs identifying authentication context classes or statements.

1779 | Comparison [Optional]

1780 | Specifies the comparison method used to evaluate the requested context classes or statements, one
1781 | of "exact", "minimum", "maximum", or "better". The default is "exact".

1782 | If <RequestAuthnContext> is specified in an <AuthnRequest> message, the authentication
1783 | statement in the resulting assertion MUST contain an authentication context that conforms to the
1784 | requested context as described below.

1785 | Either a set of class references or statement references can be used. Additionally, the set of supplied
1786 | references MUST be evaluated as an ordered set, where the first element is the most preferred
1787 | authentication context class or statement. If none of the specified classes or statements can be satisfied
1788 | in accordance with the rules below, then the identity provider MUST return a <Response> message with
1789 | a second-level <StatusCode> of samlp:NoAuthnContext.

1790 | If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication
1791 | statement MUST be the exact match of at least one of the authentication contexts specified.

1792 If `Comparison` is set to "minimum", then the resulting authentication context in the authentication
1793 statement MUST be at least as strong (as deemed by the identity provider) as one of the authentication
1794 contexts specified.

1795 If `Comparison` is set to "better", then the resulting authentication context in the authentication statement
1796 MUST be stronger (as deemed by the identity provider) than any one of the authentication contexts
1797 specified.

1798 If `Comparison` is set to "maximum", then the resulting authentication context in the authentication
1799 statement MUST be as strong as possible (as deemed by the identity provider) without exceeding the
1800 strength of at least one of the authentication contexts specified.

1801 The following schema fragment defines the `<RequestAuthnContext>` element and its
1802 **RequestAuthnContextType** complex type:

```
1803 <element name="RequestAuthnContext" type="samlp:RequestAuthnContextType"/>
1804 <complexType name="RequestAuthnContextType">
1805   <choice>
1806     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1807     <element ref="saml:AuthnContextStatementRef"
1808 maxOccurs="unbounded"/>
1809   </choice>
1810
1811   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1812 use="optional"/>
1813 </complexType>
1814 <simpleType name="AuthnContextComparisonType">
1815   <restriction base="string">
1816     <enumeration value="exact"/>
1817     <enumeration value="minimum"/>
1818     <enumeration value="maximum"/>
1819     <enumeration value="better"/>
1820   </restriction>
1821 </simpleType>
```

1822 3.4.1.3 Element `<Scoping>`

1823 The `<Scoping>` element specifies the identity providers trusted by the request issuer to authenticate the
1824 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to
1825 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following
1826 elements and attribute:

1827 `<IDPList>` [Optional]

1828 An advisory list of identity providers and associated information that the request issuer deems
1829 acceptable to respond to the request.

1830 `<RequesterID>` [Zero or More]

1831 Identifies the set requesting entities on whose behalf the request issuer is acting. Used to
1832 communicate the chain of request issuers when proxying occurs, as described in section 3.4.1.9.

1833 `ProxyCount` [Optional]

1834 Specifies the number of proxying indirections permissible between the identity provider that receives
1835 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count
1836 of zero permits no proxying, while omitting this attribute expresses no such restriction.

1837 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a
1838 `<Response>` message with a second-level `<StatusCode>` of `samlp:NoAvailableIDP` or
1839 `samlp:NoSupportedIDP` if it cannot contact or does not support any of the specified identity providers.

1840 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
1841 <element name="Scoping" type="samlp:ScopingType"/>
1842 <complexType name="ScopingType">
1843   <sequence>
1844     <element ref="samlp:IDPList" minOccurs="0"/>
1845     <element ref="samlp:RequesterID" minOccurs="0"
1846     maxOccurs="unbounded"/>
1847   </sequence>
1848   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
1849 </complexType>
1850 <element name="RequesterID" type="anyURI"/>
```

1851 **3.4.1.4 Element <IDPList>**

1852 The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the
1853 presenter. Its **IDPListType** complex type defines the following elements:

1854 <IDPEntry> [One or More]

1855 Information about a single identity provider

1856 <GetComplete> [Optional]

1857 If the <IDPList> is not complete, this element may specify a URI that resolves to the complete list.

1858 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1859 <element name="IDPList" type="samlp:IDPListType"/>
1860 <complexType name="IDPListType">
1861   <sequence>
1862     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
1863     <element ref="samlp:GetComplete" minOccurs="0"/>
1864   </sequence>
1865 </complexType>
1866 <element name="GetComplete" type="anyURI"/>
```

1867 **3.4.1.5 Element <IDPEntry>**

1868 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to
1869 authenticate the presenter. Its **IDPEntryType** complex type defines the following elements:

1870 <ID> [Required]

1871 The unique identifier of the identity provider

1872 <Name> [Optional]

1873 A human readable name for the identity provider

1874 <Loc> [Optional]

1875 The location of a profile-specific endpoint supporting the authentication request protocol. The
1876 binding to be used must be understood from the profile of use.

1877 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1878 <element name="IDPEntry" type="samlp:IDPEntryType"/>
1879 <complexType name="IDPEntryType">
1880   <sequence/>
1881   <attribute name="ID" type="anyURI" use="required"/>
1882   <attribute name="Name" type="string" use="optional"/>
1883   <attribute name="Loc" type="anyURI" use="optional"/>
1884 </complexType>
```

1885 3.4.1.6 Processing Rules

1886 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is
1887 therefore typically profiled for use in a specific context in which this optionality is constrained and specific
1888 kinds of input and output are required or prohibited. The following processing rules apply as invariant
1889 behavior across any profile of this protocol exchange.

1890 The recipient MUST validate any signature present on the request or response message. ~~To be~~
1891 ~~considered valid, the signature provided MUST be the signature of the <Issuer> contained in the~~
1892 ~~message.~~

1893 The responder MUST ultimately reply to an <AuthnRequest> with a <Response> message containing
1894 one or more assertions that meet the specifications defined by the request, or a <Status> describing
1895 the error that occurred. The responder MAY conduct additional message exchanges with the request
1896 sender as needed to initiate or complete the authentication process, subject to the nature of the protocol
1897 binding and the authentication mechanism. As described in the next section, this includes proxying the
1898 request by directing the presenter to another identity provider by issuing its own <AuthnRequest>
1899 message, so that the resulting assertion can be used to authenticate the presenter to the original
1900 responder.

1901 If the responder is unable to authenticate the presenter or does not recognize the requested subject, it
1902 MUST return a <Response> with a <Status> containing a second-level <StatusCode> of
1903 `samlp:UnknownPrincipal`.

1904 If the <Subject> element in the request is present, then the resulting assertions' <Subject> MUST
1905 **strongly match** the request <Subject>, as described in section 3.3.4.1, except that the identifier MAY
1906 be in a different form if specified by <NameIDPolicy>.

1907 All of the content defined specifically within <AuthnRequest> is optional, although some may be
1908 required by certain profiles. In the absence of any specific content at all, the following behavior is
1909 assumed:

- 1910 • The assertion(s) returned MUST contain a <Subject> element that represents the presenter.
1911 The identifier type and format are determined by the identity provider. At least one statement
1912 MUST be an <AuthenticationStatement> that describes the authentication performed by the
1913 responder or authentication service associated with it.
- 1914 • The request presenter should, to the extent possible, be the only entity able to satisfy the
1915 <SubjectConfirmation> of the assertion(s). In the case of weaker confirmation methods,
1916 binding-specific or other mechanisms will be used to help satisfy this requirement.
- 1917 • The resulting assertion(s) MUST contain an <AudienceRestrictionCondition> element
1918 referencing the request issuer as an acceptable relying party. Other audiences MAY be included
1919 as deemed appropriate by the identity provider.

1920 3.4.1.7 Proxying

1921 If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or
1922 cannot directly authenticate him/her, but believes that the presenter has already authenticated to another
1923 identity provider, it may respond to the request by issuing a new <AuthnRequest> on its own behalf to
1924 be presented to the other identity provider. The original identity provider is termed the proxying identity
1925 provider.

1926 Upon the successful return of a <Response> to the proxying provider, the enclosed assertion MAY be
1927 used to authenticate the presenter so that the proxying provider can issue an assertion of its own in
1928 response to the original <AuthnRequest>, completing the overall message exchange. Both the

1929 proxying and authenticating identity providers MAY include constraints on proxying activity in the
1930 messages and assertions they issue, as described in previous sections, and below.

1931 The request issuer can influence proxy behavior by including a <Scoping> element where the provider
1932 sets a desired ProxyCount value and/or indicates a list of preferred identity providers which may be
1933 proxied by including an ordered <IDPList> of preferred providers.

1934 An identity provider can control secondary use of its assertions by proxying identity providers using a
1935 <ProxyRestrictionCondition> element in the assertions it issues.

1936 **3.4.1.7.1 Processing Rules**

1937 An identity provider MAY proxy an <AuthnRequest> if the <ProxyCount> attribute is omitted or is
1938 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY
1939 choose to proxy for a provider specified in the <IDPList>, if provided, but is not required to do so.

1940 An identity provider MUST NOT proxy a request where <ProxyCount> is set to zero. The identity
1941 provider MUST return an error containing a second-level <samlp:StatusCode> value of
1942 samlp:ProxyCountExceeded, unless it can directly authenticate the presenter.

1943 If it chooses to proxy, when creating the new <AuthnRequest>, an identity provider MUST include
1944 equivalent or stricter forms of all the information included in the original request (such as authentication
1945 context policy). Note however that the proxying provider is free to specify whatever <NameIDPolicy> it
1946 wishes to maximize the chances of a successful response.

1947 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST
1948 have some other way to ensure that the elements governing user agent interaction (<IsPassive>, for
1949 example) will be honored by the authenticating provider.

1950 The new <AuthnRequest> MUST contain a <ProxyCount> attribute with a value of at most one less
1951 than the original value. If the original request does not contain a <ProxyCount> attribute, then the new
1952 request SHOULD contain a <ProxyCount> attribute.

1953 If an <IDPList> was specified in the original request, the new request MUST also contain an
1954 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the
1955 <IDPList>, but MUST NOT remove any from the list.

1956 The authentication request and response are processed in normal fashion, in accordance with the rules
1957 given in Section 3.4.1.8 and the profile of use. Once the presenter has authenticated to the proxying
1958 identity provider (by delivering a <Response>), the following steps are followed:

- 1959 • The proxying identity provider prepares a new assertion on its own behalf by copying in the
1960 relevant information from the original assertion. The original assertion will be restricted by
1961 <AudienceRestrictionCondition> to (at least) the proxying identity provider, while the new
1962 assertion's condition will reference (at least) the original request issuer.
- 1963 • The new assertion's <Subject> should contain an identifier that satisfies the original request
1964 issuer's preferences, as defined by its <NameIDPolicy> element.
- 1965 • The <AuthenticationStatement> in the new assertion MUST include an <AuthnContext>
1966 element containing an <ac:AuthenticatingAuthority> element referencing the identity
1967 provider to which the proxying identity provider referred the presenter. If the original assertion
1968 contains <AuthnContext> information that includes one or more
1969 <ac:AuthenticatingAuthority> elements, those elements SHOULD be included in the new
1970 assertion, with the new element placed after them.

- 1971 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider
1972 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be
1973 consistent over time across different requests. The value MUST not conflict with values used or
1974 generated by other SAML providers.
- 1975 • Any other <AuthnContext> information MAY be copied, translated, or omitted in accordance
1976 with the policies of the proxying identity provider, provided that the original requirements dictated
1977 by the request issuer are met.
- 1978 If, in the future, the identity provider is asked to authenticate the same presenter for a second request
1979 issuer, and this request is equally or less strict than the original request, the identity provider MAY skip
1980 the creation of a new <AuthnRequest> to the authenticating identity provider and immediately issue
1981 another assertion (assuming the original assertion it received is still valid). The concrete definition of
1982 "equally or less strict" is up to the proxying identity provider.

1983 3.5 Artifact Protocol

1984 The artifact protocol provides a mechanism by which SAML protocol messages can be transported in a
1985 SAML binding by reference instead of by value. Both requests and responses can be obtained by
1986 reference using this specialized protocol. A message sender, instead of binding a message to a transport
1987 protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of
1988 forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes,
1989 it can then use this protocol in conjunction with a different (generally synchronous) SAML binding
1990 protocol to dereference the artifact into the original protocol message. The most common use for this
1991 mechanism is with bindings that cannot easily carry a message because of size constraints.

1992 Depending on the characteristics of the underlying message being passed by reference, the artifact
1993 protocol MAY require protections such as mutual authentication, integrity protection, confidentiality, etc.
1994 from the protocol binding used to dereference the artifact. In all cases, the artifact MUST exhibit a single-
1995 use semantic such that once it has been successfully dereferenced, it can no longer be used by any
1996 party.

1997 Regardless of the protocol message obtained, the result of dereferencing an artifact MUST be treated
1998 exactly as if the message so obtained had been sent originally in place of the artifact.

1999 3.5.1 Element <ArtifactRequest>

2000 The <ArtifactRequest> message is used to request that a protocol message be returned in an
2001 <ArtifactResponse> message by specifying an artifact that represents the protocol message. The
2002 original transmission of the artifact is governed by the specific binding or profile of SAML that is being
2003 used; see the SAML specifications for bindings [SAMLBind] and profiles [SAMLProf] for more information
2004 on the use of artifacts in bindings and profiles.

2005 The <ArtifactRequest> message SHOULD be signed or otherwise authenticated and integrity
2006 protected by the protocol binding used to deliver the message.

2007 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a
2008 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2009 This message has the complex type **ArtifactRequestType**, which extends **RequestAbstractType** and
2010 adds the following element:

2011 <Artifact> [Required]

2012 The artifact value that the requester received and now wishes to translate into the protocol message
2013 it represents. See [SAMLBind] for specific artifact format information.

2014 The following schema fragment defines the <ArtifactRequest> element and its
2015 **ArtifactRequestType** complex type:

