



Liberty Metadata Description and Discovery Specification

Version: 1.0

Editors:

Peter Davis, NeuStar, Inc.

Contributors:

Paul Madsen, Entrust, Inc.

Jeff Hodges, Sun Microsystems, Inc.

Bronislav Kavsan, RSA Security, Inc.

Scott Cantor, Internet2

Abstract:

This document details the metadata schema and methods of resolution for discovering the location of metadata instances for the Liberty Identity Federation Framework

Filename: liberty-metadata-v1.0.pdf

1 Notice

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

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

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

24 Liberty Alliance Project
25 Licensing Administrator
26 c/o IEEE-ISTO
27 445 Hoes Lane
28 Piscataway, NJ 08855-1331, USA
29 info@projectliberty.org

30 **Contents**

| | | |
|----|---|----|
| 31 | 1. Introduction | 4 |
| 32 | 2. Metadata Schema | 5 |
| 33 | 3. Publishing the Metadata | 19 |
| 34 | 4. Metadata Resolution and Retrieval | 23 |
| 35 | 5. Post Processing of the Metadata Document | 25 |
| 36 | 6. Security Considerations | 27 |
| 37 | 7. Metadata XSD | 28 |
| 38 | References | 32 |

39 1. Introduction

40 Within **ID-FF** version 1.1 specification [[LibertyProtSchema1.1](#)] of the Liberty Alliance protocols [[LibertyProtSchema](#)], basic metadata were exchanged out-of-band between entities. This specification more formally
41 describes metadata, as well as protocols to facilitate real-time requests for this data allowing for more spontaneous
42 conversations between Liberty enabled entities.
43

44 There are three primary functions for this metadata:

- 45 • declarations of entity metadata for providers, principals and devices, and affiliations
- 46 • entity trust metadata, which enables entities to cast business decisions based on the characteristic trust information
47 provided in this class, conveyed through document signature(s), server authenticated protected channel delivery of
48 the instance using TLS [[RFC2246](#)] as amended by [[RFC3546](#)], DNS zone signatures, and, optionally, additional
49 material that publishers may convey within the `Extension` and `AdditionalMetaLocation` elements
- 50 • origin and document verification through signature use in (server authenticated) HTTPS retrieval of the instance
51 documents, DNS signatures, and document level signatures

52 This document presents extensions to the model for metadata described in Liberty ID-FF versions 1.1 to better support
53 ad-hoc interactions between entities. The location of cryptographic keys in a distributed-computing architecture that
54 contains "arms-length" peer domains presents an opportunity for some fresh thinking. Conventional solutions to this
55 problem fail to fully exploit the potential of the evolving Web Services architecture to minimize administrative costs.
56 Liberty ID-FF version 1.2 [[LibertyProtSchema](#)], ID-WSF and ID-SIS set of specification [[LibertyIDFFOverview](#)]
57 operations between previously uninitiated parties will benefit from any mechanisms that simplify how keying
58 material and service interface points can be discovered, leading to mechanisms for trust establishment and services
59 invocations in both direct and indirect means.

60 1.1. Notation and Conventions

61 This specification uses schema documents conforming to W3C XML Schema [[Schema1](#)] and normative text to
62 describe the syntax and semantics of XML-encoded protocol messages.

63 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
64 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

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

68 Within this document, a *publisher* is the subject of, or authorized representing party for, the subject of the instance
69 document, as referenced by `providerID` and a *consumer* is the entity resolving, retrieving, or otherwise processing
70 the instance as a relying party to its information.

71 1.2. Overview

72 The metadata protocols and schemas specified in this document will enable two Liberty-enabled entities to exchange
73 or request cryptographic keys, service endpoints information, and protocol and profile support in real time, allowing
74 dynamic interactions between these parties, eliminating the need for out-of-band negotiations to have occurred a-priori.
75 The addition of interactions between separate authentication authorities and identity chaining in the Liberty **ID-WSF**
76 will depend upon this exchange, as portions of a principle's identity may be previously established outside the range
77 of providers established agreements.

78 2. Metadata Schema

79 The metadata schema allows several methods of representation:

- 80 • A document expressing metadata describing an entity, referenced by a `providerID`, acting in the role of a Service
81 Provider or an Identity Provider, or both, within the `EntityDescriptor` Node
- 82 • As a single instance document expressing metadata describing multiple entities, each referenced by a
83 `providerID`, each entity acting as declared above. The metadata for each entity is contained within sep-
84 arate `EntityDescriptor` nodes, each being an immediate descendant of the plural `EntitiesDescriptor`
85 node
- 86 • As a single instance document describing a set of entities referenced by `providerID`'s collectively identified by
87 an `affiliationID`, and maintained by an entity referenced by its `affiliationOwnerID`, where each entity's
88 metadata is separately published.

89 The first two forms may also be expressed as multiple documents, involving additional metadata, which MAY
90 be of a namespace `urn:liberty:metadata:2003-08` (the default), or another namespace, as specified by the
91 element `Location`'s corresponding `namespace` attribute. Additionally, the document location(s) may be identified
92 by multiple NPATR resource records.

93 2.1. Schema Declarations

94 The metadata schema is constructed to allow an entity, referenced by one or more `providerID`'s, to publish single or
95 multiple schema instances to describe their identity services architecture.

96 The primary container for a published document is either `EntityDescriptor` or the plural form
97 `EntitiesDescriptor` (used when an affiliated set of entities chooses to publish a consolidated set of meta-
98 data documents as one).

99 The expected immediate child nodes of `EntityDescriptor` are one or more of:

- 100 • `SPDescriptor`
- 101 • `IDPDescriptor`

102 or one of:

- 103 • `AffiliationDescriptor`

104 which are described below. Additionally, an extension point `Extension` is provided in order to convey additional
105 metadata.

