



1

2 **Liberty Protocols and Schema Specification**

3

Version 1.1

4

15 January 2003

5 **Document Description:** liberty-architecture-protocols-schema-v1.1

6 **Notice**

7 Copyright © 2002,2003 ActivCard; American Express Travel Related Services; America Online,
8 Inc.; Bank of America; Bell Canada; Catavault; Cingular Wireless; Cisco Systems, Inc.; Citigroup;
9 Communicator, Inc.; Consignia; Cyberun Corporation; Deloitte & Touche LLP; Earthlink, Inc.;
10 Electronic Data Systems, Inc.; Entrust, Inc.; Ericsson; Fidelity Investments; France Telecom;
11 Gemplus; General Motors; Hewlett-Packard Company; i2 Technologies, Inc.; Intuit Inc.;
12 MasterCard International; NEC Corporation; Netegrity; NeuStar; Nextel Communications; Nippon
13 Telegraph and Telephone Company; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.;
14 OneName Corporation; Openwave Systems Inc.; PricewaterhouseCoopers LLP; Register.com;
15 RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; SK Telecom; Sony
16 Corporation; Sun Microsystems, Inc.; United Airlines; VeriSign, Inc.; Visa International;
17 Vodafone Group Plc; Wave Systems;. All rights reserved.

18 This specification document has been prepared by Sponsors of the Liberty Alliance. Permission is
19 hereby granted to use the document solely for the purpose of implementing the Specification. No
20 rights are granted to prepare derivative works of this Specification. Entities seeking permission to
21 reproduce portions of this document for other uses must contact the Liberty Alliance to determine
22 whether an appropriate license for such use is available.

23 Implementation of certain elements of this Specification may require licenses under third party
24 intellectual property rights, including without limitation, patent rights. The Sponsors of and any
25 other contributors to the Specification are not, and shall not be held responsible in any manner, for
26 identifying or failing to identify any or all such third party intellectual property rights. **This**
27 **Specification is provided "AS IS", and no participant in the Liberty Alliance makes any**
28 **warranty of any kind, express or implied, including any implied warranties of**
29 **merchantability, non-infringement of third party intellectual property rights, and fitness for**
30 **a particular purpose.** Implementors of this Specification are advised to review the Liberty
31 Alliance Project's website (<http://www.projectliberty.org>) for information concerning any
32 Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management
33 Board.

34 Liberty Alliance Project
35 Licensing Administrator
36 c/o IEEE-ISTO
37 445 Hoes Lane
38 Piscataway, NJ 08855-1331, USA
39 info@projectliberty.org
40

41 **Editors**

42 John D. Beatty, Sun Microsystems, Inc.
43 John Kemp, IEEE-ISTO

44 **Contributors**

45
46 The following Liberty Alliance Project Sponsor companies contributed to the development
47 of this specification:
48

- | | |
|--|--|
| ActivCard | NEC Corporation |
| American Express Travel Related Services | Netegrity |
| America Online, Inc. | NeuStar |
| Bank of America | Nextel Communications |
| Bell Canada | Nippon Telegraph and Telephone Company |
| Catavault | Nokia Corporation |
| Cingular Wireless | Novell, Inc. |
| Cisco Systems, Inc. | NTT DoCoMo, Inc. |
| Citigroup | OneName Corporation |
| Communicator, Inc. | Openwave Systems Inc. |
| Consignia | PricewaterhouseCoopers LLP |
| Cyberun Corporation | Register.com |
| Deloitte & Touche LLP | RSA Security Inc |
| EarthLink, Inc. | Sabre Holdings Corporation |
| Electronic Data Systems, Inc. | SAP AG |
| Entrust, Inc. | SchlumbergerSema |
| Ericsson | SK Telecom |
| Fidelity Investments | Sony Corporation |
| France Telecom | Sun Microsystems, Inc. |
| Gemplus | United Airlines |
| General Motors | VeriSign, Inc. |
| Hewlett-Packard Company | Visa International |
| i2 Technologies, Inc. | Vodafone Group Plc |
| Intuit Inc. | Wave Systems |
| MasterCard International | |

49

50 **History**

Revision	Date	Log
00	20-Oct-02	CRs implemented: 1096; 1113; 1116; 1117; 1128; 1129; 1130; 1134; 1135; 1136; 1137; 1141; 1142; 1143; 1149; 1152; 1154; 1146
01	28-Oct-02	CRs implemented: 1161/1160, 1132 1161/1160: <ul style="list-style-type: none"> • Section 3.3.1.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.1.2 – updated example to match the above changes • Section 3.3.2.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.2.2 – updated example to match the above changes • Section 4.2 – added new metadata element to the service provider metadata to specify the preferred RegisterNameIdentifier protocol

		<p>1132:</p> <ul style="list-style-type: none"> • Section 3.2.1.1 – added AuthnContextComparisonType in text and changed schema. • Section 3.2.2.2 – updated example to match the above changes • Section 3.2.3 – updated text in processing rules to note this change.
02	29-Oct-02	<p>1161:</p> <ul style="list-style-type: none"> • Sections 3.3 -- 3.3.3 -- updated textual notes to reflect symmetry of RegisterNameIdentifier protocol between providers. • Section 4.2 -- added new metadata elements for RegisterNameIdentifierReturnURL and RegisterNameIdentifierProtocolProfile to the generic provider descriptor information. Removed them from Request/Response in 3.3.1/2 <p>1132:</p> <ul style="list-style-type: none"> • Made the AuthnContextComparisonType an enumerated attrib. <p>1158:</p> <ul style="list-style-type: none"> • Section 3.2.4.2 -- added paragraphs discussing the logout of the Principal after federation termination. • Section 3.2.3 -- added a processing rule stating that an AuthnResponse MUST have a samlp:StatusCode of lib:FederationDoesNotExist if the SP is not currently federated with the IDP. <p>General:</p> <ul style="list-style-type: none"> • Updated namespace calls to /2002/12 for 1.1 namespace(s).
03	30-Oct-02	<p>1153:</p> <ul style="list-style-type: none"> • Updated all examples to have UTC compliant date/time data with a Z at the end of each date/time. <p>1133:</p> <ul style="list-style-type: none"> • Section 3.3.3 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST be verified if present. • Section 3.2.3 added a line to processing rules, noting that an error should be returned if authentication fails. • Section 3.4 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST be verified if present. • Section 3.5.2.1 -- modified the processing rules to note that the signature SHOULD be verified, if present. <p>1146:</p> <ul style="list-style-type: none"> • Slightly revised language in lines 965-968 in line with John B's comment about tightening language. <p>1152:</p> <ul style="list-style-type: none"> • Section 4.2.1 - corrected example URIs for all profiles <p>General:</p> <ul style="list-style-type: none"> • Section 3.1.6 - added a reference for W3 XMLSchema dateTime • Fixed a spelling mistake 'occurred' line 654

		<ul style="list-style-type: none"> Added references to XML Signature and Canonicalization schema documents
04	31-Oct-02	<p>1158:</p> <ul style="list-style-type: none"> Section 3.4.2 – modified language to state that a provider MAY invalidate a user’s session as the result of federation termination. <p>1161:</p> <ul style="list-style-type: none"> Section 3.3.1.1 – added a new element <OldProvidedNameIdentifier> to the RegisterNameIdentifierRequest message. Updated examples, and schema to match Section 3.3 – added text to note that the <OldProvidedNameIdentifier> should contain the previous version of the <XXXProvidedNameIdentifier>
04	01-Nov-02	<p>1161:</p> <ul style="list-style-type: none"> Section 3.2.2.4 – added paragraphs to specify that saml:AuthenticationMethod should be populated with the Liberty context URI, when any AuthnContext is specified. Also noted that the SP must look to the Liberty AuthnContext (and ignore the saml attribute) if the saml:AuthenticationMethod is equal to the Liberty context URI. Section 3.3 – added note to the effect that implementations may want to take account of possible propagation delays in processing of name identifier changes. <p>1164:</p> <p>General:</p> <ul style="list-style-type: none"> Section 4.1 – removed a reference to profiles in the text Section 3.2.3 – slightly tightened language about processing of AuthnContextComparisonType rules All sections – removed CR comments <p>Examples:</p> <ul style="list-style-type: none"> Updated AuthnRequest example
05	05-Nov-02	General edits for spelling, grammar, formatting.
06	15-Nov-02	<p>1216: section 3.2.2.2 changed Assertion to AssertionType</p> <p>1215: changed SAML reference to point to SAML 1.0 document</p> <p>1214: corrected erroneous URIs lines 1186-1190</p> <p>1213: corrected description of SingleSignOnProtocolProfile</p> <p>1209: removed line 1010 regarding federation termination</p> <p>1208: removed duplicate signing requirement</p> <p>1207: saml:Success is now samlp:Success (line 908)</p> <p>1205: the name identifier that the service provider should use when communicating with the identity provider.</p> <p>1204: removed idp:Foo element from example</p> <p>1203: added ‘supported’ regarding lib:NoAvailableIDP (lines 689-690)</p> <p>1202: removed indentation</p> <p>1201: reworded language regarding AuthnContextComparisonType (622-633)</p> <p>1199: corrected URLs in examples</p>

		<p>1198: created reference from ReauthenticateOnOrAfter to prior section discussing it</p> <p>1197: removed incorrect indentation</p> <p>1196: corrected examples</p> <p>1194: corrected typo – placed instead of place (366)</p> <p>1192: reworded lines 283-285 to better deal with the specification of sub-second values.</p> <p>1191: added further requirement on UTC time + Z specified lines 281-282</p> <p>1190: reworded to be grammatically correct (256)</p> <p>1189: added the word ‘Individual’ line 252</p> <p>1188: Some profiles of the protocols contained in this specification may require a succinct 20-byte identifier. A provider MUST derive this by generating the SHA-1 hash of its URI-based identifier.</p> <p>1187: removed section number reference (226)</p> <p>1186: reworded lines 170-172.</p> <p>1185: corrected typos line 424, 428</p> <p>1184: corrected inconsistency in location of signing requirements</p> <p>1212: corrected differences between RegisterNameIdentifier and other protocol profiles</p> <p>1210: added RegisterNameIdentifierServiceURL, and corrected RegisterNameIdentifierServiceReturnURL</p>
07	18-Nov-02	General: reformatted, added new legal boilerplate text
08	22-Nov-02	<p>1200: added more explanation of the federation state mismatch</p> <p>1161/1206: added further explanation of the name registration protocol</p> <p>1211: removed ProviderSuccinctID</p> <p>1241: updated schema with corrections</p> <p>1240: added maxOccurs=”unbounded” to all affected protocol profiles</p> <p>1239: fixed RegisterNameIdentifierServiceReturnURL</p> <p>1236: added ProviderID to StatusResponseType</p> <p>1235: fixed XML Canon. reference</p> <p>1237: Created <LogoutResponse> message, inheriting StatusResponseType</p> <p>1234: Added id attribute to RegisterNameIdentifierResponse</p> <p>1250: reworded processing rule regarding federation of Principal’s identity</p> <p>1251, 1252: re-formatted sections describing processing rules for intermediaries</p> <p>General: edits for grammar, conciseness</p>
09	16-Dec-02	<p>1265: Fixed ResponseAbstractType for RNI Schema</p> <p>1267: Added RelayState to LogoutRequest schema</p> <p>1277: Added RelayState to FederationTermination protocol</p> <p>1272: Added notes about LECP making sure to return RelayState in errors, and for SP to relax requirement on clocks being within one minute, when working with LECP</p> <p>1278: Added note on use of SAML StatusCodes</p> <p>1279: Added id, ProviderID, RelayState to AuthnResponse to be used when no assertions are returned</p>

		1280: Noted that LogoutRequests should not be forwarded back to originating SP 1281: Attempted to clarify text on RNI protocol to clear up issues with IOP General: copy edits on formatting and grammar.
10	20-Dec-02	Updated references
1.1 Final	15-Jan-03	Fixed typos, added orange bar to footer, modified legal notice