```
2016 <element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>
2017 <complexType name="ArtifactRequestType">
2018   <complexContent>
2019     <extension base="samlp:RequestAbstractType">
2020       <sequence>
2021         <element ref="samlp:Artifact"/>
2022       </sequence>
2023     </extension>
2024   </complexContent>
2025 </complexType>
2026 <element name="Artifact" type="string"/>
```

2027 3.5.2 Element <ArtifactResponse>

2028 The recipient of an <ArtifactRequest> message MUST respond with an <ArtifactResponse>
2029 message, which is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a
2030 single optional wildcard element corresponding to the protocol message being returned. This wrapped
2031 message element can be a request or a response.

2032 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity
2033 protected by the protocol binding used to deliver the message.

2034 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a
2035 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2036 The following schema fragment defines the <ArtifactResponse> element and its
2037 **ArtifactResponseType** complex type:

```
2038 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
2039 <complexType name="ArtifactResponseType">
2040   <complexContent>
2041     <extension base="samlp:StatusResponseType">
2042       <sequence>
2043         <any namespace="#any" processContents="lax"
2044           minOccurs="0"/>
2045       </sequence>
2046     </extension>
2047   </complexContent>
2048 </complexType>
```

2049 3.5.3 Processing Rules

2050 The recipient MUST validate any signature present on the request or response message. ~~To be~~
2051 ~~considered valid, the signature provided MUST be the signature of the <Issuer> contained in the~~
2052 ~~message.~~

2053 If the responder recognizes the artifact as valid, then it responds with the associated protocol message
2054 in an <ArtifactResponse> message. Otherwise, it responds with an <ArtifactResponse>
2055 message with no embedded message. In both cases, the <Status> element MUST include a
2056 <StatusCode> element with the code value Success. A response message with no embedded
2057 message inside it is termed an empty response in the remainder of this section.

2058 The responder MUST enforce a one-time-use property on the artifact by insuring that any subsequent
2059 request with the same artifact by any requester results in an empty response as described above.

2060 Some SAML protocol messages, most particularly the <AuthnRequest> message in some profiles,
2061 MAY be intended for consumption by any party that receives it and can respond appropriately. In most

2062 other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued
2063 MUST be associated with the intended recipient of the message that the artifact represents. If the artifact
2064 issuer receives an <ArtifactRequest> from a requester that cannot authenticate itself as the original
2065 intended recipient, then the artifact issuer MUST return an empty response.

2066 The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such
2067 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and
2068 return it in an <ArtifactRequest> to the issuer.

2069 Note that the <ArtifactResponse>'s InResponseTo attribute MUST contain the value of the
2070 corresponding <AssertionRequest>'s RequestID attribute, but the embedded protocol message will
2071 contain its own message identifier, and in the case of an embedded response, may contain a different
2072 InResponseTo value that corresponds to the original request message to which the embedded
2073 message is responding.

2074 **3.6 Federated Name Identifier Registration Management Protocol**

2075 ~~If a~~After establishing a persistent name identifier for a principal, an identity provider wishes to change
2076 the value and/or format of the identifier that it will use when referring to the principal, or to indicate that a
2077 name identifier will no longer be used to refer to the principal, then it MAY inform service providers of
2078 this change by sending them a <ManageNameIdentifierRequest> message. The identity
2079 provider MAY send a similar message to indicate that a name identifier will no longer be used to refer to
2080 the principal.

2081 ~~The same message MAY also be used by a service provider to register or change the~~
2082 ~~SPProvidedIdentifier value to be included when the underlying name identifier is used to~~
2083 ~~communicate with it, or to terminate the use of a name identifier between itself and the identity provider.~~

2084 ~~When an identity provider and service provider first federate a principal's identity using a~~
2085 ~~<NameIdentifier> element with a Format of urn:oasis:names:tc:SAML:2.0:nameid-~~
2086 ~~format:federated, the identity provider generates an opaque value that serves as the initial name~~
2087 ~~identifier that both the service provider and the identity provider use in referring to the principal when~~
2088 ~~communicating with each other.~~

2089 ~~Subsequent to federation, the service provider MAY register a different opaque value with the identity~~
2090 ~~provider. This opaque value is an attribute termed the SPProvidedIdentifier. Until the service provider~~
2091 ~~registers a different name, this attribute is omitted from <NameIdentifier> elements referring to the~~
2092 ~~principal.~~

2093 ~~Either the service provider or the identity provider MAY register a new name identifier for a principal with~~
2094 ~~each other at any time following federation. The name identifiers specified by providers SHOULD be~~
2095 ~~unique across the identity providers with which the principal's identity is federated and SHOULD be~~
2096 ~~unique within the group of name identifiers that have been registered with the identity provider by this~~
2097 ~~service provider.~~

2098 ~~Only federated identifiers (as defined by a Format of urn:oasis:names:tc:SAML:2.0:nameid-~~
2099 ~~format:federated) can be replaced and set with this protocol; non-federated, encrypted, or transient~~
2100 ~~identifiers MUST NOT be used.~~

2101 **3.6.1 Element <RegisterManageNameIdentifierRequest>**

2102 ~~To register an SPProvidedIdentifier attribute with an identity provider, the service provider sends a~~
2103 ~~<RegisterNameIdentifierRequest> message. The same message may be sent by an identity~~
2104 ~~provider, seeking to change the <NameIdentifier> value stored by the service provider.~~

2105 A provider sends a <ManageNameIdentifierRequest> message to inform the recipient of a changed
2106 name identifier or to indicate the termination of the use of a name identifier.

2107 The <RegisterManageNameIdentifierRequest> message SHOULD be signed or otherwise
2108 authenticated and integrity protected by the protocol binding used to deliver the message

2109 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a
2110 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2111 This message has the complex type **RegisterNameIdentifierRequestType**, which extends
2112 **RequestAbstractType** and adds the following elements:

2113 <NameIdentifier> [Required]

2114 The ~~federated~~ name identifier and associated attributes that specify the principal as currently
2115 recognized by the identity and service providers prior to this request.

2116 <NewIdentifier> or <TerminateIdentifier> [Required]

2117 The new ~~federated~~ identifier value to be used when communicating with the requesting provider
2118 concerning this principal or an indication that the use of the old identifier has been terminated. In the
2119 former case, if the requester is the service provider, the new identifier ~~will~~**MUST** appear in
2120 subsequent <NameIdentifier> elements in the SPPprovidedIdentifier attribute. If the
2121 requester is the identity provider, the new value will appear in subsequent <NameIdentifier>
2122 elements as the element's value.

2123 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its
2124 **RegisterNameIdentifierRequestType** complex type:

```
2125 <element name="NewIdentifier" type="string">  
2126 <element name="RegisterManageNameIdentifierRequest"  
2127 type="samlp:RegisterManageNameIdentifierRequestType"/>  
2128 <complexType name="RegisterManageNameIdentifierRequestType">  
2129 <complexContent>  
2130 <extension base="samlp:RequestAbstractType">  
2131 <sequence>  
2132 <element ref="saml:NameIdentifier"/>  
2133 <choice>  
2134 <element ref="samlp:NewIdentifier"/>  
2135 <element ref="samlp:TerminateIdentifier"/>  
2136 </choice>  
2137 </sequence>  
2138 </extension>  
2139 </complexContent>  
2140 </complexType>  
2141 <element name="NewIdentifier" type="string"/>  
2142 <element name="TerminateIdentifier" type="samlp:TerminateIdentifierType"/>  
2143 <complexType name="TerminateIdentifierType">  
2144 <sequence/>  
2145 </complexType>
```

2146 3.6.2 Element <RegisterManageNameIdentifierResponse>

2147 The recipient of a <RegisterManageNameIdentifierRequest> message MUST respond with a
2148 <RegisterManageNameIdentifierResponse> message, which is of type **StatusResponseType**
2149 with no additional content.

2150 The <RegisterManageNameIdentifierResponse> message SHOULD be signed or otherwise
2151 authenticated and integrity protected by the protocol binding used to deliver the message.

2152 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a
2153 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2154 The following schema fragment defines the <RegisterNameIdentifierResponse> element:

```
2155 | <element name="RegisterManageNameIdentifierResponse"  
2156 | type="samlp:StatusResponseType"/>
```

2157 3.6.3 Processing Rules

2158 The recipient MUST validate any signature present on the request or response message. ~~To be~~
2159 ~~considered valid, the signature provided MUST be the signature of the <Issuer> contained in the~~
2160 ~~message.~~

2161 If the request includes a <NameIdentifier> ~~for which no federation exists between the service~~
2162 ~~provider and the identity provider that the recipient does not recognize~~, the responding provider
2163 **MUSTMAY** respond with a <Status> containing a second-level <StatusCode> of
2164 samlp:~~FederationDoesNotExistUnknownPrincipal~~.

2165 ~~If the <TerminateIdentifier> element is included in the request, the requesting provider is~~
2166 ~~indicating that (in the case of a service provider) it will no longer accept assertions from the identifier~~
2167 ~~provider or (in the case of an identity provider) it will no longer issue assertions to the service provider~~
2168 ~~about the principal. The receiving provider MAY perform any maintenance with the knowledge that the~~
2169 ~~relationship represented by the name identifier has been terminated. It MAY choose to invalidate the~~
2170 ~~active session(s) of a principal for whom a relationship has been terminated.~~

2171 If the service provider requests that its identifier be changed ~~by including a <NewIdentifier> element~~,
2172 the identity provider MUST include the <NewIdentifier> element's value as the
2173 SPProvidedIdentifier when subsequently communicating to the service provider regarding this
2174 principal.

2175 If the identity provider requests that its identifier be changed ~~by including a <NewIdentifier> element~~,
2176 the service provider MUST use the <NewIdentifier> element's value as the <NameIdentifier>
2177 element value when subsequently communicating with the identity provider regarding this principal.

2178 In ~~either any~~ case, the <NameIdentifier> value in the request and its associated
2179 SPProvidedIdentifier attribute MUST contain the most recent name identifier information
2180 established between the providers for the principal.

2181 ~~In the case of a federated name identifier, the~~ NameQualifier attribute MUST contain the unique
2182 identifier of the identity provider. ~~If the principal's identity federation is between the identity provider~~
2183 ~~and an affiliation group of which the service provider is a member, then the SPNameQualifier attribute~~
2184 ~~MUST contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique~~
2185 ~~identifier of the service provider.~~

2186 Changes to these identifiers may take a potentially significant amount of time to propagate through the
2187 systems at both the requester and the responder. Implementations might wish to allow each party to
2188 accept either identifier for some period of time following the successful completion of a name identifier
2189 change. Not doing so could result in the inability of the principal to access resources.

2190 All other processing rules associated with the underlying request and response messages MUST be
2191 observed.

2192 3.7 Federation Termination Protocol

2193 ~~When a principal (or an appropriate agent acting on his or her behalf) terminates an identity federation~~
2194 ~~between a service provider and an identity provider through an interaction with the service provider, the~~
2195 ~~service provider MUST send a <FederationTerminationNotification> message to the identity~~
2196 ~~provider. The service provider is stating that it will no longer accept authentication assertions from the~~
2197 ~~identity provider for the specified principal.~~

2198 Likewise, when a principal terminates an identity federation through an interaction with the identity
2199 provider, the identity provider **MUST** send a `<FederationTerminationNotification>` message to
2200 the service provider. In this case, the identity provider is stating that it will no longer provide
2201 authentication assertions to the service provider for the specified principal.

2202

2203 **3.7.1 Element `<FederationTerminationNotification>`**

2204 A provider sends a `<FederationTerminationNotification>` to the provider with which it is
2205 terminating a federation. The `<FederationTerminationNotification>` message **SHOULD** be
2206 signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the
2207 message.

2208 The `<Issuer>` of the request **MUST** contain the unique identifier of the requesting provider, with a
2209 format value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

2210 This message has the complex type **FederationTerminationNotificationType**, which extends
2211 **RequestAbstractType** and adds the following elements:

2212 `<NameIdentifier>` [Required]

2213 The federated name identifier and associated attributes that specify the principal as currently
2214 recognized by the identity and service providers prior to this request. Format **MUST** be
2215 `urn:oasis:names:tc:SAML:2.0:nameid-format:federated`.

2216 The following schema fragment defines the `<RegisterNameIdentifierRequest>` element and its
2217 **RegisterNameIdentifierRequestType** complex type:

```
2218 <element name="FederationTerminationNotification"  
2219 type="samlp:FederationTerminationNotificationType"/>  
2220 <complexType name="FederationTerminationNotificationType">  
2221 <complexContent>  
2222 <extension base="samlp:RequestAbstractType">  
2223 <sequence>  
2224 <element ref="saml:NameIdentifier"/>  
2225 </sequence>  
2226 </extension>  
2227 </complexContent>  
2228 </complexType>
```

2229 **3.7.2 Element `<FederationTerminationResponse>`**

2230 The recipient of a `<FederationTerminationNotification>` message **MUST** respond with a
2231 `<FederationTerminationResponse>` message, which is of type **StatusResponseType** with no
2232 additional content.

2233 The `<FederationTerminationResponse>` message **SHOULD** be signed or otherwise authenticated
2234 and integrity protected by the protocol binding used to deliver the message.

2235 The `<Issuer>` of the response **MUST** contain the unique identifier of the responding provider, with a
2236 format value of `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

2237 The following schema fragment defines the `<FederationTerminationResponse>` element:

```
2238 <element name="FederationTerminationResponse"  
2239 type="samlp>StatusResponseType"/>
```


2240 | **3.7.3 Processing Rules**

2241 | ~~The recipient MUST validate any signature present on the request or response message. To be~~
2242 | ~~considered valid, the signature provided MUST be the signature of the <Issuer> contained in the~~
2243 | ~~message.~~

2244 | ~~If the request includes a <NameIdentifier> for which no federation exists between the service~~
2245 | ~~provider and the identity provider, the responding provider MUST respond with a <samlp:Status>~~
2246 | ~~containing a second level <samlp:StatusCode> of samlp:FederationDoesNotExist.~~

2247 | ~~Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been~~
2248 | ~~terminated. A provider MAY choose to invalidate the session of a user for whom federation has been~~
2249 | ~~terminated.~~

2250 | ~~All other processing rules associated with the underlying request and response messages MUST be~~
2251 | ~~observed.~~

2252 | **3.8 Single Logout Protocol**

2253 | The single logout protocol provides a message exchange protocol by which all sessions provided by a
2254 | particular session authority are near-simultaneously terminated. The single logout protocol is used either
2255 | when a principal logs out at a session participant or when the principal logs out directly at the
2256 | session authority. This protocol may also be used to logout a principal due to a timeout. The reason for
2257 | the logout event may be indicated through the `reason` attribute.

2258 |
2259 | The principal may have established authenticated sessions both with the session authority, and
2260 | individual session participants, based on authentication assertions supplied by the session authority.
2261 |

2262 | When the principal invokes the single logout process at a session participant, the session participant
2263 | MUST send a <LogoutRequest> message to the session authority that provided the authentication
2264 | service related to that session at the session participant.
2265 |

2266 | When either the principal invokes a logout at the session authority, or a session participant sends a
2267 | logout request to the session authority specifying that principal, the session authority MUST send a
2268 | <LogoutRequest> message to each session participant to which it provided authentication assertions
2269 | under its current session with the principal, with the exception of the session participant that sent the
2270 | <LogoutRequest> message to the session authority.

2272 | **3.8.1 Element <LogoutRequest>**

2273 | A session participant or session authority sends a <LogoutRequest> message to indicate that a
2274 | session has been terminated.

2275 | The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity
2276 | protected by the protocol binding used to deliver the message.

2277 | This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType**, and
2278 | adds the following elements and attributes:

2279 | <NameIdentifier> [Required]

2280 | The name identifier and associated attributes that specify the principal as currently recognized by
2281 | the identity and service providers prior to this request.

2282 <SessionIndex> [Optional]

2283 The identifier that indexes this session at the message recipient.

2284 NotOnOrAfter [Optional]

2285 The time at which the request expires.

2286 Reason [Optional]

2287 An indication of the reason for the logout, in the form of a URI reference.

2288 The following schema fragment defines the <LogoutRequest> element and associated
2289 **LogoutRequestType** complex type:

```
2290 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
2291 <complexType name="LogoutRequestType">
2292   <complexContent>
2293     <extension base="saml:RequestAbstractType">
2294       <sequence>
2295         <element ref="saml:NameIdentifier"/>
2296         <element ref="SessionIndex"/>
2297       </sequence>
2298       <attribute name="Reason" type="anyURI" minOccurs="0"/>
2299       <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
2300     </extension>
2301   </complexContent>
2302 </complexType>
2303 <element name="SessionIndex" type="string" minOccurs="0"
2304 maxOccurs="unbounded"/>
```

2305 3.8.2 Element <LogoutResponse>

2306 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message,
2307 of type **StatusResponseType**, with no additional content specified.

2308 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity
2309 protected by the protocol binding used to deliver the message.

2310 The following schema fragment defines the <LogoutResponse> element:

```
2311 <element name="LogoutResponse" type="saml:StatusResponseType"/>
```

2312 3.8.3 Processing Rules

2313 The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting or
2314 responding provider, with a `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-`
2315 `format:provider`.

2316 Message recipients MUST validate any signature present on the messages specified in this protocol. ~~To~~
2317 ~~be considered valid, the signature provided must be the signature of the <Issuer> contained in the~~
2318 ~~message.~~

2319 The message sender MAY use the `Reason` attribute to indicate the reason for sending the
2320 <LogoutRequest>. Other values MAY be agreed upon between participants, but the following values
2321 are defined directly by this specification for use by all message senders:

2322 `urn:oasis:names:tc:SAML:2.0:logout:user`

2323 Specifies that the message is being sent because the principal wishes to terminate the indicated
2324 session.

2325 urn:oasis:names:tc:SAML:2.0:logout:admin

2326 Specifies that the message is being sent because an administrator wishes to terminate the indicated
2327 session for that principal.

2328 All other processing rules associated with the underlying request and response messages MUST be
2329 observed.

2330 **3.8.3.1 Session Participant Rules**

2331 When a session participant receives a <LogoutRequest>, the session participant MUST authenticate
2332 the message.. If the sender is the authority that provided an assertion linked to the principal's current
2333 session, the session participant MUST invalidate the principal's session(s) referred to by the
2334 <NameIdentifier> element, and any <SessionIndex> elements supplied in the message.

2335
2336 The session participant MUST apply the logout request message to any assertion that meets the
2337 following conditions, even if the assertion arrives after the logout request:

- 2338 • The <SessionIndex> of the assertion's statements matches one specified in the logout request.
- 2339 • The assertion would otherwise be valid
- 2340 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on
2341 the message).

2342 **3.8.3.2 Session Authority Rules**

2343 When a session authority receives a <LogoutRequest>, the session authority MUST authenticate the
2344 sender. If the sender is a session participant to which the session authority provided an assertion for the
2345 current session, then the session authority SHOULD do the following:

- 2346 • Send a <LogoutRequest> message to each session participant for which the session authority
2347 provided assertions in the current session, *other than the originator of a current*
2348 *<LogoutRequest>*.
- 2349 • Send a <LogoutRequest> message to any session authority on behalf of whom the session
2350 authority proxied the user's authentication, unless the second authority is the originator of the
2351 <LogoutRequest>.
- 2352 • Terminate the principal's current session as specified by the <NameIdentifier> element, and
2353 any <SessionIndex> elements present in the logout request message.

2354 It should be noted that a session authority MAY initiate a logout for reasons other than having received a
2355 <LogoutRequest> from a session participant – these include, but are not limited to:

- 2356 • If some timeout period was agreed out-of-band with an individual session participant, the session
2357 authority MAY send a <LogoutRequest> to that individual participant alone.
- 2358 • An agreed global timeout period has been exceeded.
- 2359 • The principal, or some other trusted entity has requested logout of the principal, directly at the
2360 session authority.
- 2361 • The session authority has determined that the principal's credentials may have been compromised.

2362 When constructing a logout request message, the session authority MUST set the value of the
2363 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message.

2364 In addition to the values specified in section 3.6.3 for the `Reason` attribute, the following values are also
2365 available for use by the session authority only:

2366 urn:oasis:names:tc:SAML:2.0:logout:global-timeout

2367 Specifies that the message is being sent because of the global session timeout interval period
2368 being exceeded.

2369 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2370 Specifies that the message is being sent because a timeout interval period agreed between a
2371 participant and the authority has been exceeded.

2372 If an error occurs during this further processing of the logout (for example, relying session participants
2373 may not all implement the particular single logout protocol binding used by the requesting session
2374 participant), then the session authority MUST respond to the original requester with a
2375 <LogoutResponse> message, indicating the status of the logout request. The value
2376 samlp:UnsupportedBinding is provided for a second-level <samlp:StatusCode>, indicating that a
2377 session participant should retry the <LogoutRequest> using a different protocol binding.

2378 3.9 Name Identifier Mapping Protocol

2379 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name
2380 identifier for the same principal in a particular format or federation namespace, it can send a request to
2381 the identity provider using this protocol.

2382 For example, a service provider that wishes to communicate with another service provider with whom it
2383 does not share an identity federation for the principal can use an identity provider that shares an identity
2384 federation for the principal with both service providers to map from its own federated identifier to a new
2385 identifier, generally encrypted, with which it can communicate with the second service provider.

2386 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into an
2387 <EncryptedIdentifier> element unless a specific deployment dictates such protection is
2388 unnecessary.

2389 3.9.1 Element <NameIdentifierMappingRequest>

2390 To request an alternate name identifier for a principal from an identity provider, a requester sends an
2391 <NameIdentifierMappingRequest> message. This message has the complex type
2392 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following
2393 element:

2394 <BaseIdentifier> or <NameIdentifier> or <EncryptedIdentifier> [Required]

2395 The identifier and associated attributes that specify the principal as currently recognized by the
2396 requester and the responder.

2397 <NameIDPolicy>

2398 The format and optional name qualifier that describes the requirements for the identifier to be
2399 returned.

2400 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
2401 binding used to deliver the message.

2402 The following schema fragment defines the <NameIdentifierMappingRequest> element and its
2403 **NameIdentifierMappingRequestType** complex type:

```
2404 <element name="NameIdentifierMappingRequest"  
2405 type="samlp:NameIdentifierMappingRequestType"/>  
2406 <complexType name="NameIdentifierMappingRequestType">
```

```

2407     <complexContent>
2408         <extension base="samlp:RequestAbstractType">
2409             <sequence>
2410                 <choice>
2411                     <element ref="saml:BaseIdentifier"/>
2412                     <element ref="saml:NameIdentifier"/>
2413                     <element ref="saml:EncryptedIdentifier"/>
2414                 </choice>
2415                 <element ref="samlp:NameIDPolicy"/>
2416             </sequence>
2417         </extension>
2418     </complexContent>
2419 </complexType>

```

2420 3.9.2 Element <NameIdentifierMappingResponse>

2421 The recipient of a <NameIdentifierMappingRequest> message MUST respond with a
 2422 <NameIdentifierMappingResponse> message. This message has the complex type
 2423 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following
 2424 element:

2425 <NameIdentifier> or <EncryptedIdentifier> [Required]

2426 The identifier and associated attributes that specify the principal in the manner requested, usually in
 2427 encrypted form.

2428 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol
 2429 binding used to deliver the message.

2430 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a
 2431 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2432 The following schema fragment defines the <NameIdentifierMappingResponse> element and its
 2433 **NameIdentifierMappingResponseType** complex type:

```

2434 <element name="NameIdentifierMappingResponse"
2435 type="samlp:NameIdentifierMappingResponseType"/>
2436 <complexType name="NameIdentifierMappingResponseType">
2437     <complexContent>
2438         <extension base="samlp:StatusResponseType">
2439             <choice>
2440                 <element ref="saml:NameIdentifier">
2441                 <element ref="saml:EncryptedIdentifier">
2442             </choice>
2443         </extension>
2444     </complexContent>
2445 </complexType>

```

2446 3.9.3 Processing Rules

2447 The recipient MUST validate any signature present on the request or response message. ~~To be~~
 2448 ~~considered valid, the signature provided MUST be the signature of the <Issuer> contained in the~~
 2449 ~~message.~~

2450 If the responder does not recognize the principal identified in the request, it **MUSTMAY** respond with a
 2451 <Status> containing a second-level <StatusCode> of samlp:UnknownPrincipal.

2452 At the responder's discretion, the samlp:InvalidNameIDPolicy status code MAY be returned to
 2453 indicate an inability or unwillingness to supply an identifier in the requested format. ~~Likewise, the~~
 2454 ~~samlp:FederationDoesNotExist status code MAY be used to indicate that a requested federated~~
 2455 ~~identifier cannot be returned.~~

2456 All other processing rules associated with the underlying request and response messages MUST be
2457 observed.

2458 4 SAML Versioning

2459 The SAML specification set is versioned in two independent ways. Each is discussed in the following
2460 sections, along with processing rules for detecting and handling version differences, when applicable.
2461 Also included are guidelines on when and why specific version information is expected to change in
2462 future revisions of the specification.

2463 When version information is expressed as both a Major and Minor version, it may be expressed
2464 discretely, or in the form *Major.Minor*. The version number *Major_B.Minor_B* is higher than the version
2465 number *Major_A.Minor_A* if and only if:

2466 $Major_B > Major_A \vee ((Major_B = Major_A) \wedge Minor_B > Minor_A)$

2467 4.1 SAML Specification Set Version

2468 Each release of the SAML specification set will contain a major and minor version designation describing
2469 its relationship to earlier and later versions of the specification set. The version will be expressed in the
2470 content and filenames of published materials, including the specification set document(s), and XML
2471 schema instance(s). There are no normative processing rules surrounding specification set versioning,
2472 since it merely encompasses the collective release of normative specification documents which
2473 themselves contain processing rules.

2474 The overall size and scope of changes to the specification set document(s) will informally dictate whether
2475 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards
2476 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics
2477 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a
2478 major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0,
2479 described in Section .

2480 4.1.1 Schema Version

2481 As a non-normative documentation mechanism, any XML schema instances published as part of the
2482 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the
2483 specification set version in which it has been published. Validating implementations MAY use the
2484 attribute as a means of distinguishing which version of a schema is being used to validate messages, or
2485 to support a multiplicity of versions of the same logical schema.

2486 4.1.2 SAML Assertion Version

2487 The SAML <Assertion> element contains attributes for expressing the major and minor version of the
2488 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to
2489 document the syntax, semantics, and processing rules of the assertions of the same version. That is,
2490 specification set version 1.0 describes assertion version 1.0, and so on.

2491 There is explicitly NO relationship between the assertion version and the SAML assertion XML
2492 namespace that contains the schema definitions for that assertion version.

2493 The following processing rules apply:

- 2494 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported
2495 by the authority.
- 2496 • A SAML relying party MUST NOT process any assertion with a major assertion version number not
2497 supported by the relying party.

- 2498 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version
2499 number is higher than the minor assertion version number supported by the relying party. However,
2500 all assertions that share a major assertion version number MUST share the same general processing
2501 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1
2502 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0
2503 assertion without ill effect.

2504 **4.1.3 SAML Protocol Version**

2505 The SAML protocol `<Request>` and `<Response>` elements contain attributes for expressing the major
2506 and minor version of the request or response message using a pair of integers. Each version of the
2507 SAML specification set will be construed so as to document the syntax, semantics, and processing rules
2508 of the protocol messages of the same version. That is, specification set version 1.0 describes request
2509 and response version V1.0, and so on.

2510 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace
2511 that contains the schema definitions for protocol messages for that protocol version.

2512 The version numbers used in SAML protocol `<Request>` and `<Response>` elements will be the same
2513 for any particular revision of the SAML specification set.

2514 **4.1.3.1 Request Version**

2515 The following processing rules apply to requests:

- 2516 • A SAML requester SHOULD issue requests with the highest request version supported by both the
2517 SAML requester and the SAML responder.
- 2518 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume
2519 that it supports requests with the highest request version supported by the requester.
- 2520 • A SAML requester MUST NOT issue a request message with a request version number matching a
2521 response version number that the requester does not support.
- 2522 • A SAML responder MUST reject any request with a major request version number not supported by
2523 the responder.
- 2524 • A SAML responder MAY process or MAY reject any request whose minor request version number is
2525 higher than the highest supported request version that it supports. However, all requests that share a
2526 major request version number MUST share the same general processing rules and semantics, and
2527 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax
2528 of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

2529 **4.1.4 Response Version**

2530 The following processing rules apply to responses:

- 2531 • A SAML responder MUST NOT issue a response message with a response version number higher
2532 than the request version number of the corresponding request message.
- 2533 • A SAML responder MUST NOT issue a response message with a major response version number
2534 lower than the major request version number of the corresponding request message except to report
2535 the error `RequestVersionTooHigh`.

2536 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-
2537 level `<StatusCode>` value of `VersionMismatch`, and MAY result in reporting one of the following

2538 **second-level values:** RequestVersionTooHigh, RequestVersionTooLow, or
2539 RequestVersionDeprecated.

2540 **4.1.5 Permissible Version Combinations**

2541 In general, assertions of a particular major version may appear in response messages of the same major
2542 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol
2543 schema. Future versions of this specification are expected to explicitly describe the permitted
2544 combinations across major versions.

2545 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to
2546 appear in a V1.1 response message.

2547 **4.2 SAML Namespace Version**

2548 XML schema instances and "qualified names" (QNames) published as part of the specification set
2549 contain one or more target namespaces into which the type, element, and attribute definitions are
2550 placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and
2551 syntactical definitions that make up that part of the specification.

2552 The namespace URIs defined by the specification set will generally contain version information of the
2553 form *Major.Minor* somewhere in the URI. The major and minor version in the URI **MUST** correspond to
2554 the major and minor version of the specification set in which the namespace is first introduced and
2555 defined. This information is not typically consumed by an XML processor, which treats the namespace
2556 opaquely, but is intended to communicate the relationship between the specification set and the
2557 namespaces it defines.

2558 As a general rule, implementers can expect the namespaces (and the associated schema definitions)
2559 defined by a major revision of the specification set to remain valid and stable across minor revisions of
2560 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,
2561 but this is expected to be rare. In such cases, the older namespaces and their associated definitions
2562 should be expected to remain valid until a major specification set revision.

2563 **4.2.1 Schema Evolution**

2564 In general, maintaining namespace stability while adding or changing the content of a schema are
2565 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how
2566 older implementations will react to any given change, making forward compatibility difficult to achieve.
2567 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of
2568 namespace stability. Except in special circumstances (for example to correct major deficiencies or fix
2569 errors), implementations should expect forward compatible schema changes in minor revisions, allowing
2570 new messages to validate against older schemas.

2571 Implementations **SHOULD** expect and be prepared to deal with new extensions and message types in
2572 accordance with the processing rules laid out for those types. Minor revisions **MAY** introduce new types
2573 that leverage the extension facilities described in Section SAML Extensions. Older implementations
2574 **SHOULD** reject such extensions gracefully when they are encountered in contexts that dictate mandatory
2575 semantics. Examples include new query, statement, or condition types.

2576

5 SAML and XML Signature Syntax and Processing

2577 SAML assertions and SAML protocol request and response messages may be signed, with the following
2578 benefits:

- 2579 • An assertion signed by the SAML authority supports:
 - 2580 – Assertion integrity.
 - 2581 – Authentication of the SAML authority to a SAML relying party.
 - 2582 – If the signature is based on the SAML authority's public-private key pair, then it also provides for
2583 non-repudiation of origin.
- 2584 • A SAML protocol request or response message signed by the message originator supports:
 - 2585 – Message integrity.
 - 2586 – Authentication of message origin to a destination.
 - 2587 – If the signature is based on the originator's public-private key pair, then it also provides for non-
2588 repudiation of origin.

2589 A digital signature is not always required in SAML. For example, it may not be required in the following
2590 situations:

- 2591 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains
2592 protection from a signature on the containing protocol response message. "Inherited" signatures
2593 should be used with care when the contained object (such as the assertion) is intended to have a
2594 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,
2595 exposing the XML content and adding potentially unnecessary overhead.
- 2596 • The SAML relying party or SAML requester may have obtained an assertion or protocol message
2597 from the SAML authority or SAML responder directly (with no intermediaries) through a secure
2598 channel, with the SAML authority or SAML responder having authenticated to the relying party or
2599 SAML responder by some means other than a digital signature.

2600 Many different techniques are available for "direct" authentication and secure channel establishment
2601 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
2602 the applicable security requirements depend on the communicating applications and the nature of the
2603 assertion or message transported.

2604 It is recommended that, in all other contexts, digital signatures be used for assertions and request and
2605 response messages. Specifically:

- 2606 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority
2607 SHOULD be signed by the SAML authority.
- 2608 • A SAML protocol message arriving at a destination from an entity other than the originating site
2609 SHOULD be signed by the origin site.

2610 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that
2611 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures
2612 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure
2613 compatibility with profiles that may require other mechanisms.

2614 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST
2615 be used if signing.

2616 **5.1 Signing Assertions**

2617 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema
2618 as described in Section Assertions.

2619 **5.2 Request/Response Signing**

2620 All SAML protocol request and response messages MAY be signed using the XML Signature. This is
2621 reflected in the schema as described in Sections Requests and Responses and .

2622 **5.3 Signature Inheritance**

2623 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`
2624 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a
2625 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a
2626 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,
2627 then the assertion can be considered to inherit the signature from the enclosing element. The resulting
2628 interpretation should be equivalent to the case where the assertion itself was signed with the same key
2629 and signature options.

2630 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as
2631 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may
2632 define additional rules for interpreting SAML elements as inheriting signatures or other authentication
2633 information from the surrounding context, but no such inheritance should be inferred unless specifically
2634 identified by the profile.

2635 **5.4 XML Signature Profile**

2636 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
2637 and many choices. This section details the constraints on these facilities so that SAML processors do not
2638 have to deal with the full generality of XML Signature processing. This usage makes specific use of the
2639 **xsd:ID**-typed attributes optionally present on the root elements to which signatures can apply: the
2640 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the
2641 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section
2642 as the identifier attributes.

2643 **5.4.1 Signing Formats and Algorithms**

2644 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
2645 detached.

2646 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol
2647 messages. SAML processors SHOULD support the use of RSA signing and verification for public key
2648 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

2649 **5.4.2 References**

2650 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the
2651 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element
2652 may or may not be the root element of the actual XML document containing the signed assertion or
2653 message.

2654 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier
2655 attribute value of the root element of the message being signed. For example, if the attribute value is
2656 "foo", then the URI attribute in the `<ds:Reference>` element MUST be "#foo".

2657 **5.4.3 Canonicalization Method**

2658 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the
2659 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`
2660 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages
2661 embedded in an XML context can be verified independent of that context.

2662 **5.4.4 Transforms**

2663 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature
2664 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive
2665 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
2666 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2667 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
2668 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This
2669 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by
2670 applying the transforms manually to the content and reverifying the result as consisting of the same
2671 SAML message.

2672 **5.4.5 KeyInfo**

2673 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the
2674 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
2675 be absent.

2676 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2677 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section
2678 SAML Assertions.

2679 **5.4.7 Example**

2680 Following is an example of a signed response containing a signed assertion. Line breaks have been
2681 added for readability; the signatures are not valid and cannot be successfully verified.