106 2.1.1. Namespaces in Metadata

107 The following namespace declarations are used to complete the metadata schema:

- 108 • `ds:` is described by the W3C XML Signature [\[XMLDsig\]](#) schema
- 109 • `saml:` is described by the OASIS Security Services SAML 1.1 Assertion [\[SAMLCore11\]](#) schema

110 In addition, the Liberty Utility Schema is included allowing the common Extension element that is used throughout
111 the Liberty Specifications suite

112 Schema Fragment:

```
113
114 <xs:schema targetNamespace="urn:liberty:metadata:2003-08" xmlns:xs="http://www.w3.org/2001/XMLSchema"
115 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:ds="http://www.w3.org/2000/09/xmldsig
116 ig#" xmlns="urn:liberty:metadata:2003-08" elementFormDefault="qualified" attributeFormD
117 efault="unqualified" version="1.0">
118   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="http://www.w3.org/TR/xmldsig-core/x
119 mldsig-core-schema.xsd"/>
120   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="oasis-ss
121 tc-saml-schema-assertion-1.1.xsd"/>
122   <xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http://www.w3.org/2001/xml.xsd"/>
123
124   <xs:include schemaLocation="liberty-utility-v1.0.xsd"/>
125
```

126 2.1.2. Datatype entityType

127 The datatype entityType restricts the XML data to a length of 1024 bytes.

128 Additionally, the entityType structure is defined by the following BNF, derived from *URI Specification*
129 [\[RFC2396\]](#) as modified by [\[RFC2732\]](#)

```
130 BNF for Liberty entityIdentifiers
131 # constraint on absoluteURI
132 entityType = absoluteURI [ "#" fragment ]
133 absoluteURI = scheme ":" ( hier_part | opaque_part )
134
135 # constraint on hier_part (net_path only)
136 hier_part = net_path [ "?" query ]
137 opaque_part = uric_no_slash *uric
138
139 uric_no_slash = unreserved | escaped | ";" | "?" | ":" | "@" |
140 "&" | "=" | "+" | "$" | ","
141
142 net_path = "/" authority [ abs_path ]
143 abs_path = "/" path_segments
144
145 ; pragmatically, scheme SHOULD be an officially IANA registered URI scheme
146 ; http://www.iana.org/assignments/uri-schemes
147 scheme = alpha *( alpha | digit | "+" | "-" | "." )
148
149 authority = server | reg_name
150
151 reg_name = 1*( unreserved | escaped | "$" | "," |
152 ";" | ":" | "@" | "&" | "=" | "+" )
153
154 server = [ [ userinfo "@" ] hostport ]
155 userinfo = *( unreserved | escaped |
156 ";" | ":" | "&" | "=" | "+" | "$" | "," )
157
158 hostport = host [ ":" port ]
159 ; constraint on host (no ipAddress)
160 host = hostname
161 hostname = *( domainlabel "." ) toplabel [ "." ]
162 domainlabel = alphanum | alphanum *( alphanum | "-" ) alphanum
163 toplabel = alpha | alpha *( alphanum | "-" ) alphanum
164 port = *digit
165
166 path = [ abs_path | opaque_part ]
167 path_segments = segment *( "/" segment )
168 segment = *pchar *( ";" param )
```

```

169 param      = *pchar
170 pchar      = unreserved | escaped |
171             ":" | "@" | "&" | "=" | "+" | "$" | ", "
172
173 query      = *uric
174
175 fragment   = *uric
176
177 uric       = reserved | unreserved | escaped
178 reserved   = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" |
179             "$" | ", "
180 unreserved = alphanum | mark
181 mark       = "-" | "_" | "." | "!" | "~" | "*" | "'" |
182             "(" | ")"
183
184 escaped     = "%" hex hex
185 hex        = digit | "A" | "B" | "C" | "D" | "E" | "F" |
186             "a" | "b" | "c" | "d" | "e" | "f"
187
188 alphanum   = alpha | digit
189 alpha      = lowalpha | upalpha
190
191 lowalpha   = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
192             "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
193             "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
194 upalpha    = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
195             "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
196             "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
197 digit     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
198             "8" | "9"
199
200

```

201 The schema fragment for `entityIDType`:

```

202
203 <xsd:simpleType name="entityIDType">
204   <xsd:restriction base="xsd:anyURI">
205     <xsd:maxLength id="maxlengid" value="1024"/>
206   </xsd:restriction>
207 </xsd:simpleType>
208

```

209 2.1.3. Common Attributes

210 Several common attributes are defined and generally used throughout the schema:

- 211 • `libertyPrincipalIdentifier` of type `entityIDType` used to provide a pointer to contact an entity's
212 metadata which MAY be dereferencable
- 213 • `providerID` of type `entityIDType` indicates the providerID of the entity described by the descendants of the
214 node
- 215 • `validUntil` of type `dateTime` indicates the expiration date and time of the node (and its descendants). If
216 `dateTime` expressions evaluate to nonequivalent values, parsers MUST adhere to the most restrictive value (the
217 earliest `dateTime`).
- 218 • `cacheDuration` of type `duration` indicates the maximum elapsed time a consumer may cache the metadata
219 document (or fragment). Consistent with the `validUntil` attribute, the most restrictive value MUST be used
220 when conflicting cache directives occur

221 Publishers **MUST** provide either a `validUntil` or `cacheDuration` attribute when publishing metadata. Since this
222 directive is available at both the top-level `EntityDescriptor` and its immediate descendants, care should be taken in selecting
223 expiration settings. It is **RECOMMENDED** that publishers express document expiration at the `EntityDescriptor`
224 or `AffiliationDescriptor` element only, and not on the child nodes.

225 All Liberty time values have the type `dateTime`, which is built in to the W3C XML Schema Datatypes specification
226 [Schema2]. Liberty time values **MUST** be expressed in UTC form, indicated by a "Z" immediately following the time
227 portion of the value.

228 Liberty entities **SHOULD NOT** rely on other applications supporting time resolution finer than seconds, as imple-
229 mentations **MAY** ignore fractional second components specified in timestamp values. Implementations **MUST NOT**
230 generate time instants that specify leap seconds.

231 The consumer **MAY** reset the retrieval `dateTime`, effectively resetting the duration clock (see Section 5.2) if consumers
232 send an *HTTP (1.1) [RFC2616]* request to the publisher URL with a header *If-Modified-Since: [last retrieval*
233 *dateTime]*, the publisher server returns a *304 Not-Modified* response, and the publisher expresses the expiration as
234 a `cacheDuration`,

235 The schema fragment for the common attributes:

```
236  
237 <xsd:attribute name="libertyPrincipalIdentifier" type="entityIDType" />  
238 <xsd:attribute name="providerID" type="entityIDType" />  
239 <xsd:attribute name="validUntil" type="xs:dateTime" />  
240 <xsd:attribute name="cacheDuration" type="xs:duration" />  
241
```

242 2.1.4. Common DataTypes

243 There are several common datatypes defined globally, and used throughout the schema:

244 2.1.4.1. organizationType Data Type

245 The `organizationType` datatype provides some basic information consumers may require when interacting with a
246 principal:

- 247 • `OrganizationName` of type `string` [Required, 1-many]: a localizable ([XML] Section 2.12 Language Identifi-
248 cation) Organizational Name of the entity, generally the complete Organization Legal name
- 249 • `OrganizationDisplayName` of type `string` [Required, 1-many]: a localizable organization name suitable for
250 display to a principal
- 251 • `OrganizationURL` of type `anyURI` [Required, 1-many]: a localizable URL of the organization suitable for
252 dereferencing by a user-agent, which may be used for directing a principal for additional information on the entity

253 Localized strings SHOULD be used when present in the metadata instance, and the preferred language of the target
254 entity is known by the consumer

```
255
256 <xs:complexType name="organizationType">
257   <xs:sequence>
258     <xs:element maxOccurs="unbounded" name="OrganizationName" type="organizationNameType"/>
259     <xs:element maxOccurs="unbounded" name="OrganizationDisplayName"
260       type="organizationDisplayNameType"/>
261     <xs:element maxOccurs="unbounded" name="OrganizationURL" type="localizedURIType"/>
262     <xs:element minOccurs="0" ref="Extension"/>
263   </xs:sequence>
264 </xs:complexType>
265 <xs:complexType name="organizationNameType">
266   <xs:simpleContent>
267     <xs:extension base="xs:string">
268       <xs:attribute ref="xml:lang"/>
269     </xs:extension>
270   </xs:simpleContent>
271 </xs:complexType>
272 <xs:complexType name="organizationDisplayNameType">
273   <xs:simpleContent>
274     <xs:extension base="xs:string">
275       <xs:attribute ref="xml:lang" use="required"/>
276     </xs:extension>
277   </xs:simpleContent>
278 </xs:complexType>
279 <xs:complexType name="localizedURIType">
280   <xs:simpleContent>
281     <xs:extension base="xs:anyURI">
282       <xs:attribute ref="xml:lang" use="required"/>
283     </xs:extension>
284   </xs:simpleContent>
285 </xs:complexType>
286
```

287 2.1.4.2. contactType Data Type

288 The contactType data type conveys general contact information for human-to-human contact regarding an entity. It is
289 defined with the the following attributes:

- 290 • libertyPrincipalIdentifier [Optional]: a Principal's dereferencable nameIdentifier of type entityIDType
291 which may point to an online instance of the person's PIP profile
- 292 • contactType [Required]: the type of contact, which may be one of: technical, administrative, billing,
293 or other. The default value is technical

294 The elements defined by this type:

- 295 • Company [Optional, 0-1]: The company name of type xs:string, by which the cited individual is employed for
296 the purposes relating to the instance document
- 297 • GivenName [Optional, 0-1]: The given name of the contact of type xs:string
- 298 • SurName [Optional, 0-1]: The surname of the contact of type xs:string
- 299 • EmailAddress [Optional, 0-many]: The email address of the contact of type xs:anyURI
- 300 • TelephoneNumber [Optional, 0-many]: The contact's telephone number of type xs:string

301 The schema fragment for contactType:

```
302
303 <xs:complexType name="contactType">
304   <xs:sequence>
305     <xs:element maxOccurs="1" minOccurs="0" name="Company" type="xs:string" />
306     <xs:element maxOccurs="1" minOccurs="0" name="GivenName" type="xs:string" />
307     <xs:element maxOccurs="1" minOccurs="0" name="SurName" type="xs:string" />
308     <xs:element maxOccurs="unbounded" minOccurs="0" name="EmailAddress" type="xs:anyURI" />
309     <xs:element maxOccurs="unbounded" minOccurs="0" name="TelephoneNumber" type="xs:string" />
310     <xs:element minOccurs="0" ref="Extension" />
311   </xs:sequence>
312   <xs:attribute ref="libertyPrincipalIdentifier" use="optional" />
313   <xs:attribute name="contactType" type="attr.contactType" use="required" />
314 </xs:complexType>
315 <xs:simpleType name="attr.contactType">
316   <xs:restriction base="xs:string">
317     <xs:enumeration value="technical" />
318     <xs:enumeration value="administrative" />
319     <xs:enumeration value="billing" />
320     <xs:enumeration value="other" />
321   </xs:restriction>
322 </xs:simpleType>
323
```

324 2.1.4.3. keyDescriptorType Complex Type

325 The elements of type keyDescriptorType convey to a consumer two cryptographic metadata statements:

- 326 • encryption preferences described by EncryptionMethod [Optional, 0-1], whose valid URI values are defined in
327 [\[xmlenc-core\]](#), and
328 KeySize [Optional, 0-1] which may optionally constrain the length of keys used by the consumer when interacting
329 with another entity
- 330 • Key Material information located in ds:KeyInfo [Optional, 0-1] as described by [\[XMLDsig\]](#)

331 The schema fragment for keyDescriptorType:

```
332
333 <xs:complexType name="keyDescriptorType">
334   <xs:sequence>
335     <xs:element minOccurs="0" name="EncryptionMethod" type="xs:anyURI" />
336     <xs:element minOccurs="0" name="KeySize" type="xs:integer" />
337     <xs:element minOccurs="0" ref="ds:KeyInfo" />
338     <xs:element minOccurs="0" ref="Extension" />
339   </xs:sequence>
340   <xs:attribute name="use" type="keyTypes" use="required" />
341 </xs:complexType>
```

342 The KeyDescriptor includes the required attribute use of type keyTypes. use may have values of encryption
343 or signing:

```
344 <xs:simpleType name="keyTypes">
345   <xs:restriction base="xs:string">
346     <xs:enumeration value="encryption" />
347     <xs:enumeration value="signing" />
348   </xs:restriction>
349 </xs:simpleType>
```

350 2.1.4.4. providerDescriptorType Complex Type

351 The providerDescriptorType is a utility type, which describes generic metadata for any Liberty-enabled entity with
352 attributes that include:

- 353 • id [Optional]: The fragment identifier of the instance node (required if the node is signed as described in
354 [Section 5.1](#)).
- 355 • validUntil [Optional]: The dateTime the fragment expires. Processing rules are described in [Section 2.1.3](#) [7].
- 356 • cacheDuration [Optional]: The maximum duration a consumer may cache the fragment. Processing rules are
357 described in [Section 2.1.3](#) [7].
- 358 • protocolSupportEnumeration [Required] describes the protocol release supported by the entity de-
359 scribed by providerID. NMTOKENS type allows the enumeration of a set of Liberty ID-FF protocol
360 releases which the interfaces described within MUST support. The datatype of the tokens MUST be
361 URNs (presently <http://projectliberty.org/schemas/core/2002/12> for release ID-FF 1.1 and
362 <urn:liberty:iff:2003-08> for release ID-FF 1.2). Subsequent releases of ID-FF shall express protocol
363 support using the defined namespace attribute of the corresponding ID-FF schema.
364 When an entity supports both ID-FF 1.1 and ID-FF 1.2 protocols, it SHOULD publish a ID-FF 1.1 valid in-
365 stance and make reference to it within AdditionalMetaLocation, using the appropriate corresponding names-
366 pace identifier for that schema. Metadata consumers MUST retrieve (or otherwise obtain) this instance if they
367 intend to use the protocols of ID-FF 1.1. If publisher entities support both protocols on the same SoapEndPoint,
368 they MAY publish one document which describes both protocols uniformly, citing both protocols in the attribute
369 protocolSupportEnumeration or they MAY make reference to it using AdditionalMetaLocation. Con-
370 sumers in possession of an ID-FF 1.1 provider's metadata obtained in an "Out-of-Band" manner as described in that
371 version of the specification, MAY continue to use this instance, but SHOULD check for a newer version whenever
372 possible.

373 The elements describing the entity include:

- 374 • KeyDescriptor [Optional, 0-many] expresses a set of keying material and key metadata which the cooresponding
375 entity providerID will use within Liberty protocols and interactions.
- 376 • SoapEndpoint [Required, 1] The provider's SOAP endpoint URI.
- 377 • SingleLogoutServiceURL [Optional, 1] The URL used for user-agent-based Single Logout Protocol profiles.
- 378 • SingleLogoutServiceReturnURL [Optional, 0-1] The URL to which the provider redirects at the end of user-
379 agent-based Single Logout Protocol profiles.
- 380 • FederationTerminationServiceURL [Optional, 0-1] The URL used for user-agent-based Federation Termi-
381 nation Notification Protocol profiles.
- 382 • FederationTerminationServiceReturnURL [Optional, 0-1] The URL to which the provider redirects at the
383 end of user-agent-based Federation Termination Notification Protocol profiles.
- 384 • FederationTerminationNotificationProtocolProfile [Optional, 0-many] The Federation Termination
385 Notification Protocol profiles supported by the provider. Each value of the element MUST contain a valid
386 Federation Termination Notification Protocol profile identification URI as defined in [\[LibertyBindProf\]](#). The
387 absence of this element SHALL mean that provider does not support any profile of the Federation Termination
388 Notification Protocol.

- 389 • SingleLogoutProtocolProfile [Optional, 0-many] The Single Logout Protocol profiles supported by the
 390 provider. Each element MUST contain a valid Single Logout Protocol profile identification URI. The absence of
 391 this element SHALL mean that the provider does not support any profile of the Single Logout Protocol.

- 392 • RegisterNameIdentifierProtocolProfile [Optional, 0-many] The provider's preferred Register Name
 393 Identifier Protocol profile, which should be used by other providers when registering a new identifier. Each element
 394 MUST contain a valid Register Name Identifier Protocol profile identification URI as defined in [\[LibertyBindProf\]](#).
 395 The absence of this element SHALL mean that the provider does not support any profile of the Register Name
 396 Identifier Protocol.

- 397 • RegisterNameIdentifierServiceURL [Optional, 0-1] The URL used for user-agent-based Register Name
 398 Identifier Protocol profiles.

- 399 • RegisterNameIdentifierServiceReturnURL [Optional, 0-1] The provider's redirecting URL for use after
 400 HTTP name registration has taken place.

- 401 • RelationshipTerminationNotificationProtocolProfile [Optional, 0-many] an unbounded URI type
 402 describing the profile(s) the entity supports for relationship termination as defined in [\[LibertyBindProf\]](#)

- 403 • NameIdentifierMappingProtocolProfile [Optional, 0-many] of type anyURI, which indicates the profile
 404 of the NameIdentifierMapping protocol supported by the Provider.

- 405 • Organization [Optional, 0-1] The Organization (see [Section 2.1.4.1](#)) information about the provider.

- 406 • ContactPerson [Optional, 0-many] A Container expressing one or more contacts responsible for technical,
 407 administrative, billing, or other information concerning an identity service implementation expressed in the
 408 metadata (see [Section 2.1.4.2](#))

- 409 • AdditionalMetaLocation [Optional, 0-many] The location of other relevant metadata about the provider which
 410 MAY contain the attribute namespace, indicating the namespace of the target document.

- 411 • Extension [Optional, 0-1] Provides for metadata extensions describing an *SP* or *IdP*

- 412 • ds:Signature [Optional, 0-1] An optional signature of the provider metadata (see [Section 5.1](#))

413 Each of these elements is optional. The schema fragment for providerDescriptorType:

```

414 <xs:complexType name="providerDescriptorType">
415   <xs:sequence>
416     <xs:element maxOccurs="unbounded" minOccurs="0" name="KeyDescriptor" type="keyDescriptorType" />
417   </xs:sequence>
418 </xs:complexType>
419   <xs:element minOccurs="0" name="SoapEndpoint" type="xs:anyURI" />
420   <xs:element minOccurs="0" name="SingleLogoutServiceURL" type="xs:anyURI" />
421   <xs:element minOccurs="0" name="SingleLogoutServiceReturnURL" type="xs:anyURI" />
422   <xs:element minOccurs="0" name="FederationTerminationServiceURL" type="xs:anyURI" />
423   <xs:element minOccurs="0" name="FederationTerminationServiceReturnURL" type="xs:anyURI" />
424   <xs:element maxOccurs="unbounded" minOccurs="0"
425     name="FederationTerminationNotificationProtocolProfile" type="xs:anyURI" />
426   <xs:element maxOccurs="unbounded" minOccurs="0" name="SingleLogoutProtocolProfile" type="xs:anyURI" />
427   <xs:element maxOccurs="unbounded" minOccurs="0"
428     name="RegisterNameIdentifierProtocolProfile" type="xs:anyURI" />
429   <xs:element minOccurs="0" name="RegisterNameIdentifierServiceURL" type="xs:anyURI" />
430   <xs:element minOccurs="0" name="RegisterNameIdentifierServiceReturnURL" type="xs:anyURI" />
431   <xs:element maxOccurs="unbounded" minOccurs="0"
432     name="RelationshipTerminationNotificationProtocolProfile" type="xs:anyURI" />
433   <xs:element maxOccurs="unbounded" minOccurs="0"
434     name="NameIdentifierMappingProtocolProfile" type="xs:anyURI" />
435   <xs:element minOccurs="0" name="Organization" type="organizationType" />
436
437