51 **Table of Contents**

52	Introduction.....	9
53	1.1 Notation.....	9
54	1.2 Overview.....	10
55	2 Schema Declarations.....	10
56	2.1 Schema Header and Namespace Declarations.....	10
57	2.2 Type and Element Declarations.....	11
58	3 Protocols.....	11
59	3.1 General Requirements.....	11
60	3.1.1 XML Signature.....	11
61	3.1.2 Protocol and Assertion Versioning.....	11
62	3.1.3 Provider ID Uniqueness.....	11
63	3.1.4 Name Identifier Construction.....	12
64	3.1.5 Signature Verification.....	12
65	3.1.6 Security.....	12
66	3.1.7 Time Values.....	13
67	3.1.8 Time Synchronization.....	13
68	3.1.9 Response Status Codes.....	13
69	3.2 Single Sign-On and Federation Protocol.....	13
70	3.2.1 Request.....	14
71	3.2.2 Response.....	16
72	3.2.3 Processing Rules.....	19
73	3.2.4 Request Envelope.....	22
74	3.2.5 Response Envelope.....	25
75	3.3 Name Registration Protocol.....	25
76	3.3.1 Request.....	26
77	3.3.2 Response.....	27
78	3.3.3 Processing Rules.....	28
79	3.4 Federation Termination Notification Protocol.....	29
80	3.4.1 Message.....	29
81	3.4.2 Processing Rules.....	30
82	3.5 Single Logout Protocol.....	30
83	3.5.1 Request.....	30
84	3.5.2 Response.....	32
85	3.5.3 Processing Rules.....	32
86	4 Provider Metadata Schema.....	33
87	4.1 Generic Provider Descriptor.....	33
88	4.2 Service Provider Descriptor.....	34
89	4.2.1 Example.....	35
90	4.3 Identity Provider Descriptor.....	35
91	4.3.1 Example.....	36
92	5 Schema Definition.....	36
93	6 References.....	40

94 Introduction

95 This specification defines the abstract Liberty protocols for identity federation, single sign-on,
96 name registration, federation termination, and single logout. Several concrete bindings and profiles
97 of these protocols are defined in [[LibertyBindProf](#)].

98 1.1 Notation

99 This specification uses schema documents conforming to W3C XML Schema (see [[Schema1](#)]) and
100 normative text to describe the syntax and semantics of XML-encoded SAML assertions and
101 protocol messages. Note: Phrases and numbers in brackets [] refer to other documents; details of
102 these references can be found in Section 6 (at the end of this document).

103 The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,”
104 “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this
105 specification are to be interpreted as described in [[RFC2119](#)]: “they MUST only be used where it
106 is actually required for interoperoperation or to limit behavior which has potential for causing harm
107 (e.g., limiting retransmissions).”

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

112 Listings of schemas appear like this.

113
114 Listings of instance fragments appear like this.

115
116 The following namespaces are referred to in this document:

- 117 • The prefix `lib:` stands for the Liberty namespace
118 (<http://projectliberty.org/schemas/core/2002/12>). This namespace is the
119 default for instance fragments, type names, and element names in this document.
- 120 • The prefix `ac:` stands for the Liberty authentication context namespace
121 (<http://projectliberty.org/schemas/authctx/2002/05>)
- 122 • The prefix `saml:` stands for the SAML assertion namespace
123 (<urn:oasis:names:tc:SAML:1.0:assertion>).
- 124 • The prefix `samlp:` stands for the SAML protocol namespace
125 (<urn:oasis:names:tc:SAML:1.0:protocol>).
- 126 • The prefix `ds:` stands for the W3C XML signature namespace
127 (<http://www.w3.org/2000/09/xmldsig#>).
- 128 • The prefix `xsd:` stands for the W3C XML schema namespace
129 (<http://www.w3.org/2001/XMLSchema>). In schema listings, this is the default
130 namespace and no prefix is shown.
- 131 • The prefix `xsi:` stands for the W3C XML schema instance namespace
132 (<http://www.w3.org/2001/XMLSchema-instance>).

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

135 For readability, when an XML Schema type is specified to be `xsd:boolean`, this document
136 discusses the values as “true” and “false” rather than the “1” and “0” which are also legal
137 `xsd:boolean` values.

138 Definitions for Liberty-specific terms can be found in [[LibertyGloss](#)].

139 1.2 Overview

140 This specification defines a set of protocols that collectively provide a solution for identity
141 federation management, cross-domain authentication, and session management. This specification
142 also defines provider metadata schemas that may be used for making a priori arrangements
143 between providers.

144 The Liberty architecture contains three actors: Principal, identity provider, and service provider. A
145 Principal is an entity (for example, an end user) that has an identity provided by an identity
146 provider. A service provider provides services to the Principal.

147 Once the Principal is *authenticated* to the identity provider, the identity provider can provide an
148 authentication assertion to the Principal, who can present the assertion to the service provider. The
149 Principal is then also authenticated to the service provider if the service provider trusts the
150 assertion. An *identity federation* is said to exist between an identity provider and a service
151 provider when the service provider accepts authentication assertions regarding a particular
152 Principal from the identity provider. This specification defines a protocol where the identity of the
153 Principal can be *federated* between the identity provider and the service provider.

154 This specification relies on the SAML specification in [[SAMLCore](#)]. In SAML terminology, an
155 identity provider acts as an Asserting Party and an Authentication Authority, while a service
156 provider acts as a Relying Party.

157 2 Schema Declarations

158 This document specifies an XML schema for Liberty. The schema header along with namespace,
159 type, and element declarations are in 2.1 and 2.2.

160 2.1 Schema Header and Namespace Declarations

161 The following schema fragment defines the XML namespaces and other header information for the
162 Liberty schema:

```
163 <?xml version="1.0" encoding="UTF-8"?>  
164 <schema targetNamespace="http://projectliberty.org/schemas/core/2002/12"  
165 xmlns:lib="http://projectliberty.org/schemas/core/2002/12"  
166 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
167 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
168 xmlns:AC="http://www.projectliberty.org/schemas/authctx/2002/05"  
169 xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"  
170 attributeFormDefault="unqualified">  
171 <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation=" http://www.oasis-  
172 open.org/committees/security/docs/cs-sstc-schema-assertion-01.xsd"/>  
173 <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation=" http://www.oasis-  
174 open.org/committees/security/docs/cs-sstc-schema-protocol-01.xsd"/>  
175 <import namespace="http://www.w3.org/2000/09/xmldsig#"   
176 schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
```

```
177 <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05"  
178 schemaLocation="http://www.projectliberty.org/specs/liberty-architecture-authentication-context-  
179 v1.1.xsd"/>
```

180 2.2 Type and Element Declarations

181 Declarations for types and elements that are subsequently referred to in this document are as
182 follows:

```
183 <element name="ProviderID" type="anyURI"/>
```

184 3 Protocols

185 The Liberty protocol suite consists of the following protocols:

- 186 • Single Sign-On and Federation: The protocol by which identities are federated and by
187 which single sign-on occurs.
- 188 • Name Registration: The protocol by which a provider can register an alternative opaque
189 handle (or *name identifier*) for a Principal.
- 190 • Federation Termination Notification: The protocol by which a provider can notify another
191 provider than a particular identity federation has been terminated (also known as de-
192 federation).
- 193 • Single Logout: The protocol by which providers notify each other of logout events.

194 3.1 General Requirements

195 The following sections define a set of general requirements applicable to all protocols.

196 3.1.1 XML Signature

197 The XML signature specification calls out a general XML syntax for signing data with many
198 flexibilities and choices. All signed XML entities MUST adhere to the “XML Signature Profile”
199 constraints defined in [[SAMLCore](#)].

200 3.1.2 Protocol and Assertion Versioning

201 Version information appears in protocol messages and assertions defined in this specification. This
202 specification defines version 1.0 for the protocol messages and assertions. Version numbering of
203 assertions is independent of the version numbering of the protocol messages.

204 This specification follows the version numbering requirements, processing rules, and error
205 conditions specified in “SAML Versioning” in [[SAMLCore](#)].

206 3.1.3 Provider ID Uniqueness

207 All providers have a URI-based identifier. The provider’s URI-based identifier MUST be unique
208 within the scope of all providers with which it communicates. It is RECOMMENDED that a
209 provider use a URL with its own domain name for this identifier. The URI-based identifier MUST
210 NOT be more than 1024 characters in length.

211 Some profiles of the protocols contained in this specification may require a succinct 20-byte
212 identifier. A provider MUST derive any such identifier by generating the SHA-1 hash of its URI-
213 based identifier.

214 **3.1.4 Name Identifier Construction**

215 Principals are assigned name identifiers by identity providers and potentially by service providers.
216 When generated by the identity provider, a name identifier MUST be constructed using pseudo-
217 random values that have no discernible correspondence with the Principal's identifier (e.g.,
218 username) at the identity provider. The intent is to create a non-public pseudonym to prevent the
219 discovery of the Principal's identity or activities. Service providers SHOULD follow the same
220 construction rules. Name identifier values MUST NOT exceed a length of 256 characters.

221 **3.1.5 Signature Verification**

222 Processing rules for the protocols defined in this document commonly specify digital signature
223 verification. In these cases, it is not sufficient to only verify the signature of the signed object.
224 Verification of the `<ds:Signature>` element MUST be performed in accordance with the best
225 practices for the certification path technology in use. For example, when using X.509 v3 public
226 key certificates it is strongly RECOMMENDED that certification path validation be performed in
227 accordance to the PKIX Profile as specified in [[RFC3280](#)].

228 XML signatures SHOULD NOT be performed with any transforms other than:

- 229 • Enveloped Signature [[XMLDsig](#)]
- 230 • Exclusive XML Canonicalization [[XMLCanon](#)]

231 Receivers MUST NOT accept XML signatures created using other transforms without verification
232 that the transforms do not omit any part of the data to be signed from the signed byte stream.
233 Receivers MAY reject any messages with transforms other than the set specified above. Senders
234 MUST NOT send messages using other transforms without prior agreement as to their contents.

235 XML signatures in messages MUST use a proper URI fragment in the URI attribute of the
236 Reference element to identify the signed element. This URI fragment SHOULD reference the `id`
237 attribute of an element in the same document using an XPointer [[XPointer](#)] shortcut reference.

238 The signer MUST NOT assume that the signed element will be at the root of the document during
239 verification. It MUST be possible to validate the signature after adding or removing surrounding
240 context for the profile in use (for example, the SOAP envelope, or the `<samlp:Response>`
241 element). Implementers are encouraged to verify compliance with this requirement via empirical
242 testing.

243 The **SignedSAMLRequestType** is provided to allow SAML requests to be signed using these
244 guidelines. Specific usage of this type is shown in the relevant sections of [[LibertyBindProf](#)].

245 **3.1.6 Security**

246 Because this specification defines only abstract protocols and does not define specific protocol
247 profiles or the environment in which protocols will be deployed, most security requirements are
248 deferred to individual profiles. See [[LibertyBindProf](#)] for security considerations for the Liberty-
249 defined bindings and profiles. When a general security requirement can be stated for one of the
250 abstract protocols described in this specification, the requirement is stated in line with the specific
251 protocol.

252 **3.1.7 Time Values**

253 All Liberty time values have the type **dateTime**, which is built in to the W3C XML Schema
254 Datatypes specification [[Schema2](#)]. Liberty time values MUST be expressed in UTC form,
255 indicated by a “Z” immediately following the time portion of the value.

256 Liberty requesters and responders SHOULD NOT rely on other applications supporting time
257 resolution finer than seconds, as implementations MAY ignore fractional second components
258 specified in timestamp values. Implementations MUST NOT generate time instants that specify
259 leap seconds.

260 **3.1.8 Time Synchronization**

261 Providers SHOULD NOT assume that other providers have clocks that are synchronized closer
262 than one minute.

263 The Identity Provider SHOULD NOT include a `NotBefore` attribute on the `Conditions` element
264 of the assertion it generates which contains the time the assertion was generated.