```
2682 <Response  
2683   IssueInstant="2003-04-17T00:46:02Z"  
2684   MajorVersion="1"  
2685   MinorVersion="1"  
2686   Recipient="www.opensaml.org"  
2687   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
2688   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
2689   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
2690   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2691   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
2692 <ds:Signature  
2693   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
2694 <ds:SignedInfo>  
2695 <ds:CanonicalizationMethod  
2696   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2697 <ds:SignatureMethod
```

```

2698   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
2699 <ds:Reference
2700   URI="#_c7055387-af61-4fce-8b98-e2927324b306">
2701 <ds:Transforms>
2702 <ds:Transform
2703   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
2704 <ds:Transform
2705   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2706 <InclusiveNamespaces
2707   PrefixList="#default saml samlp ds xsd xsi"
2708   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2709 </ds:Transform>
2710 </ds:Transforms>
2711 <ds:DigestMethod
2712   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
2713 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2714 </ds:Reference>
2715 </ds:SignedInfo>
2716 <ds:SignatureValue>
2717 x/GyPbzmfEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
2718 EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1yws7gFgsD01qjyen3CP+m3D
2719 w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
2720 <ds:KeyInfo>
2721 <ds:X509Data>
2722 <ds:X509Certificate>
2723 MIICyJCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
2724 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2725 F1VuaXZlcnNpdHkqb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
2726 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZSJ2ZXIqQ0Eg
2727 LS0gMjAwMjA3MDFBMB4XDTAyMjYyMjYyMjYyMjYyMjYyMjYyMjYyMjYyMjYyMjYy
2728 CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2729 Ym9yMQ4wDAYDVQQKEwVWVQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJ1ZXQyLmVh
2730 dTEuMCUGCSqGSIB3DQEJARYYcm9vdEBzaGlMS5pbNlcm5ldDIuZWRR1MIGfMA0G
2731 CSqGSIB3DQEBAAQAA4GNADCBiQKBgQDZSAb2sxvhaXnXVIVTtx8vuRay+x50z7GJj
2732 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZBl13R6+KYiE7x4XAWIrCP+
2733 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
2734 pmqOIFGTWQIDAQABox0wGzAMBgNVHRMBAf8EAJAAMAsGA1UdDwQEAwIFoDANBgkq
2735 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2736 qgi7lFV6MDkhhmTvTqBtjtmNk3No7v/dnP6Hr7wHxvCCRwubnmlfz6QZAv2FU78pLX
2737 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMnTREG8cCx3w/w==
2738 </ds:X509Certificate>
2739 </ds:X509Data>
2740 </ds:KeyInfo>
2741 </ds:Signature>
2742 <Status><StatusCode Value="samlp:Success"/></Status>
2743 <Assertion
2744   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
2745   IssueInstant="2003-04-17T00:46:02Z"
2746   Issuer="www.opensaml.org"
2747   MajorVersion="1"
2748   MinorVersion="1"
2749   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
2750   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2751   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2752 <Conditions
2753   NotBefore="2003-04-17T00:46:02Z"
2754   NotOnOrAfter="2003-04-17T00:51:02Z">
2755 <AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
2756 </AudienceRestrictionCondition></Conditions>
2757 <AuthenticationStatement
2758   AuthenticationInstant="2003-04-17T00:46:00Z"
2759   AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
2760 <Subject>
2761 <NameIdentifier
2762   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
2763 scott@example.org</NameIdentifier>
2764 <SubjectConfirmation>

```

```

2765 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
2766 </SubjectConfirmation></Subject>
2767 <SubjectLocality
2768   IPAddress="127.0.0.1"/>
2769 </AuthenticationStatement>
2770 <ds:Signature
2771   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2772 <ds:SignedInfo>
2773 <ds:CanonicalizationMethod
2774   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2775 <ds:SignatureMethod
2776   Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2777 <ds:Reference
2778   URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
2779 <ds:Transforms>
2780 <ds:Transform
2781   Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
2782 <ds:Transform
2783   Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2784 <InclusiveNamespaces
2785   PrefixList="#default saml samlp ds xsd xsi"
2786   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2787 </ds:Transform>
2788 </ds:Transforms>
2789 <ds:DigestMethod
2790   Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2791 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
2792 </ds:Reference>
2793 </ds:SignedInfo>
2794 <ds:SignatureValue>
2795 hq4zk+ZknjggCQgZm7ea8fI79gJESRy3E8LHDpYXWQIgzpkJN9CMLG8ENR4Nrw+n
2796 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJf0Th6qaAsNdeCyG86jmtp3TD
2797 MWuL/cBUj20tBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=</ds:SignatureValue>
2798 <ds:KeyInfo>
2799 <ds:X509Data>
2800 <ds:X509Certificate>
2801 MIICyJCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTAlVT
2802 MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2803 FlVuaXZlcnNpdHkgb2YgV2l2y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
2804 bmZvcmlhdGlubiBvZiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgd0Eg
2805 LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MV0xMDAwMjA3Mjc1MVowgYsxCzAJBgNVBAYTAlVTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2806 Ym9yMQ4wDAYDVQQKEwVvVQ0FJRDEcMBoGAlUEAxMTc2hpYjEuaW50ZXJvZXQyLmVka
2807 dTEuMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MV0xMDAwMjA3Mjc1MVowgYsxCzAJBgNVBAYTAlVTMREwDwYDVQQIEWhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2808 CSqGS1b3DQEBAQUAA4GNADCBiQKBgQDZSAb2svhAXnXVIVTx8vuRay+X50z7GJj
2809 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYIE7x4XAWIrcP+
2810 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRje
2811 pmqOIFGTWQIDAQABox0wGzAMBGNVHRMBAf8EAjAAMASGA1UdDwQEAWIFoDANBgkq
2812 hkiG9w0BAQQFAAOBqBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
2813 qgi71fV6MDkkmTvtqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfZ6QZAv2FU78pLX
2814 8I3bsbmRAUg4UP9hH6ABVq4KQKmknxulxQxLhpR1ylGpdioWMNTrEG8cC3w/w==
2815 </ds:X509Certificate>
2816 </ds:X509Data>
2817 </ds:KeyInfo>
2818 </ds:Signature></Assertion></Response>

```

2820

6 SAML Extensions

2821 The SAML schemas support extensibility. An example of an application that extends SAML assertions is
2822 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use
2823 the extensibility features in SAML to create extension schemas.

2824 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML
2825 elements can serve as the head element of a substitution group. However, SAML types are not defined
2826 as *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only
2827 elements and types that have been specifically designed to support extensibility.

2828 6.1 Assertion Schema Extension

2829 The SAML assertion schema is designed to permit separate processing of the assertion package and the
2830 statements it contains, if the extension mechanism is used for either part.

2831 The following elements are intended specifically for use as extension points in an extension schema;
2832 their types are set to *abstract*, and are thus usable only as the base of a derived type:

2833 • <Condition>

2834 • <Statement>

2835 • The following elements that are directly usable as part of SAML MAY be extended:

2836 • <AuthenticationStatement>

2837 • <AuthorizationDecisionStatement>

2838 • <AttributeStatement>

2839 • <AudienceRestrictionCondition>

2840 The following elements are defined to allow elements from arbitrary namespaces within them, which
2841 serves as a built-in extension point without requiring an extension schema:

2842 • <BaseIdentifier>

2843 • <SubjectConfirmationData>

2844 • <AttributeValue>

2845 • <Advice>

2846 • <AuthnContext>

2847 6.2 Protocol Schema Extension

2848 The following SAML protocol elements are intended specifically for use as extension points in an
2849 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived
2850 type:

2851 • <Query>

2852 • <SubjectQuery>

2853 The following elements that are directly usable as part of SAML MAY be extended:

- 2854 • <Request>
- 2855 • <AuthenticationQuery>
- 2856 • <AuthorizationDecisionQuery>
- 2857 • <AttributeQuery>
- 2858 • <Response>

2859 7 SAML-Defined Identifiers

2860 The following sections define URI-based identifiers for common authentication methods, resource access
2861 actions, and subject name identifier formats.

2862 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of
2863 the most current RFC that specifies the protocol is used. URI references created specifically for SAML
2864 have one of the following stems:

```
2865 urn:oasis:names:tc:SAML:1.0:  
2866 urn:oasis:names:tc:SAML:1.1:
```

2867 7.1 Authentication Method Identifiers

2868 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the
2869 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although
2870 both can refer to the same underlying mechanisms. An authentication statement with an
2871 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The
2872 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the
2873 authentication statement does not provide the means to perform that authentication, such as a password,
2874 key, or certificate.

2875 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,
2876 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML
2877 relying party to confirm that the request or message came from a system entity that corresponds to the
2878 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method
2879 that the relying party can use to do this in the future. This may or may not have any relationship to an
2880 authentication that was performed previously. Unlike the authentication method, the subject confirmation
2881 method may be accompanied by some piece of information, such as a certificate or key, that will allow
2882 the relying party to perform the necessary check.

2883 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML
2884 profiles specification [SAMLProf] for more information. Additional methods may be added by defining
2885 new profiles or by private agreement.

2886 The following identifiers refer to SAML-specified authentication methods.

2887 7.1.1 Password

2888 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

2889 The authentication was performed by means of a password.

2890 7.1.2 Kerberos

2891 **URI:** urn:ietf:rfc:1510

2892 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the
2893 Needham-Schroeder symmetric key authentication mechanism [Needham78].

2894 7.1.3 Secure Remote Password (SRP)

2895 **URI:** urn:ietf:rfc:2945

2896 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC
2897 2945].

2898 **7.1.4 Hardware Token**

2899 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

2900 The authentication was performed using some (unspecified) hardware token.

2901 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

2902 **URI:** urn:ietf:rfc:2246

2903 The authentication was performed using either the SSL or TLS protocol with certificate-based client
2904 authentication. TLS is described in [RFC 2246].

2905 **7.1.6 X.509 Public Key**

2906 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

2907 The authentication was performed by some (unspecified) mechanism on a key authenticated by means
2908 of an X.509 PKI [X.500][PKIX]. It may have been one of the mechanisms for which a more specific
2909 identifier has been defined below.

2910 **7.1.7 PGP Public Key**

2911 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

2912 The authentication was performed by some (unspecified) mechanism on a key authenticated by means
2913 of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific
2914 identifier has been defined below.

2915 **7.1.8 SPKI Public Key**

2916 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

2917 The authentication was performed by some (unspecified) mechanism on a key authenticated by means
2918 of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has
2919 been defined below.

2920 **7.1.9 XKMS Public Key**

2921 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

2922 The authentication was performed by some (unspecified) mechanism on a key authenticated by means
2923 of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific
2924 identifier has been defined below.

2925 **7.1.10 XML Digital Signature**

2926 **URI:** urn:ietf:rfc:3075

2927 The authentication was performed by means of an XML digital signature [RFC 3075].

2928 **7.1.11 Authentication Context**

2929 **URI:** urn:oasis:names:tc:SAML:2.0:am:authncontext

2930 The authentication method is described by the proximal <AuthnContext> element.

2931 **7.1.12 Unspecified**

2932 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

2933 The authentication was performed by an unspecified means.

2934 **7.2 Action Namespace Identifiers**

2935 The following identifiers MAY be used in the `Namespace` attribute of the <Action> element (see
2936 Section Element <Action>) to refer to common sets of actions to perform on resources.

2937 **7.2.1 Read/Write/Execute/Delete/Control**

2938 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

2939 Defined actions:

2940 Read Write Execute Delete Control

2941 These actions are interpreted as follows:

2942 Read

2943 The subject may read the resource.

2944 Write

2945 The subject may modify the resource.

2946 Execute

2947 The subject may execute the resource.

2948 Delete

2949 The subject may delete the resource.

2950 Control

2951 The subject may specify the access control policy for the resource.

2952 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

2953 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

2954 Defined actions:

2955 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

2956 The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner
2957 described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively
2958 specify that the stated permission is denied. Thus a subject described as being authorized to perform the
2959 action ~Read is affirmatively denied read permission.

2960 A SAML authority MUST NOT authorize both an action and its negated form.

2961 **7.2.3 Get/Head/Put/Post**

2962 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2963 Defined actions:

2964 GET HEAD PUT POST

2965 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform
2966 the GET action on a resource is authorized to retrieve it.

2967 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and
2968 POST actions to the write permission. The correspondence is not exact however since an HTTP GET
2969 operation may cause data to be modified and a POST operation may cause modification to a resource
2970 other than the one specified in the request. For this reason a separate Action URI reference specifier is
2971 provided.

2972 **7.2.4 UNIX File Permissions**

2973 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2974 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)
2975 notation.

2976 The action string is a four-digit numeric code:

2977 *extended user group world*

2978 Where the *extended* access permission has the value

2979 +2 if sgid is set

2980 +4 if suid is set

2981 The *user group* and *world* access permissions have the value

2982 +1 if execute permission is granted

2983 +2 if write permission is granted

2984 +4 if read permission is granted

2985 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read
2986 and execute; and world read.

2987 **7.3 NameIdentifier Format Identifiers**

2988 The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see
2989 Section) to refer to common formats for the content of the `<NameIdentifier>` element and the
2990 associated processing rules, if any.

2991 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of
2992 SAML.

2993 **7.3.1 Unspecified**

2994 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

2995 The interpretation of the content of the element is left to individual implementations.

2996 **7.3.2 Email Address**

2997 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

2998 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as
2999 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that
3000 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in
3001 parentheses) after it, and is not surrounded by "<" and ">".

3002 **7.3.3 X.509 Subject Name**

3003 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

3004 Indicates that the content of the element is in the form specified for the contents of the
3005 <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors
3006 should note that the XML Signature specification specifies encoding rules for X.509 subject names that
3007 differ from the rules given in IETF RFC 2253 [RFC 2253].

3008 **7.3.4 Windows Domain Qualified Name**

3009 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3010 Indicates that the content of the element is a Windows domain qualified name. A Windows domain
3011 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"
3012 separator MAY be omitted.

3013 **7.3.5 Kerberos Principal Name**

3014 **URI:** <urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos>

3015 [Indicates that the content of the element is in the form of a Kerberos principal name using the format](#)
3016 [name\[/instance\]@REALM. The syntax, format and characters allowed for the name, instance, and](#)
3017 [realm are described in \[RFC 1510\].](#)

3018 **7.3.6 Provider Identifier**

3019 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:provider

3020 Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a
3021 SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser
3022 profiles). Such an identifier can be used to make assertions about system entities that can issue SAML
3023 requests, responses, and assertions.

3024 **7.3.7 Federated Identifier**

3025 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:federated

3026 Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity
3027 federation between an identity provider and a service provider (or affiliation of service providers).
3028 Federated name identifiers generated by identity providers MUST be constructed using pseudo-random
3029 values that have no discernible correspondence with the subject's actual identifier (for example,
3030 username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's
3031 identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

3032 The element's content MUST contain the most recent identifier of the subject set by the identity provider.

3033 The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider
3034 participating in the identity federation. It MAY be omitted if the value can be derived from the context of
3035 the message containing the element, such as the issuer of an assertion.

3036 The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider
3037 or affiliation of providers participating in the identity federation. It MAY be omitted if the element is
3038 contained in a message intended only for consumption directly by the service provider, and the value
3039 would be the name of that service provider.

3040 The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject
3041 most recently set by the service provider or affiliation, if any. If no such identifier has been established,
3042 than the attribute MUST be omitted.

3043 Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear
3044 text with providers other than the providers that have established the identity federation. Furthermore,
3045 they MUST NOT appear in log files or similar locations without appropriate controls and protections.
3046 Deployments without such requirements are free to use other kinds of identifiers in their SAML
3047 exchanges.

3048 Note also that while federated identifiers are typically used to reflect an account linking relationship
3049 between a pair of providers, a service provider is not obligated to recognize or make use of the long term
3050 nature of the persistent identifier or establish such a link. Such a "one-sided" identity federation is not
3051 discernibly different and does not affect the behavior of the identity provider or any processing rules
3052 specific to federated identifiers in the protocols defined in this specification.

3053 **7.3.8 Transient Identifier**

3054 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

3055 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated
3056 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated
3057 in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of
3058 256 characters.

3059 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier
3060 represents a transient and temporary identity federation, as described in Section Federated Identifier. In
3061 such a case, they MAY be omitted in accordance with the rules specified in that section.

3062 **7.4 Attribute NameFormat Identifiers**

3063 The following identifiers MAY be used in the `NameFormat` attribute defined on the
3064 **AttributeDesignatorType** complex type (see Section x) to refer to the classification of the attribute name
3065 for purposes of interpreting the name.

3066 **7.4.1 Unspecified**

3067 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:unspecified

3068 The interpretation of the attribute name is left to individual implementations.

3069 **7.4.2 URI Reference**

3070 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:uri

3071 The attribute name follows the convention for URI references [RFC 2396], for example as used in
3072 XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is
3073 application-specific. See the Baseline Identities and Attributes specification [SAMLBaseAtts] for a full