```

```

438     <xs:element maxOccurs="unbounded" minOccurs="0" name="ContactPerson" type="contactType" />
439     <xs:element maxOccurs="unbounded" minOccurs="0"
440       name="AdditionalMetaLocation" type="additionalMetadataLocationType" />
441     <xs:element minOccurs="0" ref="Extension" />
442     <xs:element minOccurs="0" ref="ds:Signature" />
443   </xs:sequence>
444   <xs:attribute name="protocolSupportEnumeration" type="xs:NMTOKENS" use="required" />
445   <xs:attribute name="id" type="xs:ID" use="optional" />
446   <xs:attribute ref="validUntil" use="optional" />
447   <xs:attribute ref="cacheDuration" use="optional" />
448 </xs:complexType>
449

```

450 2.1.5. Descriptors for Entities

451 2.1.5.1. SPDescriptor Element

452 SPDescriptor extends providerDescriptorType with the following elements:

453 • AssertionConsumerServiceURL [Required, 1-many] One or more URI(s) of the SP for receiving Authentica-
454 tion Assertions from an authenticating party. When an SP sends an *AuthNRequest* to the IdP, it may indicate the
455 preferred AssertionConsumerServiceURL using the provided id (QNAME) attribute to direct the principal to
456 for consumption of the *AuthNResponse*.
457 IdP's should inspect the Service Providers metadata for the appropriate URL, or the default (indicated
458 by the isDefault attribute) location, if no id is provided. Publishers MUST express only one default
459 AssertionConsumerServiceURL. AssertionConsumerServiceURL requires the following attributes:
460

461 • id [Required]. The fragment identifier of the AssertionConsumerServiceURL used as a reference in an
462 AuthNRequest.

463 • isDefault [Required]. A boolean indicator for the default AssertionConsumerServiceURL value to use
464 when no identifier is provided in the request.

465 • AuthnRequestsSigned [Required, 1] boolean element indicating whether the Service Provider will always
466 signed it's *AuthNRequests*

467 the schema fragment for SPDescriptor:

```

468
469 <xs:complexType name="SPDescriptorType">
470   <xs:complexContent>
471     <xs:extension base="providerDescriptorType">
472       <xs:sequence>
473         <xs:element maxOccurs="unbounded" name="AssertionConsumerServiceURL">
474           <xs:complexType>
475             <xs:simpleContent>
476               <xs:extension base="xs:anyURI">
477                 <xs:attribute name="id" type="xs:ID" use="required" />
478                 <xs:attribute default="false" name="isDefault" type="xs:boolean" />
479               </xs:extension>
480             </xs:simpleContent>
481           </xs:complexType>
482         </xs:element>
483         <xs:element name="AuthnRequestsSigned" type="xs:boolean" />
484       </xs:sequence>
485     </xs:extension>
486   </xs:complexContent>

```

487 </xs:complexType>
488

489 2.1.5.2. IDPDescriptor Element

490 IDPDescriptor extends providerDescriptorType with the following elements:

- 491 • SingleSignOnServiceURL [Required, 1]. The identity provider's URL for accepting authentication requests for
492 the Single Sign-On and Federation Protocol.
- 493 • SingleSignOnProtocolProfile [Required, 1-many]. The Single Sign-On Protocol profiles supported by the
494 provider. Each element MUST contain a valid Single Sign-On Protocol profile identification URI as defined in
495 [\[LibertyBindProf\]](#).
- 496 • IntroductionNotificationProtocolProfile [Optional, 0-many] of URI type describes the profile of this
497 protocol supported by the identity provider as defined in [\[LibertyBindProf\]](#).

498 The schema fragment for IDPDescriptor:

```
499 <xs:complexType name="IDPDescriptorType">  
500   <xs:complexContent>  
501     <xs:extension base="providerDescriptorType">  
502       <xs:sequence>  
503         <xs:element name="SingleSignOnServiceURL" type="xs:anyURI"/>  
504         <xs:element maxOccurs="unbounded" name="SingleSignOnProtocolProfile" type="xs:anyURI"/>  
505         <xs:element maxOccurs="unbounded" minOccurs="0"  
506           name="IntroductionNotificationProtocolProfile" type="xs:anyURI"/>  
507       </xs:sequence>  
508     </xs:extension>  
509   </xs:complexContent>  
510 </xs:complexType>  
511  
512  
513
```

514 2.1.5.3. EntityDescriptor Element

515 The element EntityDescriptor is used to contain one or more descriptor types for a given organization. Publishers
516 MUST NOT convey metadata for other unaffiliated organizations within this node. Representations of multiple,
517 unaffiliated providers within a single instance document MUST be done using the plural node form EntitiesDescriptor
518 ([Section 2.1.5.4](#)) instead. Publishers MUST publish all relevant roles in this single document, or indirectly through
519 AdditionalMetaLocation.

520 Note that it is possible for a single organization to be represented by more than one providerID, by indicating
521 different providerID attributes for each entity descriptor, and publishing the document as EntitiesDescriptor.

522 EntityDescriptor may contain **either**: zero or more IDPDescriptors and zero or more SPDescriptors, **or**
523 exactly one AffiliationDescriptor followed by any of: ContactPerson, Organization, ds:Signature,
524 and Extension.

525 Attributes for EntityDescriptor:

- 526 • providerID [Required]: the providerID of the entity whose metadata is represented by all descendants of
527 EntityDescriptor
- 528 • id [Optional] fragment identifier which is required if ds:Signature is present.

- 529 • `validUntil` [Optional] The expiration dateTime of the metadata.
- 530 • `cacheDuration` [Optional] The cache duration period for the metadata.

531 Elements contained in `EntityDescriptor`:

- 532 • `IDPDescriptor` Metadata describing an entity acting as an Identity Provider.
- 533 • `SPDescriptor` Metadata describing an entity acting as a Service Provider.
- 534 • `AffiliationDescriptor` Metadata describing a set of entities identified by their respective `providerIDs`
535 collectively referred to as an affiliation [Section 2.1.5.5](#)
- 536 • `ContactPerson` [Optional, 0-1] Contact information for the overall entity (see [Section 2.1.4.2](#)).
- 537 • `Organization` [Optional, 0-1] Organizational information about the entity (see [Section 2.1.4.1](#)).
- 538 • `Extension` [Optional, 0-1] provides extension point for additional entity metadata
- 539 • `ds:Signature` [Optional, 0-1] An XML Signature on the entire entity metadata instance.