265 The Identity Provider SHOULD NOT include a `NotOnOrAfter` attribute on the `Conditions`
266 element of the assertion it generates which is less than one minute later than the time when the
267 assertion was generated.

268 The Service Provider SHOULD NOT terminate the principal's session based solely on the
269 `NotOnOrAfter` attribute of the `Conditions` element of the assertion used to authenticate the
270 principal. If the assertion was valid when the principal was authenticated, the principal SHOULD
271 remain authenticated until one of the following occurs:

- 272 • `<LogoutRequest>` is received
- 273 • The user's session times out via normal means
- 274 • The `ReauthenticateOnOrAfter` time on the `<AuthenticationStatement>` used to
275 authenticate the principal, if any, is reached

276 **3.1.9 Response Status Codes**

277 All Liberty response messages use `<samlp:StatusCode>` elements to indicate the status of a
278 corresponding request. Responders MUST comply with the rules governing
279 `<samlp:StatusCode>` elements specified in [[SAMLCore](#)] regarding the use of nested second-
280 level response codes to provide specific information relating to particular errors. A number of
281 status codes are defined within the Liberty namespace for use with this specification.

282 **3.2 Single Sign-On and Federation Protocol**

283 The Single Sign-On and Federation Protocol defines a request and response protocol by which
284 single sign-on and identity federation occurs. The protocol works as follows:

- 285 1. A service provider issues an `<AuthnRequest>` request to an identity provider, instructing
286 the identity provider to provide an authentication assertion to the service provider.
287 Optionally, the service provider MAY request that the identity be federated.
- 288 2. The identity provider responds with either an `<AuthnResponse>` containing
289 authentication assertions to the service provider or an artifact that can be de-referenced into
290 an authentication assertion. Additionally, the identity provider potentially federates the

291 Principal's identity at the identity provider with the Principal's identity at the service
292 provider.

293 The resulting authentication statement in the assertion by the identity provider MAY contain a
294 `ReauthenticateOnOrAfter` attribute. If this attribute is included, the service provider MUST
295 send a new `<AuthnRequest>` for the Principal to the identity provider at the next point of
296 interaction with the Principal on or after the time specified by the `ReauthenticateOnOrAfter`
297 attribute. It is then up to the identity provider to authenticate the user.

298 Note: The Principal may already have an authenticated session with the identity provider, in which
299 case the identity provider would generate a new authentication assertion without any intervention by
300 the Principal.

301 3.2.1 Request

302 The service provider issues an `<AuthnRequest>` request to the identity provider. A set of
303 parameters is included in the request that allows the service provider to specify desired behavior at
304 the identity provider in processing the request. The service provider can control the following
305 identity provider behaviors:

- 306 • Prompt the Principal for credentials if the Principal is not presently authenticated.
- 307 • Prompt the Principal for credentials, even if the Principal is presently authenticated.
- 308 • Federate the Principal's identity at the identity provider with the Principal's identity at the
309 service provider.
- 310 • Use a specific protocol profile in responding to the request.
- 311 • Use a specific authentication context (for example, smartcard-based authentication vs.
312 username/password-based authentication).

313 Additionally, the service provider MAY include any desired state information in the request that
314 the identity provider should relay back to the service provider in the response.

315 The `<AuthnRequest>` message SHOULD be signed.

316 3.2.1.1 Element `<AuthnRequest>`

317 The `<AuthnRequest>` is defined as an extension of `samlp:RequestAbstractType`. The
318 `RequestID` attribute in `samlp:RequestAbstractType` has uniqueness requirements placed on it
319 by [[SAMLCore](#)], which require it to have the properties of a nonce.

320 The elements of the request are as follows:

321 `ProviderID` [Required]

322 The service provider's URI-based identifier.

323 `IsPassive` [Optional]

324 If "true," specifies that the identity provider MUST NOT interact with the Principal and
325 MUST NOT take control of the user interface from the service provider. If "false," the
326 identity provider MAY interact with the user and MAY temporarily take control of the user
327 interface for that purpose. If not specified, "true" is presumed.

328 `ForceAuthn` [Optional]

329 Controls whether the identity provider authenticates the Principal regardless of whether the
330 Principal is already authenticated. This element is specified only when <IsPassive> is
331 “false.” If <ForceAuthn> is “true,” specifies that the identity provider MUST always
332 authenticate the Principal, regardless of whether the Principal is presently authenticated. If
333 “false,” specifies that the identity provider MUST re-authenticate the user only if the
334 Principal is not presently authenticated. If not specified, “false” is presumed.

335 Federate [Optional]

336 Specifies that the service provider wishes to federate the Principal’s identity at the service
337 provider with the Principal’s identity at the identity provider. If the element is not
338 specified, it is presumed that the service provider does not wish to federate the identity.

339 ProtocolProfile [Optional]

340 The protocol profile that the service provider wishes to use for the response. If the element
341 is not specified, the default protocol profile is
342 <http://projectliberty.org/profiles/brws-art>, defined in [[LibertyBindProf](#)].

343 AuthnContext [Optional]

344 Information describing which authentication context the service provider desires the
345 identity provider to use in authenticating the Principal.

346 RelayState [Optional]

347 This contains state information that will be relayed back in the response. This data
348 SHOULD be integrity-protected by the request author and MAY have other protections
349 placed on it by the request author. An example of such protection is confidentiality.

350 id [Optional]

351 Identifier used to identify this element in the signature. See section 3.1.5, Signature
352 Verification for more information.

353 AuthnContextComparison [Optional]

354 If set to “exact”, then the identity provider is asked to match at least one of the specified
355 <AuthnContext> elements exactly. This can also be set to “minimum”, which asks that
356 the identity provider use a context that he feels is at least as good as any specified in the
357 <AuthnContext> or “better”, which means that the they can use any context better than
358 any that were supplied. If not specified, this is assumed to be “exact”.

359 The <AuthnContext> element has the following mutually exclusive elements:

360 AuthnContextClassRef [Optional]

361 The ordered set of authentication context class references the service provider desires the
362 identity provider to use in authenticating the Principal.

363 AuthnContextStatementRef [Optional]

364 The ordered set of exact authentication statements the service provider desires the identity
365 provider to use in authenticating the Principal.

366 The schema fragment defining the element and its type is as follows:

```
367 <element name="AuthnRequest" type="lib:AuthnRequestType"/>  
368 <complexType name="AuthnRequestType">  
369 <complexContent>
```

```

370     <extension base="samlp:RequestAbstractType">
371         <sequence>
372             <element ref="lib:ProviderID"/>
373             <element name="ForceAuthn" type="boolean" minOccurs="0"/>
374             <element name="IsPassive" type="boolean" minOccurs="0"/>
375             <element name="Federate" type="boolean" minOccurs="0"/>
376             <element ref="lib:ProtocolProfile" minOccurs="0"/>
377             <element ref="lib:AuthnContext" minOccurs="0"/>
378             <element ref="lib:RelayState" minOccurs="0"/>
379             <element name="AuthnContextComparison" type="lib:AuthnContextComparisonType"
380 minOccurs="0" maxOccurs="1"/>
381         </sequence>
382         <attribute name="id" type="ID" use="optional"/>
383     </extension>
384 </complexContent>
385 </complexType>
386 <simpleType name="AuthnContextComparisonType">
387     <restriction base="string">
388         <enumeration value="exact"/>
389         <enumeration value="minimum"/>
390         <enumeration value="better"/>
391     </restriction>
392 </simpleType>
393 <element name="RelayState"/>
394 <element name="ProtocolProfile" type="anyURI"/>
395 <element name="AuthnContext">
396     <complexType>
397         <choice>
398             <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>
399             <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>
400         </choice>
401     </complexType>
402 </element>

```

403 3.2.1.2 Example

```

404 <lib:AuthnRequest id="12345" RequestID="RPCUk211+GVz+t11LURp51oFvJXk" MajorVersion="1"
405 MinorVersion="0" IssueInstant="2001-12-17T21:42:4Z"
406 xmlns:lib="http://projectliberty.org/schemas/core/2002/12">
407     <ds:Signature> ... </ds:Signature>
408     <lib:ProviderID>http://ServiceProvider.com</lib:ProviderID>
409     <lib:ForceAuthn>>false</lib:ForceAuthn>
410     <lib:IsPassive>>false</lib:IsPassive>
411     <lib:Federate>>true</lib:Federate>
412     <lib:ProtocolProfile>http://projectliberty.org/profiles/brws-post</lib:ProtocolProfile>
413     <lib:AuthnContext>
414         <lib:AuthnContextClassRef>http://projectliberty.org/schemas/authctx/classes/Password-
415 ProtectedTransport</lib:AuthnContextClassRef>
416     </lib:AuthnContext>
417     <lib:RelayState>R01GODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</lib:RelayState>
418     <lib:AuthnContextComparison>exact</lib:AuthnContextComparison>
419 </lib:AuthnRequest>

```

420 3.2.2 Response

421 The response is an <AuthnResponse> element containing either a set of authentication assertions
422 or a set of artifacts the service provider can dereference into a set of authentication assertions.

423 All authentication assertions generated by an identity provider for a service provider MUST be of
424 type **AssertionType**. The <saml:Subject> element in any subject statement MUST be of type
425 **SubjectType**. If the service provider registered a name identifier for the Principal (see 3.3), the
426 <saml:NameIdentifier> element in the <saml:Subject> element MUST be the service
427 provider-provided name identifier for the Principal. Otherwise, <saml:NameIdentifier>
428 MUST be the most current name identifier supplied by the identity provider. The
429 <IDPProvidedNameIdentifier> MUST contain the most recent name identifier supplied by
430 the identity provider.

431 All authentication statements MUST be of type **AuthenticationStatementType**.

432 Identity providers MUST include a `<saml:AudienceRestrictionCondition>` element that
433 specifies the intended consumers of the assertion. The `<saml:Audience>` element MUST be set
434 to the intended recipient's ProviderID. The recipient MUST validate that it is the intended viewer
435 before using the assertion.

436 Identity providers MAY include a `SessionIndex` attribute in resulting authentication statements,
437 which is used to aid the identity provider in managing multiple sessions with the Principal. If the
438 identity provider includes this `SessionIndex` attribute, subsequent messages from the service
439 provider to the identity provider that are session-dependent MUST include this `SessionIndex`
440 attribute.

441 Each assertion in the `<AuthnResponse>` message MUST be individually signed by the identity
442 provider (that is, each assertion must contain a `Signature` element which signs only the assertion).
443 It is RECOMMENDED that the signature be omitted from the `<AuthnResponse>` itself, but
444 signing of the message is not forbidden.

445 3.2.2.1 Element `<AuthnResponse>`

446 The type `AuthnResponseType` is extended from `samlp:ResponseType`.

447 The response contains the following elements:

448 `ProviderID` [Required]

449 The identity provider's URI-based identifier.

450 `RelayState` [Optional]

451 This contains state information being relayed.

452 `id` [Optional]

453 Identifier used to identify this element in a signature. See section 3.1.5, Signature
454 Verification for more information.

455 The schema fragment is as follows:

```
456 <element name="AuthnResponse" type="lib:AuthnResponseType"/>  
457 <complexType name="AuthnResponseType">  
458   <complexContent>  
459     <extension base="samlp:ResponseType">  
460       <sequence>  
461         <element ref="lib:ProviderID"/>  
462         <element ref="lib:RelayState" minOccurs="0"/>  
463       </sequence>  
464       <attribute name="id" type="ID" use="optional"/>  
465     </extension>  
466   </complexContent>  
467 </complexType>
```

468 3.2.2.2 Element `<AssertionType>`

469 Assertions provided in an `<AuthnResponse>` element MUST be of type `AssertionType`, which
470 is an extension of `saml:AssertionType`, so that the `RequestID` attribute from the original
471 `<AuthnRequest>` is included in the `InResponseTo` attribute in the `<Assertion>` element.

472 This is done because it is not required that the `<AuthnResponse>` element itself be signed.
473 Instead, the individual `<Assertion>` elements contained must each be signed. The `id` attribute is
474 also included to facilitate such signatures (see section 3.1.5, Signature Verification).