3074 discussion of representing names of XACML, X.500, and LDAP attributes.

3075 **7.5 Attribute ValueType Identifiers**

3076 The following identifier MAY be used in the `ValueType` attribute defined on the
3077 **AttributeDesignatorType** complex type (see Section x) to refer to the URI-based datatype of the
3078 desired or supplied attribute.

3079 **7.5.1 Unspecified Value Type**

3080 **URI:** urn:oasis:names:tc:SAML:2.0:valuetype-format:unspecified

3081 Indicates that the datatype of the desired or supplied attribute is application-specific. Note that any
3082 `ValueType` setting (default or explicit) in an attribute query, including this setting, needs to be exactly
3083 matched (in addition to other exact matches) in order for an attribute to be returned.

8 References

3084

3085 The following works are cited in the body of this specification.

8.1 Normative References

3086

- 3087 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web
3088 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 3089 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
3090 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.
3091 Note that this specification normatively references [Schema2], listed below.
- 3092 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium
3093 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 3094 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World
3095 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 3096 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,
3097 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
3098 Consortium. Note that this specification normatively references [XMLEnc-XSD],
3099 listed below.
- 3100 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.
3101 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 3102 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January
3103 1999. <http://www.w3.org/TR/REC-xml-names>.
- 3104 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web
3105 Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this
3106 specification normatively references [XMLSig-XSD], listed below.
- 3107 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.
3108 [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
3109 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

8.2 Non-Normative References

3110

- 3111 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty
3112 Alliance Project, January 2003,
3113 [http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)
3114 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 3115 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of*
3116 *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December
3117 1978.
- 3118 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann. *PGP Message Exchange Formats*.
3119 IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 3120 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*
3121 *Certificate and CRL Profile*. IETF RFC 2459, January 1999.
3122 <http://www.ietf.org/rfc/rfc2459.txt>.
- 3123 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
3124 IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 3125 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
3126 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

3127 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January
3128 1999. <http://www.ietf.org/rfc/rfc2246.txt>.

3129 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*
3130 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.
3131 <http://www.ietf.org/rfc/rfc2253.txt>.

3132 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF
3133 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

3134 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.
3135 <http://www.ietf.org/rfc/rfc2630.txt>.

3136 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.
3137 <http://www.ietf.org/rfc/rfc2822.txt>.

3138 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,
3139 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

3140 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF
3141 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

3142 [SAMLAuthnCxt] J. Kemp. *Authentication Context for the OASIS Security Assertion Markup*
3143 *Language (SAML)*. OASIS, February 2004. Document ID sstc-saml-authn-
3144 context-2.0. <http://www.oasis-open.org/committees/security/>.

3145 [SAMLBaseAtts] J. Hughes and P. Mishra. *Baseline Identities and Attributes for the OASIS*
3146 *Security Assertion Markup Language (SAML)*. OASIS, March 2004. Document ID
3147 sstc-saml-baseline-atts-2.0. <http://www.oasis-open.org/committees/security/>.

3148 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*
3149 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.
3150 <http://www.oasis-open.org/committees/security/>.

3151 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*
3152 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.
3153 <http://www.oasis-open.org/committees/security/>.

3154 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*
3155 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3156 oasis-sstc-saml-conform-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)
3157 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3158 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*
3159 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)
3160 [open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3161 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*
3162 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1.
3163 HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)
3164 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"[http://www.oasis-](http://www.oasis-open.org/committees/security/)
[open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3165 [SAMLXSD] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID
3166 oasis-sstc-saml-schema-protocol-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)
3167 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3168 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*
3169 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID
3170 oasis-sstc-saml-sec-consider-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)
3171 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3172 [SAMLXSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID
3173 oasis-sstc-saml-schema-assertion-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)
3174 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3175 **[SPKI]** C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. *SPKI/*
3176 *Certificate Theory*. IETF RFC 2693, September 1999.
3177 <http://www.ietf.org/rfc/rfc2693.txt>.

3178 **[UNICODE-C]** M. Davis, M. J. Dürst. *Unicode Normalization Forms*. UNICODE Consortium,
3179 March 2001. <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>.

3180 **[W3C-CHAR]** M. J. Dürst. *Requirements for String Identity Matching and String Indexing*. World
3181 Wide Web Consortium, July 1998. <http://www.w3.org/TR/WD-charreq>.

3182 **[W3C-CharMod]** M. J. Dürst. *Character Model for the World Wide Web 1.0*. World Wide Web
3183 Consortium, April, 2002. <http://www.w3.org/TR/charmod/>.

3184 **[X.500]** ITU-T Recommendation X.501: Information Technology - Open Systems
3185 Interconnection - The Directory: Models. 1993.

3186 **[XACML]** eXtensible Access Control Markup Language (XACML), product of the OASIS
3187 XACML TC. [http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)
3188 [open.org/committees/tc_home.php?wg_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).

3189 **[XKMS]** W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J.
3190 Lapp. XML Key Management Specification (XKMS). W3C Note 30 March 2001.
3191 <http://www.w3.org/TR/xkms/>.

3192 **Appendix A. Acknowledgments**

3193 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
3194 Committee, whose voting members at the time of publication were:

- 3195 • @@

Appendix B. Revision History

Rev	Date	By Whom	What
01	20 Oct 2003	Eve Maler	Initial draft. Converted to OpenOffice. CORE-1 through CORE-4 . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed. http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf
02	4 Jan 2004	Eve Maler	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal (http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587) for work item W-2 , Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet). Fixed CORE-10 (the description of subelement occurrence in the <Evidence> element). http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf
03	24 Jan 2004	Scott Cantor	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars. 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/5180/sstc-saml-core-2.0-draft-03-diff.pdf
04	1 Feb 2004	Eve Maler	Made minor edits to new and existing material; changed new <AssertionRequest> element name to <AssertionIDRequest>; changed new <AssertionArtifact> and <NewIdentifier> 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 W-2 , 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. http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol (W-2), 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 (W-1); still unfinished. http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf
06	20 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added AssertionURIReference (W-19), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response (W-1). Implemented the freezing of authZ decision statement functionality (W-28b). http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf

Rev	Date	By Whom	What
07	7 Mar 2004	Scott Cantor, Eve Maler	<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 (W-2).</p>
http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf			
08	15 Mar 2004	Scott Cantor, Eve Maler	<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>
http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf			
09	8 Apr 2004	Eve Maler	<p>Minor cleanup, plus decisions from March-April 2004 F2F meeting: Moved Signature element up in Assertion contents. Clarified that DoNotCacheCondition has one-time-use semantics. Made NameFormat on the Attribute element clearly optional. Changed the default ValueType identifier name. Added the ability to put arbitrary attributes on the AttributeDesignator element. Removed Source on the Attribute element. Changed the content of Extensions in the Request element to ##other. Removed the restriction saying only federated identifiers could be replaced and set with the termination protocol. Changed Reason on the LogoutRequest element to be a URI reference. Made SessionIndex in the LogoutRequest element globally declared. Added bibliographic references to the new SAML specs.</p>
http://www.oasis-open.org/committees/download.php/6323/sstc-saml-core-2.0-draft-09-diff.pdf			
10	12 Apr 2004	Scott Cantor, Eve Maler	<p>Allowed assertions to be subjectless. Allowed Audience to reference a specific provider URI. Changed AuthnMethod and AuthnContext handling. Removed RelayState. Added AllowCreate on NameIDPolicy. Consolidated two protocols into the name identifier management protocol. Added a name identifier URI for Kerberos principals.</p>

3197

Appendix C. Notices

3199 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
3200 might be claimed to pertain to the implementation or use of the technology described in this document or
3201 the extent to which any license under such rights might or might not be available; neither does it
3202 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
3203 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
3204 made available for publication and any assurances of licenses to be made available, or the result of an
3205 attempt made to obtain a general license or permission for the use of such proprietary rights by
3206 implementors or users of this specification, can be obtained from the OASIS Executive Director.

3207 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
3208 or other proprietary rights which may cover technology that may be required to implement this
3209 specification. Please address the information to the OASIS Executive Director.

3210 **Copyright © OASIS Open 2004. All Rights Reserved.**

3211 This document and translations of it may be copied and furnished to others, and derivative works that
3212 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
3213 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright
3214 notice and this paragraph are included on all such copies and derivative works. However, this document
3215 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,
3216 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
3217 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required
3218 to translate it into languages other than English.

3219 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
3220 or assigns.

3221 This document and the information contained herein is provided on an "AS IS" basis and OASIS
3222 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
3223 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS
3224 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
3225 PURPOSE.