540 The schema fragment for `entityDescriptorType`:

```
541 <xs:complexType name="entityDescriptorType">
542   <xs:sequence>
543     <xs:choice>
544       <xs:group ref="providerGroup"/>
545       <xs:element name="AffiliationDescriptor" type="affiliationDescriptorType"/>
546     </xs:choice>
547     <xs:element minOccurs="0" name="ContactPerson" type="contactType"/>
548     <xs:element minOccurs="0" name="Organization" type="organizationType"/>
549     <xs:element minOccurs="0" ref="Extension"/>
550     <xs:element minOccurs="0" ref="ds:Signature"/>
551   </xs:sequence>
552   <xs:attribute ref="providerID" use="required"/>
553   <xs:attribute name="id" type="xs:ID" use="optional"/>
554   <xs:attribute ref="validUntil" use="optional"/>
555   <xs:attribute ref="cacheDuration" use="optional"/>
556 </xs:complexType>
557
558 <xs:element name="EntityDescriptor" type="entityDescriptorType"/>
559
560 <xs:group name="providerGroup">
561   <xs:sequence>
562     <xs:element maxOccurs="unbounded" minOccurs="0" name="IDPDescriptor" type="IDPDescriptorType"/>
563     <xs:element maxOccurs="unbounded" minOccurs="0" name="SPDescriptor" type="SPDescriptorType"/>
564   </xs:sequence>
565 </xs:group>
```

570 2.1.5.4. EntitiesDescriptor

571 The element `EntitiesDescriptor` describes more than one organization in a single instance document. It consists
572 of 2 or more `EntityDescriptors`.

573 The schema fragment for `EntitiesDescriptor` element:

```
574 <xs:element name="EntitiesDescriptor" type="entitiesDescriptorType"/>
575
```

```
576 <xs:complexType name="entitiesDescriptorType">
577 <xs:sequence>
578 <xs:element maxOccurs="unbounded" minOccurs="2" ref="EntityDescriptor"/>
579 </xs:sequence>
580 </xs:complexType>
581
```

582 2.1.5.5. AffiliationDescriptor

583 The `AffiliationDescriptor` element describes a group of entities, identified collectively by `affiliationID`,
584 as an enumeration of `providerID`'s. The uniqueness constraints for `providerID` also apply for `affiliationID`,
585 such that it **MUST** be unique across all Liberty entities with which the affiliation expects to interact, including other
586 affiliations and providers therefore, it **MUST NOT** be the `providerID` of any of the members of the affiliation, and
587 **SHOULD** be unique across the set of `providerID`'s and `affiliationID`'s with which the affiliation expects to
588 interact. It is the responsibility of the entity represented by `affiliationOwnerID` to administer this identifier, and
589 thus, its members.

590 `AffiliationDescriptor` element contains the following attributes:

- 591 • `affiliationID` [Required] the identifier for the affiliation (with identical structure constraints as `providerID`.
592 See [Section 2.1.2](#))
- 593 • `affiliationOwnerID` [Required] the `providerID` of the owner or parent operator of the affiliation, from which,
594 additional metadata may be derived
- 595 • `validUntil` [Optional] The expiration dateTime of the metadata.
- 596 • `cacheDuration` [Optional] The cache duration period for the metadata.
- 597 • `id` [Optional].

598 and the following elements:

- 599 • `AffiliateMember` [Required, 1-many] One or more providers who are members of the affiliation. The value
600 **MUST** be a `providerID` who's metadata **MUST** be obtained via methods described in [Section 3](#)
- 601 • `Extension` [Optional, 0-1] provides an extension point to convey additional metadata concerning the affiliation
- 602 • `KeyDescriptor` [Optional, 0-many] Zero or more public key material reference that is the property of the
603 affiliation. This keying material **SHOULD** be separate from the keying material of the `providerID` who may
604 be referenced as the `affiliateOwnerID` and **MAY** be used for encryption or signing, as indicated by its
605 corresponding `use` attribute.
- 606 • `ds:Signature` [Optional, 0-1] An XML Signature of the metadata node `AffiliationDescriptor`.

607 The schema fragment for the AffiliationDescriptor element:

```

608
609 <xs:complexType name="affiliationDescriptorType">
610 <xs:sequence>
611 <xs:element maxOccurs="unbounded" name="AffiliateMember" type="entityIDType"/>
612 <xs:element minOccurs="0" ref="Extension"/>
613 <xs:element maxOccurs="unbounded" minOccurs="0" name="KeyDescriptor" type="keyDescriptorT
614 ype"/>
615 <xs:element minOccurs="0" ref="ds:Signature"/>
616 </xs:sequence>
617 <xs:attribute name="affiliationID" type="entityIDType" use="required"/>
618 <xs:attribute name="affiliationOwnerID" type="entityIDType" use="required"/>
619 <xs:attribute ref="validUntil" use="optional"/>
620 <xs:attribute ref="cacheDuration" use="optional"/>
621 <xs:attribute name="id" type="xs:ID" use="optional"/>
622 </xs:complexType>
623

```

624 2.1.6. WSDL Usage

625 A WSDL [WSDLv1.1] document MAY be used to describe the web services available at the location SoapEndpoint
626 in addition to the metadata itself. Following is the abstract WSDL describing the ID-FF services:

```

627 <?xml version="1.0" encoding="UTF-8"?>
628 <definitions name="IDFF" targetNamespace="urn:liberty:md:IDFF:wsdl" xmlns="http://schemas.xmlsoap.org/wsdl
629 /"
630     xmlns:iff="urn:liberty:iff:2003-08" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
631
632     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
633
634     xmlns:tns="urn:liberty:md:IDFF:wsdl" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
635
636     <import location="oasis-sstc-saml-schema-assertion-1.1.xsd" namespace="urn:oasis:names:tc:SAML:1.0:
637 assertion"/>
638 <import location="liberty-idff-protocols-schema-v1.2.xsd" namespace="urn:liberty:iff:2003-08"/>
639 <import location="oasis-sstc-saml-schema-protocol-1.1.xsd" namespace="urn:oasis:names:tc:
640 SAML:1.0:protocol"/>
641 <types/>
642 <!--annotation>
643 <documentation>WSDL fom Metadata description and discovery protocols</documentation>
644 <documentation>The source code in this WSDL file was excerpted verbatim from:
645
646 Liberty Metadata Description and Discovery Specification
647 Version 1.0
648 12th November 2003
649
650 Copyright (c) 2003 Liberty Alliance participants, see
651 https://www.projectliberty.org/specs/idff_copyrights.html
652
653 </documentation>
654 </annotation-->
655 <message name="authenticationResponse">
656 <part name="body" type="iff:AuthnResponseEnvelope"/>
657 </message>
658 <message name="NameIdentifierMappingResponse">
659 <part name="body" type="saml:Assertion"/>
660 </message>
661 <message name="artifactResponse">
662 <part name="body" type="samlp:Response"/>
663 </message>
664 <message name="LogoutRequest">
665 <part name="body" type="iff:LogoutRequest"/>
666 </message>
667 <message name="registerNameIdentifierRequest">
668