475 The schema fragment is as follows:

```
476 <complexType name="AssertionType">
477   <complexContent>
478     <extension base="saml:AssertionType">
479       <attribute name="InResponseTo" type="saml:IDReferenceType"/>
480       <attribute name="id" type="ID" use="optional"/>
481     </extension>
482   </complexContent>
483 </complexType>
```

484 3.2.2.3 Type SubjectType

485 The type **SubjectType**, extended from **saml:SubjectType**, is used to include the
486 <IDPProvidedNameIdentifier> element in subject statements. The schema fragment is as
487 follows:

```
488 <complexType name="SubjectType">
489   <complexContent>
490     <extension base="saml:SubjectType">
491       <sequence>
492         <element ref="lib:IDPProvidedNameIdentifier"/>
493       </sequence>
494     </extension>
495   </complexContent>
496 </complexType>
```

497 3.2.2.4 Type AuthenticationStatementType

498 The type **AuthenticationStatementType** is an extension of **saml:AuthenticationStatementType**,
499 which allows for the following elements and attributes:

500 AuthnContext [Optional]

501 The context used by the identity provider in the authentication event that yielded this
502 statement. Contains either an authentication context statement or a reference to an
503 authentication context statement. Optionally contains a reference to an authentication
504 context class.

505 ReauthenticateOnOrAfter [Optional]

506 The time at, or after which the service provider reauthenticates the Principal with the
507 identity provider (as required in [Section 3.2](#) above).

508 SessionIndex [Optional]

509 Indexes the particular session between the Principal and the identity provider under which
510 this authentication statement is being issued. This value SHOULD be a small, positive
511 integer but may be any string of text. However, this value MUST NOT be a globally
512 unique value for the Principal's session at the Identity Provider.

513 When an <AuthnContext> element is specified, the **saml:AuthenticationMethod** attribute on
514 the <saml:AuthenticationStatement> MUST be
515 "http://projectliberty.org/schemas/authctx/2002/05".

516 When the Service Provider is processing a <saml:AuthenticationStatement> of type
517 **lib:AuthenticationStatementType** and the **saml:AuthenticationMethod** attribute is
518 "http://projectliberty.org/schemas/authctx/2002/05", the Service Provider MUST refer to the
519 <AuthnContext> element and ignore the **saml:AuthenticationMethod** attribute.

520

521 The schema fragment is as follows:

```
522 <complexType name="AuthenticationStatementType">
```

```

523     <complexContent>
524         <extension base="saml:AuthenticationStatementType">
525             <sequence>
526                 <element name="AuthnContext" minOccurs="0">
527                     <complexType>
528                         <sequence>
529                             <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
530                             <choice>
531                                 <element ref="ac:AuthenticationContextStatement"/>
532                                 <element name="AuthnContextStatementRef" type="anyURI"/>
533                             </choice>
534                         </sequence>
535                     </complexType>
536                 </element>
537             </sequence>
538             <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
539             <attribute name="SessionIndex" type="string" use="optional"/>
540         </extension>
541     </complexContent>
542 </complexType>

```

543 3.2.2.5 Example

```

544 <lib:AuthnResponse id="54321" ResponseID="9hhuujalbc744hGJn5Q9A5yvEIgS"
545 InResponseTo="Zon3WjJ2KL7j+bJu7MuIr4Pt2go5" MajorVersion="1" MinorVersion="0" IssueInstant="2002-
546 10-31T21:55:41Z">
547     <samlp:Status>
548         <samlp:StatusCode Value="samlp:Success"/>
549     </samlp:Status>
550     <saml:Assertion id="12345" MajorVersion="1" MinorVersion="0" AssertionID="e06e5a28-bc80-4ba6-
551 9ecb-712949db686e" Issuer="http://IdentityProvider.com" IssueInstant="2001-12-17T09:30:47Z"
552 InResponseTo="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" xsi:type="AssertionType">
553         <saml:Conditions NotBefore="2001-12-17T09:30:47Z" NotOnOrAfter="2001-12-17T09:35:47Z">
554             <saml:AudienceRestrictionCondition>
555                 <saml:Audience>http://ServiceProvider.com</saml:Audience>
556             </saml:AudienceRestrictionCondition>
557         </saml:Conditions>
558         <saml:AuthenticationStatement AuthenticationInstant="2001-12-17T09:30:47Z" SessionIndex="3"
559 ReauthenticateOnOrAfter="2001-12-17T11:30:47Z" xsi:type="AuthenticationStatementType"
560 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
561             <saml:Subject xsi:type="SubjectType">
562                 <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
563                 <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-
564 cc9e7db39e2b</IDPProvidedNameIdentifier>
565             </saml:Subject>
566         </saml:AuthenticationStatement>
567         <ds:Signature>...</ds:Signature>
568     </saml:Assertion>
569     <lib:ProviderID>http://IdentityProvider.com</lib:ProviderID>
570     <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
571 </AuthnResponse>

```

572 3.2.3 Processing Rules

573 When an identity provider receives an authentication request, it MUST process the request
574 according to the following rules:

- 575 • The <ProviderID> in the request MUST be the Provider ID of a known service provider
576 with which the identity provider has established a relationship. The <ProviderID>
577 MUST be resolvable to an assertion consumer service URL at the service provider that the
578 identity provider may use when returning the corresponding assertion reference.
- 579 • <ds:Signature>, if present, MUST be the signature of the service provider as specified
580 by the <ProviderID>.
- 581 • If <IsPassive> is "true," the identity provider MUST NOT interact with the Principal
582 and MUST NOT take control of the user interface (if applicable).

- 583 • The identity provider **MUST** attempt to authenticate the Principal if `<ForceAuthn>` is
584 “true,” regardless of whether the Principal is presently authenticated, unless
585 `<IsPassive>` is “true.”
- 586 • Failure to authenticate the Principal is indicated by a status code other than “Success.” For
587 failures, assertions **MUST NOT** appear in the `<AuthnResponse>`.
- 588 • The identity provider **MAY** federate the Principal’s identity at the service provider with
589 the user’s identity at the identity provider if `<Federate>` is “true” and the Principal has
590 consented for such an action to occur. The identity provider **MUST NOT** federate if
591 `<Federate>` is “false.” If `<Federate>` is “true” but the identity provider already has a
592 previous federation on record for the Principal’s identity at the service provider (such as
593 when a service provider previously issued a
594 `<FederationTerminationNotification>` which was not received by the identity
595 provider), the identity provider **SHOULD** treat the request as if `<Federate>` were
596 “false”.
- 597 • The identity provider **MUST** respond using the specified `<ProtocolProfile>`.
- 598 • If `<RelayState>` contains a value, the identity provider **MUST** include this value in
599 unmodified form in the `<RelayState>` element of the returned authentication assertion.
- 600 • The `InResponseTo` attribute in all generated `<Assertion>` elements in the
601 `<AuthnResponse>` element **MUST** be set to the value of the `RequestID` attribute in the
602 corresponding `<AuthnRequest>` element.
- 603 • If the Principal’s identity at the service provider is not federated with the identity provider,
604 and the `<Federate>` flag is not set, then the identity provider **MUST** return a second-
605 level `<samlp:StatusCode>` of `lib:FederationDoesNotExist` indicating that the
606 Principal’s identity at the service provider is not federated with this identity provider.
- 607 • If `<AuthnContextComparison>` is specified and set to “exact”, then the resulting
608 authentication statement in the assertion (if any) **MUST** be the exact match of at least one
609 of the authentication contexts specified.
- 610 • If `<AuthnContextComparison>` is specified and set to “minimum”, then the resulting
611 authentication statement in the assertion (if any) **MUST** be at least as strong (as deemed by
612 the identity provider) as one of the authentication contexts specified.
- 613 • If `<AuthnContextComparison>` is specified and set to “better”, then the resulting
614 authentication statement in the assertion (if any) **MUST** be stronger (as deemed by the
615 identity provider) than any specified in the supplied authentication contexts.
- 616 Additionally, if the `<AuthnContext>` element is specified, the identity provider **MUST**
617 authenticate the Principal according to the following rules:
- 618 • If one or more `<AuthnContextClassRef>` elements are included, then the resulting
619 authentication statement in the assertion (if any) **MUST** contain an authentication
620 statement that conforms to one of the specified classes. Additionally, the set of
621 `<AuthnContextClassRef>` elements **MUST** be evaluated as an ordered set, where the
622 first element is the most preferred authentication context class. If none of the specified
623 authentication context classes can be satisfied, the identity provider **MUST** not include an
624 authentication statement in the resulting assertion.

625 • If one or more `<AuthnContextStatementRef>` elements are included, then the
626 resulting authentication statement in the assertion (if any) **MUST** follow the rule specified
627 in the `<AuthnContextComparison>` element. If this requirement cannot be satisfied,
628 the identity provider **MUST** not include an authentication statement in the resulting
629 assertion.

630 If an identity is being federated, the identity provider **MUST** adhere to the following rules in
631 generating the name identifier:

- 632 • The name identifier **MUST** be unique across all Principals in the scope of that service
633 provider-identity provider relationship.
- 634 • The name identifier for the specific Principal **MUST** be unique across all service providers
635 with which an identity federation exists with the identity provider.

636 Failure to either authenticate the Principal and/or federate the identity is indicated by a status code
637 other than "Success." For failures, assertions **MUST NOT** appear in the `<AuthnResponse>`.

638 If the service provider attempts to federate a Principal's identity with an identity provider, but
639 another Principal's identity at the same service provider is already federated with the same identity
640 provider, it will receive the other Principal's established name identifier in the
641 `<AuthnResponse>`, rather than a new random one. The service provider **MUST** detect this error
642 and handle it appropriately without leaving either Principal's identity at the service provider in an
643 unusable state.

644 **3.2.3.1 Active Intermediaries**

645 In some profiles, an intermediary is active between the service provider's authentication request
646 and the identity provider's authentication response. Examples of an active intermediary include a
647 user agent or client proxy that implements the "Liberty-Enabled Client and Proxy Profile"
648 described in [[LibertyBindProf](#)].

649 NOTE: an active intermediary has the capability to return status codes to the service provider it
650 interacts with. For example, the intermediary may be unable to contact an identity provider
651 identified by the service provider, and the intermediary may return a status code to the service
652 provider indicating that an error occurred. Status codes **MUST** be conveyed within
653 `<AuthnResponse>` messages using the `<samlp:Status>` element, according to the rules
654 specified in [[SAMLCore](#)], utilizing second-level `<samlp:StatusCode>` elements. Specific
655 values are defined below. Service providers should also note that intermediaries are not providers,
656 and hence may not have clocks as accurately synchronized. This may invalidate the
657 `IssueInstant` attribute included in the `<AuthnResponse>` received by the service provider.

658 For all profiles specifying an active intermediary, the profile specification must:

- 659 • Specify whether the `<AuthnRequest>` element sent from the service provider to the
660 identity provider via the intermediary is wrapped in an `<AuthnRequestEnvelope>`. See
661 section 3.2.4.
- 662 • Specify whether the `<AuthnResponse>` element sent from the identity provider to the
663 service provider via the intermediary is wrapped in an `<AuthnResponseEnvelope>`. See
664 section 3.2.5.

665 3.2.3.1.1 Processing Rules for Active Intermediaries

666 For all profiles specifying an active intermediary, the intermediary MUST follow these processing
667 rules:

- 668 • If the profile specifies that the message sent from the service provider to the identity provider,
669 via the intermediary, is wrapped in an `<AuthnRequestEnvelope>`:
 - 670 • The intermediary MUST remove the enveloping `<AuthnRequestEnvelope>` before
671 forwarding the `<AuthnRequest>` element to the identity provider.
 - 672 • The intermediary MAY locally generate `<AuthnResponse>` elements and send them to the
673 service provider using the `<AssertionConsumerServiceURL>` contained within the
674 `<AuthnRequestEnvelope>`. Such `<AuthnResponse>` elements MUST NOT contain any
675 `<lib:Assertion>` elements. The `<AuthnResponse>` elements MUST have an
676 `InResponseTo` attribute set to the `RequestID` of the `<AuthnRequest>` that could not be
677 serviced. If the `<AuthnRequest>` contained a `<RelayState>` element, the
678 `<AuthnResponse>` MUST include a `<RelayState>` element with its value set to that
679 supplied in the `<AuthnRequest>`. Such responses MAY be generated as a result of local
680 errors on the intermediary, and should indicate the underlying reasons in the
681 `<samlp:Status>` element in the `<AuthnResponse>`. The following are error conditions for
682 which second-level `<samlp:StatusCode>` values are defined in section 3.2.3.1.2:
 - 683 • The identity provider cannot be reached
 - 684 • There is no identity provider in common between the intermediary and the service
685 provider
 - 686 • If the profile specifies that the message from the identity provider to the service provider, via
687 the intermediary, is wrapped in an `<AuthnResponseEnvelope>`:
 - 688 • The intermediary MUST remove the enveloping `<AuthnResponseEnvelope>` before
689 forwarding the `<AuthnResponse>` element to the service provider.
 - 690 • The intermediary MUST send `<AuthnResponse>` messages received from the identity
691 provider to the service provider using the `<AssertionConsumerServiceURL>`
692 contained within the `<AuthnResponseEnvelope>` sent by the identity provider.

693 3.2.3.1.2 Status Code Values for Error Conditions

694 If an error occurs in the processing at the intermediary, the following values are defined for use in
695 second-level `<samlp:StatusCode>` elements:

- 696 • `lib:NoAvailableIDP`: Used to indicate that none of the supported identity provider
697 URLs from the `<IDPList>` can be resolved or that none of the supported identity
698 providers are available.
- 699 • `lib:NoSupportedIDP`: Used to indicate that none of the identity providers are supported
700 by the intermediary.

701 3.2.4 Request Envelope

702 Some profiles MAY wrap the `<AuthnRequest>` element in an envelope. This envelope allows
703 for extra processing by an intermediary between the service provider and the identity provider. An
704 example of an intermediary is a user agent or proxy. Processing rules are given in section 3.2.3.1.1.

705 Note that the envelope is for consumption by the intermediary and is removed before the
706 enveloped <AuthnRequest> element is forwarded to the identity provider.

707 **3.2.4.1 Element <AuthnRequestEnvelope>**

708 The authentication request envelope contains the following elements:

709 AuthnRequest [Required]

710 The authentication request contained within the envelope.

711 ProviderID [Required]

712 The requestor's ProviderID.

713 ProviderName [Optional]

714 The human-readable name of the requestor.

715 AssertionConsumerServiceURL [Required]

716 A URL specifying where <AuthnResponse> elements, locally generated by an
717 intermediary, should be sent. See the processing rules for active intermediaries specified in
718 section 3.2.3.1.1.

719 IDPList [Optional]

720 A list of identity providers, from which, one may be chosen to service the authentication
721 request.

722 IsPassive [Optional]

723 If "true," specifies that any intermediary between the service provider and identity provider
724 MUST NOT interact with the Principal. If not specified, "true" is presumed.

725 The schema fragment is as follows:

```
726 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>  
727 <complexType name="AuthnRequestEnvelopeType">  
728   <complexContent>  
729     <extension base="lib:RequestEnvelopeType">  
730       <sequence>  
731         <element ref="lib:AuthnRequest"/>  
732         <element ref="lib:ProviderID"/>  
733         <element name="ProviderName" type="string" minOccurs="0"/>  
734         <element name="AssertionConsumerServiceURL" type="anyURI"/>  
735         <element ref="lib:IDPList" minOccurs="0"/>  
736         <element name="IsPassive" type="boolean" minOccurs="0"/>  
737       </sequence>  
738     </extension>  
739   </complexContent>  
740 </complexType>  
741 <complexType name="RequestEnvelopeType">  
742   <sequence>  
743     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>  
744   </sequence>  
745 </complexType>
```

746 **3.2.4.2 Element <IDPList>**

747 In the request envelope, some profiles may wish to allow the service provider to transport a list of
748 identity providers to the user agent. This specification provides a schema that profiles SHOULD
749 use for this purpose. The elements are as follows:

750 IDPList

751 The container element for an IDP List.

752 IDPEntries

753 Contains a list of identity provider entries.

754 IDPEntry

755 Describes an identity provider that the service provider supports.

756 ProviderID

757 The identity provider's ProviderID.

758 ProviderName

759 The identity provider's human-readable name.

760 Loc

761 The identity provider's URI, to which authentication requests may be sent.

762 GetComplete

763 If the identity provider list is not complete, this element is included with a URI that points
764 to where the complete list can be retrieved.

765 The schema fragment is as follows:

```
766 <element name="IDPList" type="lib:IDPListType"/>
767 <complexType name="IDPListType">
768   <sequence>
769     <element ref="lib:IDPEntries"/>
770     <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
771   </sequence>
772 </complexType>
773 <element name="IDPEntry">
774   <complexType>
775     <sequence>
776       <element ref="lib:ProviderID"/>
777       <element name="ProviderName" type="string" minOccurs="0"/>
778       <element name="Loc" type="anyURI"/>
779     </sequence>
780   </complexType>
781 </element>
782 <element name="IDPEntries">
783   <complexType>
784     <sequence>
785       <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
786     </sequence>
787   </complexType>
788 </element>
789 <element name="GetComplete" type="anyURI"/>
```

790 3.2.4.3 Example

```
791 <AuthnRequestEnvelope>
792   <AuthnRequest> ... </AuthnRequest>
793   <ProviderID>http://ServiceProvider.com</ProviderID>
794   <ProviderName>Service Provider X</ProviderName>
795   <AssertionConsumerServiceURL>http://ServiceProvider.com/lecp_assertion_consumer</AssertionConsu
796 merServiceURL>
797   <IDPList>
798     <IDPEntries>
799       <IDPEntry>
800         <ProviderID>http://IdentityProvider.com</ProviderID>
801         <ProviderName>Identity Provider X</ProviderName>
802         <Loc>http://www.IdentityProvider.com/liberty/sso</Loc>
803       </IDPEntry>
804     </IDPEntries>
805     <GetComplete>https://ServiceProvider.com/idplist?id=604be136-fe91-441e-afb8-f88748ae3b8b
806 </GetComplete>
```



```
807 </IDPList>  
808 <IsPassive>0</IsPassive>  
809 </AuthnRequestEnvelope>
```

810 3.2.5 Response Envelope

811 As with the <AuthnRequest> element, some profiles MAY wrap the <AuthnResponse>
812 element in an envelope. This envelope allows for extra processing by an intermediary (such as a
813 user agent or proxy) between the identity provider and the service provider. Applicable processing
814 rules are given in section 3.2.3.1.1. Note that the envelope is for consumption by the intermediary
815 and is removed prior to the forwarding of the enveloped <AuthnResponse> element to the
816 service provider.

817 3.2.5.1 Element <AuthnResponseEnvelope>

818 The authentication response envelope contains the following elements:

819 AuthnResponse [Required]

820 The enveloped authentication response.

821 AssertionConsumerServiceURL [Required]

822 The service provider's URL where the authentication response should be sent. This
823 element's value SHOULD be obtained from the element of the same name in the service
824 provider's Provider Metadata.

825 The schema fragment is as follows:

```
826 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>  
827 <complexType name="AuthnResponseEnvelopeType">  
828 <complexContent>  
829 <extension base="lib:ResponseEnvelopeType">  
830 <sequence>  
831 <element ref="lib:AuthnResponse"/>  
832 <element name="AssertionConsumerServiceURL" type="anyURI"/>  
833 </sequence>  
834 </extension>  
835 </complexContent>  
836 </complexType>  
837 <complexType name="ResponseEnvelopeType">  
838 <sequence>  
839 <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>  
840 </sequence>  
841 </complexType>
```

842 3.2.5.2 Example

```
843 <AuthnResponseEnvelope>  
844 <AuthnResponse> ... </AuthnResponse>  
845 <AssertionConsumerServiceURL>  
846 http://ServiceProvider.com/lecp_assertion_consumer  
847 </AssertionConsumerServiceURL>  
848 </AuthnResponseEnvelope>
```

849 3.3 Name Registration Protocol

850 During federation, the identity provider generates an opaque handle that serves as the initial name
851 identifier that both the service provider and the identity provider use in referring to the Principal
852 when communicating with each other. This name identifier is termed the
853 <IDPProvidedNameIdentifier>.

854 Subsequent to federation, the service provider MAY register a different opaque handle with the
855 identity provider. This opaque handle is termed the `<SPProvidedNameIdentifier>`. Until the
856 service provider registers a different name, the identity provider will use
857 `<IDPProvidedNameIdentifier>` to refer to the Principal when communicating with the
858 service provider.

859 After a service provider's name registration, the identity provider MUST use the
860 `<SPProvidedNameIdentifier>` for `<saml:NameIdentifier>` elements when
861 communicating to the service provider about the Principal. The service provider MUST use the
862 current (most recently supplied) `<IDPProvidedNameIdentifier>` for
863 `<saml:NameIdentifier>` elements when communicating to the identity provider about the
864 Principal.

865 Either the service provider or the identity provider MAY register a new name identifier for a
866 Principal with each other at any time following federation. The name identifiers specified by
867 providers SHOULD adhere to the following guidelines:

- 868 • The name identifier SHOULD be unique across the identity providers with which the
869 Principal's identity is federated.
- 870 • The name identifier SHOULD be unique within the group of name identifiers that have
871 been registered with the identity provider by this service provider.

872 **3.3.1 Request**

873 To register a `<SPProvidedNameIdentifier>` with an identity provider, the service provider
874 sends a `<RegisterNameIdentifierRequest>` message.

875 The same `<RegisterNameIdentifierRequest>` message may be sent by an identity provider,
876 seeking to change the `<IDPProvidedNameIdentifier>` stored by the service provider.

877 The `<RegisterNameIdentifierRequest>` message SHOULD be signed.

878 **3.3.1.1 Element `<RegisterNameIdentifierRequest>`**

879 The elements of the message are as follows:

880 `ProviderID` [Required]

881 The provider's identifier.

882 `IDPProvidedNameIdentifier` [Required]

883 The name identifier the service provider should use when communicating with the identity
884 provider.

885 `SPProvidedNameIdentifier` [Required]

886 The name identifier the identity provider should use when communicating to the service
887 provider.

888 `OldProvidedNameIdentifier` [Required]

889 In the case of either provider choosing to request a change of provided name identifiers,
890 this element holds the previous version. For a service provider making their first name
891 change following federation, the `<OldProvidedNameIdentifier>` will contain the

892 current <IDPProvidedNameIdentifier>. The <SPPProvidedNameIdentifier>
893 will contain the new name that the service provider wishes the identity provider to use.

894 id [Optional]

895 Identifier used to identify this element in the signature. See section 3.1.5, Signature
896 Verification.

897 RelayState [Optional]

898 This contains state information that will be relayed back in the response. This data
899 SHOULD be integrity-protected by the request author and MAY have other protections
900 placed on it by the request author. An example of such protection is confidentiality.

901 The schema fragment is as follows:

```
902     <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
903     <complexType name="RegisterNameIdentifierRequestType">
904       <complexContent>
905         <extension base="sampl:RequestAbstractType">
906           <sequence>
907             <element ref="lib:ProviderID"/>
908             <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
909             <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
910             <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
911             <element ref="lib:RelayState" minOccurs="0"/>
912           </sequence>
913           <attribute name="id" type="ID" use="optional"/>
914         </extension>
915       </complexContent>
916     </complexType>
917     <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
918     <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
919     <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
920
```

921 3.3.1.2 Example

```
922 <RegisterNameIdentifierRequest id="12345" RequestID="eb20e77f-d982-44f9-936e-dd135bf437d4"
923 MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
924   <ds:Signature>...</ds:Signature>
925   <ProviderID>http://ServiceProvider.com</ProviderID>
926   <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</IDPProvidedNameIdentifier>
927   <SPPProvidedNameIdentifier>e958019a</SPPProvidedNameIdentifier>
928   <OldProvidedNameIdentifier>e895014a</OldProvidedNameIdentifier>
929   <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
930 </RegisterNameIdentifierRequest>
```

931 3.3.2 Response

932 The responding provider MUST respond with <RegisterNameIdentifierResponse>, which
933 is of type **StatusResponseType**. **StatusResponseType** is an extension of **sampl:ResponseType**
934 and a <sampl:Status> element and a <RelayState> may exist in the body.