```

```
669     <part name="body" type="iff:RegisterNameIdentifierRequest" />
670 </message>
671 <message name="LogoutResponse">
672   <part name="body" type="iff:LogoutResponse" />
673 </message>
674 <message name="NameIdentifierMappingRequest">
675   <part name="body" type="samlp:Request" />
676 </message>
677 <message name="FederationTermination">
678   <part name="body" type="iff:FederationTerminationNotification" />
679 </message>
680 <message name="authenticationRequest">
681   <part name="body" type="iff:AuthnRequest" />
682 </message>
683 <message name="artifactRequest">
684   <part name="body" type="samlp:Request" />
685 </message>
686 <message name="registerNameIdentifierResponse">
687   <part name="body" type="iff:RegisterNameIdentifierResponse" />
688 </message>
689 <portType name="IDPPort">
690   <operation name="registerNameIdentifier">
691     <input message="tns:registerNameIdentifierRequest" />
692     <output message="tns:registerNameIdentifierResponse" />
693   </operation>
694   <operation name="FederationTermination">
695     <input message="tns:FederationTermination" />
696   </operation>
697   <operation name="Logout">
698     <input message="tns:LogoutRequest" />
699     <output message="tns:authenticationResponse" />
700   </operation>
701   <operation name="NameIdentifierMapping">
702     <input message="tns:NameIdentifierMappingRequest" />
703     <output message="tns:NameIdentifierMappingResponse" />
704   </operation>
705   <operation name="authentication">
706     <input message="tns:authenticationRequest" />
707     <output message="tns:authenticationResponse" />
708   </operation>
709   <operation name="artifact">
710     <input message="tns:artifactRequest" />
711     <output message="tns:artifactResponse" />
712   </operation>
713 </portType>
714 <service name="IDFF">
715   <port binding="tns:IDPBinding" name="IDFFPort">
716     <soap:address location="http://localhost:8000/ccx/IDFF" />
717   </port>
718 </service>
719 </definitions>
720
721
```

722 3. Publishing the Metadata

723 Two mechanisms are provided for entities to publish metadata document locations: via the DNS and via a "well-
724 known-location" by directly dereferencing the entities' `providerIDs`.

725 When retrieval requires network transport of the document, in both cases above the transport SHOULD be protected
726 with *TLS/SSL* [RFC2246] as amended by [RFC3546]. This will ensure the integrity of the metadata document,
727 as among other information within the document, trust establishment may be based in part on information provided
728 within the metadata. Relying parties of this metadata are RECOMMENDED to authenticate the server via *TLS/SSL*
729 validation procedures.

730 Trust establishment of the Metadata will be based on one or more of the following: DNS signatures (RECOM-
731 MENDED); TLS server authentication (RECOMMENDED); and Metadata `ds:Signature` (STRONGLY RECOM-
732 MENDED) evaluations. Publishers MAY implement additional trust mechanisms, in conjunction with the required
733 suggested server authentication. Additional trust metadata content, if supplied, MUST be placed in the extension
734 points provided.

735 3.1. Instance Publication Forms

736 If separate documents are used, references to each MUST be made, either through one or more additional PID2MD
737 NAPTR record(s), or by using the `AdditionalMetaLocation` element within a document which has an associated
738 NAPTR RR, or which is situated at the "well-known location" (see Section 3.3).

739 3.2. Using the DNS to Publish Metadata Location(s)

740 In order to ensure that all providers have accessible metadata locations, entities are STRONGLY RECOMMENDED
741 to publish their metadata document locations in a zone of their corresponding DNS [RFC1034]. As *providerIDs* are
742 flexible identifiers, publication and resolution is determined by an entity's URI scheme and fully qualified name part
743 of the identifier. URI locations for metadata will subsequently be derived through queries of the NAPTR Resource
744 Record (RR) as defined in [RFC2915] and [RFC3403].

745 It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535] such that
746 relying parties may establish the validity of the published location and authority of the zone, and integrity of the DNS
747 response. If DNS zone signatures are present, relying parties MUST properly validate the signature.

748 3.2.1. Publication of Metadata Locations

749 This specification makes use of the resource record described in [RFC2915] and [RFC3403]. Familiarity with these
750 documents is encouraged.

751 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of infor-
752 mation based on an application-specific input string and the application of well known rules to transform that string
753 until a terminal condition is reached requiring a look-up into an application-specific defined database or execution of
754 a URL based on the rules defined by the application. DDDS defines a specific type of DNS Resource Record, NAPTR
755 records, for the storage of information in the DNS necessary to apply DDDS rules.

756 Entities MAY publish separate URL's when the metadata documents need to be distributed, or when different metadata
757 documents are required due to multiple Authentication Domain memberships which require separate keying material,
758 or when service interfaces require separate metadata declarations. This may be accomplished through the use of the
759 optional `AdditionalMetaLocation` attribute in the core or other subordinate metadata document, or through the
760 `regexp` facility and multiple service definition fields in the NAPTR resource record itself.

761 If `providerID` is a URN, resolution of the `MetadataLocation` proceeds as specified in [RFC3404]. Otherwise, the
762 resolution of the metadata location proceeds as specified in this specification.

763 Following are the application-specific descriptions for the DDDS application for the Liberty Metadata resolution
764 protocols.

765 **3.2.1.1. Application Unique String**

766 Liberty metadata resolution shall begin with the application unique string of `providerID`.

767 **3.2.1.2. First Well Known Rule**

768 The "first well-known-rule" for processing Liberty Alliance Metadata resolution is to parse the `providerID` URI and
769 extract the fully qualified domain name (subexpression 3) as described in section [Section 4.1.1](#).