935 This message SHOULD be signed.

936 3.3.2.1 Element <RegisterNameIdentifierResponse>

937 The elements of the message are as follows:

938 ProviderID [Required]

939 The provider's unique identifier.

940 Status [Required]

941 The status of the request processing.

942 id [Optional]

943 Identifier used to identify this element in the signature. See section 3.1.5, Signature
944 Verification.

945 RelayState [Optional]

946 This element contains state information that will be relayed back in the response, if it has
947 been supplied in the request.

948 The schema fragment is as follows:

```
949 <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
950 <complexType name="StatusResponseType">
951   <complexContent>
952     <extension base="samlp:ResponseAbstractType">
953       <sequence>
954         <element ref="lib:ProviderID"/>
955         <element ref="samlp:Status"/>
956         <element ref="lib:RelayState" minOccurs="0"/>
957       </sequence>
958       <attribute name="id" type="ID" use="optional"/>
959     </extension>
960   </complexContent>
961 </complexType>
```

962 3.3.2.2 Example

```
963 <RegisterNameIdentifierResponse id="12345" ResponseID="74ffec0f-1165-4fa3-b088-3dd2c2388b91"
964 InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0"
965 IssueInstant="2001-12-17T09:30:47Z" Recipient="http://ServiceProvider.com">
966   <ds:Signature>...</ds:Signature>
967   <ProviderID>http://ServiceProvider.com</ProviderID>
968   <samlp:Status>
969     <samlp:StatusCode Value="samlp:Success"/>
970   </samlp:Status>
971   <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
972 </RegisterNameIdentifierResponse>
```

973 3.3.3 Processing Rules

974 The recipient MUST validate any signature present on the message. To be considered valid, the
975 signature provided MUST be the signature of the <ProviderID> contained in the message.
976 If the request includes an <IDPProvidedNameIdentifier> for which no federation exists
977 between the service provider and the identity provider, the provider MUST respond with a
978 <samlp:Status> element containing a second-level <samlp:StatusCode> of
979 lib:FederationDoesNotExist. Otherwise, the identity provider MUST use
980 <SPProvidedNameIdentifier> when subsequently communicating to the service provider
981 regarding this Principal.

982 Either provider MAY choose to change their provided name identifier. In this case, the
983 <OldProvidedNameIdentifier> should contain the previous version of their name identifier.
984 When a service provider chooses to change their provided name identifier, the
985 <OldProvidedNameIdentifier> should contain the current
986 <SPProvidedNameIdentifier>. Note that when they first change their name, this will be
987 equal to the <IDPProvidedNameIdentifier>. Similarly, when an identity provider wishes to
988 change their provided name identifier, they will move the previous version to the
989 <OldProvidedNameIdentifier> when sending this message.

990 Changes to these identifiers may take a potentially significant amount of time to propagate through

991 the systems at both the sender and the receiver. Implementations MAY wish to allow each party to
992 accept either identifier for some period of time following the successful completion of a name
993 identifier change. Not doing so could result in the inability of the Principal to access resources.

994 If <RelayState> contains a value, the recipient MUST include this value in unmodified form in
995 the <RelayState> element of the response.

996 **3.4 Federation Termination Notification Protocol**

997 When the Principal terminates an identity federation between a service provider and an identity
998 provider from the service provider, the service provider MUST send a
999 <FederationTerminationNotification> message to the identity provider. Semantically,
1000 the service provider is stating that it will no longer accept authentication assertions from the
1001 identity provider for the specified Principal.

1002 Likewise, when the Principal terminates an identity federation from the identity provider, the
1003 identity provider MUST send a <FederationTerminationNotification> message to the
1004 service provider. Semantically, the identity provider is stating that it will no longer provide
1005 authentication assertions to the service provider for the specified Principal.

1006 This notification message is a one-way asynchronous message. Reasonable, best-effort delivery
1007 MUST be employed by all providers sending this message.

1008 **3.4.1 Message**

1009 The provider sends a <FederationTerminationNotification> to the provider with which it
1010 is terminating a federation.

1011 The <FederationTerminationNotification> message SHOULD be signed.

1012 **3.4.1.1 Element <FederationTerminationNotification>**

1013 The elements are as follows:

1014 `ProviderID` [Required]

1015 The identifier of the provider that is sending this message.

1016 `NameIdentifier` [Required]

1017 The name identifier of the Principal terminating federation. This name identifier MUST be
1018 equal to the <saml:NameIdentifier> element (and its included attributes) agreed upon
1019 earlier between the two communicating providers.

1020 `id` [Optional]

1021 Identifier used to identify this element in the signature. See section 3.1.5, Signature
1022 Verification.

1023 `RelayState` [Optional]

1024 This element contains state information that may be relayed back.

1025 The schema fragment is as follows:

```
1026 <element name="FederationTerminationNotification"  
1027 type="lib:FederationTerminationNotificationType"/>  
1028 <complexType name="FederationTerminationNotificationType">  
1029 <complexContent>
```

1030
1031
1032
1033
1034
1035
1036
1037
1038
1039

```
<extension base="sampl:RequestAbstractType">
  <sequence>
    <element ref="lib:ProviderID"/>
    <element ref="saml:NameIdentifier"/>
    <element ref="lib:RelayState" minOccurs="0"/>
  </sequence>
  <attribute name="id" type="ID" use="optional"/>
</extension>
</complexContent>
</complexType>
```

1040

3.4.1.2 Example

1041
1042
1043
1044
1045
1046
1047

```
<FederationTerminationNotification id="12345" RequestID="9ec2-eb65-4bce-ab8f-4becdf229815"
MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
  <ds:Signature>...</ds:Signature>
  <ProviderID>http://IdentityProvider.com</ProviderID>
  <saml:NameIdentifier>e958019a</saml:NameIdentifier>
  <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
</FederationTerminationNotification>
```

1048

3.4.2 Processing Rules

1049
1050
1051

The receiving provider MUST validate any signature present on the message. The signature on the message MUST be the signature of the <ProviderID> contained in the message. If the signature is not valid, the provider MUST ignore the message.

1052
1053
1054
1055

If a provider receives a federation termination notification message that refers to a federation that does not exist from the perspective of the provider, the provider MUST ignore the message. Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been terminated.

1056
1057

A provider MAY choose to invalidate the session of a user for whom federation has been terminated.

1058

3.5 Single Logout Protocol

1059
1060
1061
1062

The Single Logout Protocol provides a message exchange protocol by which all sessions authenticated by a particular identity provider are near-simultaneously terminated. The Single Logout Protocol is used either when a Principal logs out at a service provider or when the Principal logs out at an identity provider.

1063
1064
1065

When the Principal invokes the single logout process at a service provider, the service provider MUST send a <LogoutRequest> message to the identity provider that provided the authentication service for the session.

1066
1067
1068
1069
1070

When either the Principal invokes a logout at the identity provider or a service provider sends a logout request to the identity provider specifying that Principal, the identity provider MUST send a <LogoutRequest> message to each service provider to which it provided authentication assertions in the current session with the Principal, with the exception of the service provider that sent the <LogoutRequest> message to the Identity Provider.

1071

3.5.1 Request

1072
1073
1074

The <LogoutRequest> message indicates to the message receiver that a Principal's session was terminated. The message includes an optional <SessionIndex> element that MUST be specified if and only if the authentication statement in the assertion that the service provider used in

1075 establishing the session with the Principal contained a `SessionIndex` attribute. This message
1076 SHOULD be signed.

1077 3.5.1.1 Element <LogoutRequest>

1078 `NameIdentifier` [Required]

1079 The name identifier of the Principal that logged out. This name identifier MUST be equal
1080 to the <saml:NameIdentifier> element (including the equality of contained attributes)
1081 agreed upon between the two communicating providers.

1082 `ProviderID` [Required]

1083 The identifier of the provider that is making the request.

1084 `SessionIndex` [Optional]

1085 The session index specified in the authentication statement of the assertion used to
1086 establish the session being terminated. If a <SessionIndex> element was present in the
1087 authentication statement, an identical <SessionIndex> MUST be present in the
1088 <LogoutRequest>. If no <SessionIndex> element was present in the authentication
1089 statement, the <SessionIndex> MUST be omitted from the <LogoutRequest>.

1090 `id` [Optional]

1091 Identifier used to identify this element in the signature. See section 3.1.5, Signature
1092 Verification.

1093 `RelayState` [Optional]

1094 This may contain state information that will be relayed back in the response. This data
1095 SHOULD be integrity-protected by the request author and MAY have other protections
1096 placed on it by the request author. An example of such protection is confidentiality.

1097 The schema fragment is as follows:

```
1098 <element name="LogoutRequest" type="lib:LogoutRequestType"/>
1099 <complexType name="LogoutRequestType">
1100   <complexContent>
1101     <extension base="samlp:RequestAbstractType">
1102       <sequence>
1103         <element ref="lib:ProviderID"/>
1104         <element ref="saml:NameIdentifier"/>
1105         <element name="SessionIndex" type="string" minOccurs="0"/>
1106         <element ref="lib:RelayState" minOccurs="0"/>
1107       </sequence>
1108       <attribute name="id" type="ID" use="optional"/>
1109     </extension>
1110   </complexContent>
1111 </complexType>
```

1112 3.5.1.2 Example

```
1113 <LogoutRequest id="12345" RequestID="47693d03-7c33-4d65-931f-ddeb19fa6a73" MajorVersion="1"
1114 MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
1115   <ds:Signature>...</ds:Signature>
1116   <ProviderID>http://ServiceProvider.com</ProviderID>
1117   <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
1118   <SessionIndex>3</SessionIndex>
1119   <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
1120 </LogoutRequest>
```

1121

1122 **3.5.2 Response**

1123 The responding provider MUST return a <LogoutResponse> message, which is of type
1124 **StatusResponseType**.

1125 This message SHOULD be signed.

1126 **3.5.2.1 Element <LogoutResponse>**

1127 The elements of the message are as follows:

1128 ProviderID [Required]

1129 The identifier of the provider responding.

1130 Status [Required]

1131 A status code that indicates the result of the request.

1132 id [Optional]

1133 Identifier used to identify this element in the signature. See section 3.1.5, Signature
1134 Verification.

1135 RelayState [Optional]

1136 This contains state information that may have appeared in the request, and is being relayed
1137 back to the sender.

1138 The schema fragment is as follows:

```
1139 <element name="LogoutResponse" type="lib:StatusResponseType"/>
```

1140 **3.5.2.2 Example**

```
1141 <LogoutResponse id="12345" ResponseID="74ffec0f-1165-4fa3-b088-3dd2c2388b91"  
1142 InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0"  
1143 IssueInstant="2001-12-17T09:30:47Z" Recipient="http://ServiceProvider.com">  
1144   <ds:Signature>...</ds:Signature>  
1145   <ProviderID>http://IdentityProvider.com</ProviderID>  
1146   <samlp:Status>  
1147     <samlp:StatusCode Value="samlp:Success"/>  
1148   </samlp:Status>  
1149   <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>  
1150 </LogoutResponse>
```

1151 **3.5.3 Processing Rules**

1152 If <RelayState> contains a value, the recipient MUST include this value in unmodified form in
1153 the <RelayState> element of the response.

1154 Other unique processing rules apply based on whether the message receiver is an identity provider
1155 or a service provider.

1156 **3.5.3.1 Identity Provider Processing Rules**

1157 When an identity provider receives the <LogoutRequest> message, the identity provider MUST
1158 validate that any signature present on the message is the signature of a service provider to which
1159 the identity provider provided an authentication assertion for the current session. If that holds, the
1160 identity provider SHOULD do the following:

- 1161
- Send a `<LogoutRequest>` message to each service provider for which the identity provider provided authentication assertions in the current session, *other than the originator of the* `<LogoutRequest>`. If an error occurs during this further processing of the logout (for example, relying service providers may not all implement the Single Logout profile used by the requesting service provider), the identity provider MUST respond to the original requestor with a `<LogoutResponse>` message, indicating the status of the logout request. The value `<lib:UnsupportedProfile>` is provided for a second-level `<samlp:StatusCode>`, indicating that a service provider should retry the `<LogoutRequest>` using a different profile.
- 1162
- 1163
- 1164
- 1165
- 1166
- 1167
- 1168
- 1169
- Terminate the Principal's current session as specified by the `<saml:NameIdentifier>` element.
- 1170
- 1171

1172 3.5.3.2 Service Provider Processing Rules

1173 When the service provider receives the `<LogoutRequest>` message, the service provider MUST
1174 validate the identity provider's signature contained in the `<ds:Signature>` element. If the
1175 signature is that of the identity provider that provided the authentication for the Principal's current
1176 session, the service provider MUST invalidate the Principal's session referred to in the
1177 `<saml:NameIdentifier>` element.

1178 4 Provider Metadata Schema

1179 For providers to communicate with each other, they must a priori have obtained metadata
1180 regarding each other. These provider metadata include items such as X.509 certificates and service
1181 endpoints. This specification defines metadata schemas for identity providers and service providers
1182 that may be used for provider metadata exchange. However, provider metadata exchange protocols
1183 are outside the scope of this specification.

1184 4.1 Generic Provider Descriptor

1185 Certain provider metadata are generic to both service providers and identity providers. The
1186 complex type **ProviderDescriptorType** contains the following elements:

1187 `ProviderID` [Required]

1188 The provider's URI-based identifier.

1189 `KeyInfo` [Optional]

1190 The provider's public key.

1191 `SoapEndpoint` [Optional]

1192 The provider's SOAP endpoint URI.

1193 `SingleLogoutServiceURL` [Optional]

1194 The URL used for user-agent-based Single Logout Protocol profiles.

1195 `SingleLogoutServiceReturnURL` [Optional]

1196 The URL to which the provider redirects at the end of user-agent-based Single Logout
1197 Protocol profiles.

- 1198 FederationTerminationServiceURL [Optional]
1199 The URL used for user-agent-based Federation Termination Notification Protocol profiles.
- 1200 FederationTerminationServiceReturnURL [Optional]
1201 The URL to which the provider redirects at the end of user-agent-based Federation
1202 Termination Notification Protocol profiles.
- 1203 FederationTerminationNotificationProtocolProfile [Optional]
1204 The Federation Termination Notification Protocol profiles supported by the provider. Each
1205 value of the element MUST contain a valid Federation Termination Notification Protocol
1206 profile identification URI. The absence of this element SHALL mean that provider does
1207 not support any profile of the Federation Termination Notification Protocol.
- 1208 SingleLogoutProtocolProfile [Optional]
1209 The Single Logout Protocol profiles supported by the provider. Each element MUST
1210 contain a valid Single Logout Protocol profile identification URI. The absence of this
1211 element SHALL mean that the provider does not support any profile of the Single Logout
1212 Protocol.
- 1213 RegisterNameIdentifierProtocolProfile [Optional]
1214 The provider's preferred Register Name Identifier Protocol profile, which should be used
1215 by other providers when registering a new identifier. Each element MUST contain a valid
1216 Register Name Identifier Protocol profile identification URI. The absence of this element
1217 SHALL mean that the provider does not support any profile of the Register Name Identifier
1218 Protocol.
- 1219 RegisterNameIdentifierServiceURL [Optional]
1220 The URL used for user-agent-based Register Name Identifier Protocol profiles.
- 1221 RegisterNameIdentifierServiceReturnURL [Optional]
1222 The provider's redirecting URL for use after HTTP name registration has taken place.

1223 The schema fragment is as follows:

```
1224 <complexType name="ProviderDescriptorType">
1225   <sequence>
1226     <element name="ProviderID" type="anyURI"/>
1227     <element ref="ds:KeyInfo" minOccurs="0"/>
1228     <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
1229     <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
1230     <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
1231     <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
1232     <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
1233     <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"
1234 minOccurs="0" maxOccurs="unbounded"/>
1235     <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"
1236 maxOccurs="unbounded"/>
1237     <element name="RegisterNameIdentifierProtocolProfile" type="anyURI" minOccurs="0"
1238 maxOccurs="unbounded"/>
1239     <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs="0"/>
1240     <element name="RegisterNameIdentifierServiceReturnURL" type="anyURI" minOccurs="0"/>
1241   </sequence>
1242 </complexType>
```

1243 4.2 Service Provider Descriptor

1244 The additional service provider-specific metadata are as follows:

1245 AssertionConsumerServiceURL [Required]

1246 The service provider's URL for consuming assertions from identity providers.

1247 AuthnRequestsSigned [Required]

1248 Specifies whether the service provider will always sign authentication requests it sends to
1249 the identity provider.

1250 The schema fragment is as follows:

```
1251 <element name="SPDescriptor" type="lib:SPDescriptorType"/>
1252 <complexType name="SPDescriptorType">
1253   <complexContent>
1254     <extension base="lib:ProviderDescriptorType">
1255       <sequence>
1256         <element name="AssertionConsumerServiceURL" type="anyURI"/>
1257         <element name="AuthnRequestsSigned" type="boolean"/>
1258       </sequence>
1259     </extension>
1260   </complexContent>
1261 </complexType>
```

1262 4.2.1 Example

```
1263 <SPDescriptor>
1264   <ProviderID>http://ServiceProvider.com</ProviderID>
1265   <ds:KeyInfo>...</ds:KeyInfo>
1266   <SoapEndpoint>http://ServiceProvider.com/soap</SoapEndpoint>
1267   <SingleLogoutServiceURL>http://ServiceProvider.com/liberty/slo</SingleLogoutServiceURL>
1268   <SingleLogoutServiceReturnURL>http://ServiceProvider.com/liberty/slo_return</SingleLogoutService
1269   eReturnURL>
1270   <FederationTerminationServiceURL>http://ServiceProvider.com/liberty/term</FederationTermination
1271   ServiceURL>
1272   <FederationTerminationServiceReturnURL>http://ServiceProvider.com/liberty/term_return</Federati
1273   onTerminationServiceReturnURL>
1274   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-
1275   idp-soap</FederationTerminationNotificationProtocolProfile>
1276   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-
1277   idp-http</FederationTerminationNotificationProtocolProfile>
1278   <SingleLogoutProtocolProfile>http://projectliberty.org/profiles/slo-idp-
1279   http</SingleLogoutProtocolProfile>
1280   <RegisterNameIdentifierProtocolProfile>http://projectliberty.org/profiles/rni-idp-soap</
1281   RegisterNameIdentifierProtocolProfile>
1282   <RegisterNameIdentifierProtocolProfile>http://projectliberty.org/profiles/rni-idp-http</
1283   RegisterNameIdentifierProtocolProfile>
1284   <RegisterNameIdentifierServiceURL>http://Provider.com/liberty/register_name</RegisterNameIdenti
1285   fierServiceURL>
1286   <RegisterNameIdentifierServiceReturnURL>http://Provider.com/liberty/ReturnURL</
1287   RegisterNameIdentifierServiceReturnURL>
1288   <AssertionConsumerServiceURL>http://ServiceProvider.com/liberty/assertion_consumer</AssertionCo
1289   nsumerServiceURL>
1290   <AuthnRequestsSigned>1</AuthnRequestsSigned>
1291 </SPDescriptor>
```

1292 4.3 Identity Provider Descriptor

1293 The additional identity provider-specific metadata are as follows:

1294 SingleSignOnServiceURL [Required]

1295 The identity provider's URL for accepting authentication requests for the Single Sign-On
1296 and Federation Protocol.

1297 SingleSignOnProtocolProfile [Required]

1298 The Single Sign-On Protocol profiles supported by the provider. Each element MUST
1299 contain a valid Single Sign-On Protocol profile identification URI.

1300 The schema fragment is as follows:

```

1301 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
1302 <complexType name="IDPDescriptorType">
1303 <complexContent>
1304 <extension base="lib:ProviderDescriptorType">
1305 <sequence>
1306 <element name="SingleSignOnServiceURL" type="anyURI"/>
1307 <element name="SingleSignOnProtocolProfile" type="anyURI"
1308 maxOccurs="unbounded"/>
1309 </sequence>
1310 </extension>
1311 </complexContent>
1312 </complexType>

```

1313 4.3.1 Example

```

1314 <IDPDescriptor>
1315 <ProviderID>http://IdentityProvider.com</ProviderID>
1316 <ds:KeyInfo>...</ds:KeyInfo>
1317 <SoapEndpoint>http://IdentityProvider.com/soap</SoapEndpoint>
1318 <SingleLogoutServiceURL>http://IdentityProvider.com/liberty/slo</SingleLogoutServiceURL>
1319 <SingleLogoutServiceReturnURL>http://IdentityProvider.com/liberty/slo_return</SingleLogoutServi
1320 ceReturnURL>
1321 <FederationTerminationServiceURL>http://IdentityProvider.com/liberty/term</FederationTerminatio
1322 nServiceURL>
1323 <FederationTerminationServiceReturnURL>http://IdentityProvider.com/liberty/term_return</Federat
1324 ionTerminationServiceReturnURL>
1325 <SingleSignOnServiceURL>http://IdentityProvider.com/liberty/sso</SingleSignOnServiceURL>
1326 <SingleSignOnProtocolProfile>http://projectliberty.org/profiles/brws-
1327 post</SingleSignOnProtocolProfile>
1328 </IDPDescriptor>

```

1329 5 Schema Definition

```

1330 <?xml version="1.0" encoding="UTF-8"?>
1331 <schema targetNamespace="http://projectliberty.org/schemas/core/2002/12"
1332 xmlns="http://www.w3.org/2001/XMLSchema"
1333 xmlns:AC="http://www.projectliberty.org/schemas/authctx/2002/05"
1334 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1335 xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1336 xmlns:lib="http://projectliberty.org/schemas/core/2002/12" elementFormDefault="qualified"
1337 attributeFormDefault="unqualified">
1338 <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation=" http://www.oasis-
1339 open.org/committees/security/docs/cs-sstc-schema-assertion-01.xsd"/>
1340 <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation=" http://www.oasis-
1341 open.org/committees/security/docs/cs-sstc-schema-protocol-01.xsd"/>
1342 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1343 schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
1344 <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05"
1345 schemaLocation="http://www.projectliberty.org/specs/liberty-architecture-authentication-context-
1346 v1.1.xsd"/>
1347
1348 <!-- Begin protocols schema -->
1349 <element name="ProviderID" type="anyURI"/>
1350 <element name="AuthnRequest" type="lib:AuthnRequestType"/>
1351 <complexType name="AuthnRequestType">
1352 <complexContent>
1353 <extension base="samlp:RequestAbstractType">
1354 <sequence>
1355 <element ref="lib:ProviderID"/>
1356 <element name="ForceAuthn" type="boolean" minOccurs="0"/>
1357 <element name="IsPassive" type="boolean" minOccurs="0"/>
1358 <element name="Federate" type="boolean" minOccurs="0"/>
1359 <element ref="lib:ProtocolProfile" minOccurs="0"/>
1360 <element ref="lib:AuthnContext" minOccurs="0"/>
1361 <element ref="lib:RelayState" minOccurs="0"/>
1362 <element name="AuthnContextComparison" type="lib:AuthnContextComparisonType"
1363 minOccurs="0" maxOccurs="1"/>
1364 </sequence>
1365 <attribute name="id" type="ID" use="optional"/>
1366 </extension>
1367 </complexContent>

```

```
1368 </complexType>
1369 <simpleType name="AuthnContextComparisonType">
1370   <restriction base="string">
1371     <enumeration value="exact"/>
1372     <enumeration value="minimum"/>
1373     <enumeration value="better"/>
1374   </restriction>
1375 </simpleType>
1376 <element name="RelayState"/>
1377 <element name="ProtocolProfile" type="anyURI"/>
1378 <element name="AuthnContext">
1379   <complexType>
1380     <choice>
1381       <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>
1382       <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>
1383     </choice>
1384   </complexType>
1385 </element>
1386 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
1387 <complexType name="AuthnRequestEnvelopeType">
1388   <complexContent>
1389     <extension base="lib:RequestEnvelopeType">
1390       <sequence>
1391         <element ref="lib:AuthnRequest"/>
1392         <element ref="lib:ProviderID"/>
1393         <element name="ProviderName" type="string" minOccurs="0"/>
1394         <element name="AssertionConsumerServiceURL" type="anyURI"/>
1395         <element ref="lib:IDPList" minOccurs="0"/>
1396         <element name="IsPassive" type="boolean" minOccurs="0"/>
1397       </sequence>
1398     </extension>
1399   </complexContent>
1400 </complexType>
1401 <complexType name="RequestEnvelopeType">
1402   <sequence>
1403     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1404   </sequence>
1405 </complexType>
1406 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
1407 <complexType name="AuthnResponseEnvelopeType">
1408   <complexContent>
1409     <extension base="lib:ResponseEnvelopeType">
1410       <sequence>
1411         <element ref="lib:AuthnResponse"/>
1412         <element name="AssertionConsumerServiceURL" type="anyURI"/>
1413       </sequence>
1414     </extension>
1415   </complexContent>
1416 </complexType>
1417 <complexType name="ResponseEnvelopeType">
1418   <sequence>
1419     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1420   </sequence>
1421 </complexType>
1422 <complexType name="SubjectType">
1423   <complexContent>
1424     <extension base="saml:SubjectType">
1425       <sequence>
1426         <element ref="lib:IDPProvidedNameIdentifier"/>
1427       </sequence>
1428     </extension>
1429   </complexContent>
1430 </complexType>
1431 <element name="AuthnResponse" type="lib:AuthnResponseType"/>
1432 <complexType name="AuthnResponseType">
1433   <complexContent>
1434     <extension base="samlp:ResponseType">
1435       <sequence>
1436         <element ref="lib:ProviderID"/>
1437         <element ref="lib:RelayState" minOccurs="0"/>
1438       </sequence>
1439       <attribute name="id" type="ID" use="optional"/>
1440     </extension>
1441   </complexContent>
```

```

1442 </complexType>
1443 <complexType name="AuthenticationStatementType">
1444   <complexContent>
1445     <extension base="saml:AuthenticationStatementType">
1446       <sequence>
1447         <element name="AuthnContext" minOccurs="0">
1448           <complexType>
1449             <sequence>
1450               <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
1451               <choice>
1452                 <element ref="AC:AuthenticationContextStatement"/>
1453                 <element name="AuthnContextStatementRef" type="anyURI"/>
1454               </choice>
1455             </sequence>
1456           </complexType>
1457         </element>
1458       </sequence>
1459       <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
1460       <attribute name="SessionIndex" type="string" use="optional"/>
1461     </extension>
1462   </complexContent>
1463 </complexType>
1464 <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
1465 <complexType name="RegisterNameIdentifierRequestType">
1466   <complexContent>
1467     <extension base="samlp:RequestAbstractType">
1468       <sequence>
1469         <element ref="lib:ProviderID"/>
1470         <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1471         <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1472         <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1473         <element ref="lib:RelayState" minOccurs="0"/>
1474       </sequence>
1475       <attribute name="id" type="ID" use="optional"/>
1476     </extension>
1477   </complexContent>
1478 </complexType>
1479 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1480 <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1481 <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1482 <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
1483 <complexType name="StatusResponseType">
1484   <complexContent>
1485     <extension base="samlp:ResponseAbstractType">
1486       <sequence>
1487         <element ref="lib:ProviderID"/>
1488         <element ref="samlp:Status"/>
1489         <element ref="lib:RelayState" minOccurs="0"/>
1490       </sequence>
1491       <attribute name="id" type="ID" use="optional"/>
1492     </extension>
1493   </complexContent>
1494 </complexType>
1495 <element name="FederationTerminationNotification"
1496 type="lib:FederationTerminationNotificationType"/>
1497 <complexType name="FederationTerminationNotificationType">
1498   <complexContent>
1499     <extension base="samlp:RequestAbstractType">
1500       <sequence>
1501         <element ref="lib:ProviderID"/>
1502         <element ref="saml:NameIdentifier"/>
1503         <element ref="lib:RelayState" minOccurs="0"/>
1504       </sequence>
1505       <attribute name="id" type="ID" use="optional"/>
1506     </extension>
1507   </complexContent>
1508 </complexType>
1509 <element name="LogoutRequest" type="lib:LogoutRequestType"/>
1510 <complexType name="LogoutRequestType">
1511   <complexContent>
1512     <extension base="samlp:RequestAbstractType">
1513       <sequence>
1514         <element ref="lib:ProviderID"/>
1515         <element ref="saml:NameIdentifier"/>

```

```
1516         <element name="SessionIndex" type="string" minOccurs="0"/>
1517         <element ref="lib:RelayState" minOccurs="0"/>
1518     </sequence>
1519     <attribute name="id" type="ID" use="optional"/>
1520 </extension>
1521 </complexContent>
1522 </complexType>
1523 <element name="LogoutResponse" type="lib:StatusResponseType"/>
1524 <complexType name="SignedSAMLRequestType">
1525     <complexContent>
1526         <extension base="samlp:RequestType">
1527             <attribute name="id" type="ID" use="optional"/>
1528         </extension>
1529     </complexContent>
1530 </complexType>
1531 <!-- End protocols schema -->
1532 <!-- Begin assertion schema -->
1533 <element name="Assertion" type="lib:AssertionType"/>
1534 <complexType name="AssertionType">
1535     <complexContent>
1536         <extension base="saml:AssertionType">
1537             <attribute name="InResponseTo" type="saml:IDReferenceType"/>
1538             <attribute name="id" type="ID" use="optional"/>
1539         </extension>
1540     </complexContent>
1541 </complexType>
1542 <!-- End assertion schema -->
1543 <!-- Begin IDP list schema -->
1544 <element name="IDPList" type="lib:IDPListType"/>
1545 <complexType name="IDPListType">
1546     <sequence>
1547         <element ref="lib:IDPEntries"/>
1548         <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
1549     </sequence>
1550 </complexType>
1551 <element name="IDPEntry">
1552     <complexType>
1553         <sequence>
1554             <element ref="lib:ProviderID"/>
1555             <element name="ProviderName" type="string" minOccurs="0"/>
1556             <element name="Loc" type="anyURI"/>
1557         </sequence>
1558     </complexType>
1559 </element>
1560 <element name="IDPEntries">
1561     <complexType>
1562         <sequence>
1563             <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
1564         </sequence>
1565     </complexType>
1566 </element>
1567 <element name="GetComplete" type="anyURI"/>
1568 <!-- End IDP list schema -->
1569 <!-- Begin provider metadata schema -->
1570 <complexType name="ProviderDescriptorType">
1571     <sequence>
1572         <element name="ProviderID" type="anyURI"/>
1573         <element ref="ds:KeyInfo" minOccurs="0"/>
1574         <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
1575         <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
1576         <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
1577         <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
1578         <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
1579         <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"
1580 minOccurs="0" maxOccurs="unbounded"/>
1581         <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"
1582 maxOccurs="unbounded"/>
1583         <element name="RegisterNameIdentifierProtocolProfile" type="anyURI" minOccurs="0"
1584 maxOccurs="unbounded"/>
1585         <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs="0"/>
1586         <element name="RegisterNameIdentifierServiceReturnURL" type="anyURI" minOccurs="0"/>
1587     </sequence>
1588 </complexType>
1589 <element name="SPDescriptor" type="lib:SPDescriptorType"/>
```

```
1590 <complexType name="SPDescriptorType">
1591   <complexContent>
1592     <extension base="lib:ProviderDescriptorType">
1593       <sequence>
1594         <element name="AssertionConsumerServiceURL" type="anyURI"/>
1595         <element name="AuthnRequestsSigned" type="boolean"/>
1596       </sequence>
1597     </extension>
1598   </complexContent>
1599 </complexType>
1600 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
1601 <complexType name="IDPDescriptorType">
1602   <complexContent>
1603     <extension base="lib:ProviderDescriptorType">
1604       <sequence>
1605         <element name="SingleSignOnServiceURL" type="anyURI"/>
1606         <element name="SingleSignOnProtocolProfile" type="anyURI"
1607 maxOccurs="unbounded"/>
1608       </sequence>
1609     </extension>
1610   </complexContent>
1611 </complexType>
1612 <!-- End provider metadata schema -->
1613 </schema>
```

1614 6 References

- 1615 [LibertyBindProf] Rouault, J., & Wason, T., eds. (January 2003). “Liberty Bindings and
1616 Profiles Specification,” Version 1.1. Liberty Alliance Project,
1617 <<http://www.projectliberty.org/specs/>>.
- 1618 [LibertyGloss] Mauldin, H., & Wason, T., eds. (January 2003). “Liberty Architecture
1619 Glossary,” Version 1.1. Liberty Alliance Project,
1620 <<http://www.projectliberty.org/specs/>>.
- 1621 [RFC2119] Bradner, S. (March 1997). “Key words for use in RFCs to Indicate
1622 Requirement Levels,” RFC 2119. The Internet Engineering Task Force,
1623 <<http://www.rfc-editor.org/rfc/rfc2119.txt>> [18 December 2002].
- 1624 [RFC3280] Housley, R. (April 2002). “Internet X.509 Public Key Infrastructure
1625 Certificate and Certificate Revocation List (CRL) Profile,” RFC 3280.
1626 The Internet Engineering Task Force, <[http://www.rfc-
editor.org/rfc/rfc3280.txt](http://www.rfc-
1627 editor.org/rfc/rfc3280.txt)> [18 December 2002]
- 1628 [SAMLCore] Hallam-Baker, P., Maler, E., eds. (05 Nov. 2002). “Assertions and
1629 Protocol for the OASIS Security Assertion Markup Language (SAML),”
1630 Version 1.0, OASIS Standard. Organization for the Advancement of
1631 Structured Information Standards, <[http://www.oasis-
open.org/committees/security/#documents](http://www.oasis-
1632 open.org/committees/security/#documents)> [18 December 2002].
- 1633 [Schema1] Thompson, H. S., Beech, D., Maloney, M., & Mendleson, N., eds. (May
1634 2002). “XML Schema Part 1: Structures,” Recommendation. World
1635 Wide Web Consortium, <<http://www.w3.org/TR/xmlschema-1/>> [18
1636 December 2002].
- 1637 [Schema2] Biron, P. V., & Malhotra, A., eds. (May 2002). “XML Schema Part 2:
1638 Datatypes,” Recommendation. World Wide Web Consortium,
1639 <<http://www.w3.org/TR/xmlschema-2/>> [18 December 2002].
- 1640 [XMLDsig] D. Eastlake, J. Reagle, D. Solo et al, (12 February 2002). “XML-
1641 Signature Syntax and Processing”, Recommendation. World Wide Web

- 1642 Consortium, <<http://www.w3.org/TR/xmlsig-core>> [18 December
1643 2002].
- 1644 [XMLCanon] J. Boyer, D. Eastlake, & J. Reagle (18 July 2002). “Exclusive XML
1645 Canonicalization,” Recommendation. World Wide Web Consortium,
1646 <<http://www.w3.org/TR/xml-exc-c14n>> [18 December 2002].
- 1647 [XPointer] DeRose, S., Maler, E., & Daniel, R. (16 August 2002). “XML Pointer
1648 Language (XPointer),” Working Draft. World Wide Web Consortium,
1649 <<http://www.w3.org/TR/xptr/>> [18 December 2002].