770 **3.2.1.3. The Order Field**

771 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY provide
772 multiple NAPTR resource record's which MUST be processed by the resolver application in the order indicated by
773 this field.

774 **3.2.1.4. The Preference Field**

775 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving application.
776 The resolving application MAY ignore this order, in cases where the service field value does not meet the resolver's
777 requirements (e.g.: the resource record returns a protocol the application does not support).

778 **3.2.1.5. The Flag Field**

779 Liberty Metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying additional
780 resource record's are to be processed). The "U" flag indicates that the output of the rule is a URI.

781 **3.2.1.6. The Service Field**

782 The Liberty specific service fields shall include:

```
783  
784 servicefield = 1("PID2U" / "NID2U") "+" proto [*( ":" class ) *( ":" ser vicetype)]  
785 proto = 1("https" / "uddi ")  
786 class = 1[ "entity" / "entitygroup" ]  
787 servicetype = 1(si / "sp" / "id p" / " authn" / alphanum )  
788 si = "si" [ ":" alphanum ] [ ":" endpoint"]  
789 alphanum = 1*32(ALPHA / DIGIT)
```

790 where:

- 791 • `PID2U` resolves a `providerID` identifier to metadata URL
- 792 • `NID2U` resolves a `nameIdentifier` (principal) metadata URL
- 793 • `proto` describes the retrieval protocol (https or uddi). In the case of UDDI, the resulting URI will be an http(s)
794 URI referencing a WSDL document.
- 795 • `class` identifies which indicates whether the referenced metadata document describes a single provider, or
796 multiple. In the latter case, the referenced document MUST contain the entity defined by `providerID` as a
797 member of a group of entities within the document itself.

798 • `servicetype` allows a publishers to publish service provider, identity provider and service instance metadata
799 locations as separate documents. Resolvers who encounter multiple `servicetype` declarations will dereference the
800 appropriate URI, depending on which service type required for an operation (e.g.: a provider operating both and
801 IdP and an SP service, may publish SP and IdP metadata at different locations).

802 • the `si` component (with optional endpoint component) allows the publisher to either directly publish the metadata
803 for a service instance, or by articulating the soap endpoint (using `endpoint`

804 For example:

805 • `PID2U+https:entity` - represents the complete entity metadata document via the `https` protocol

806 • `PID2U+https:entity:si:pip` - returns the PIP metadata URL for the entity described by `providerID` via the `https`
807 protocol profile

808 • `PID2U+uddi:entity:si:foo` - returns the WSDL document location which describes a service instance "foo"

809 • `PID2U+https:entitygroup:idp` - returns the metadata for a group of entities, of which `providerID` is a member. the
810 referenced document describes (one or more) IdPs in the group

811 • `NID2U+https:idp` - returns an IdP `providerIDs`, who can provider authentication services for a principal

812 • `NID2U+https:authn` - returns a URL to attempt to authenticate the principal against

813 3.2.1.7. The Regex and Replacement Fields

814 The expected output after processing the application-unique string through the regex MUST be a valid `https` URL or
815 UDDI node (`http` references `wsdl` document) address.

816 3.2.2. NAPTR Examples

817 3.2.2.1. Provider Metadata NAPTR Examples

818 Entities publish metadata URLs in the following manner:

```
819 $ORIGIN provider.biz
820
821 ;; order pref f service regexp or replacement
822
823 IN NAPTR 100 10 "U" PID2U+https:entity "!^.*$!https://host.provider.biz/some/directory/trust_
824 .xml!" ""
825 IN NAPTR 110 10 "U" PID2U+https:entity:trust "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!"
826 ""
827 IN NAPTR 125 10 "U" PID2U+https:
828 IN NAPTR 110 10 "U" PID2U+uddi:entity "!^.*$ !https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

830 3.2.2.2. Name Identifier Examples

831 Principals employer example.int operates an IdP which may be used by a office supply company to authenticate
832 authorized buyers. The supplier takes users email address `buyer@example.int` as input to the resolution process,
833 and parses the email address to extract the FQDN (`example.int`). The employer publishes the following NAPTR in
834 `example.int`:

```
835 $ORIGIN
836 example.int.
```

```
839
840     IN NAPTR 100 10 "U" NID2U+https:authn "!^([ ^@]+)@ (.*)$!https://serv.example.int:8000/cgi-bin/getmd?\  
841 1!" ""
842     IN NAP TR 100 10 "U" NID2U+https:idp "!^([ ^@]+)@(.*)$!https://auth.example.int/app/auth?1" ""
843
```

844 3.3. Publication via Well-Known Location

845 Entities MAY publish their metadata documents at a well known location. The core metadata document location in
846 this profile simply involves directly dereferencing the providerID and obtaining the document directly (or through
847 schema-specific means of indirection)

848 For well known location documents, the XML document MUST describe the metadata for the **providerID** entity only.
849 If other entities need to be described, the **AdditionalMetaLocation** MUST be used. Thus the **entitiesDescriptor**
850 MUST NOT be used in documents published at a well know location, since entities as a group, are not defined by such
851 an identifier.

852 4. Metadata Resolution and Retrieval

853 Metadata publication is provided in two fashions: via a "well-known-location" and via queries on the DNS. Both
854 mechanisms depend upon processing of the `providerID` element (see [Section 3]), which is the primary identifier
855 for Liberty-enabled entities. Consumers are **STRONGLY RECOMMENDED** to attempt DNS-based resolution prior
856 to performing a direct dereferencing of a `providerID`.

857 The `providerID` is defined as a restricted form of `anyURI` Section 2.1.2; therefore, it shall be parsed as in Section 4.1.1
858 for these resolution profiles.

859 4.1. Resolving Locations and Retrieving Metadata

860 The summarized steps for retrieving metadata from a given `providerID` is as follows:

- 861 • If the `providerID` is a URN, proceed with the resolution steps as defined in [RFC3404]
- 862 • parse the `providerID` to obtain the *FQDN*
- 863 • query the DNS for NAPTR resource records of the `domain` name iteratively until a terminal resource record is
864 returned
865 (optionally, or if DNS-based resolution fails) attempt locating the metadata document(s) via the *well known*
866 *location* profile by directly dereferencing the `providerID` (end if a document was located, validated and fulfills
867 the metadata requirements for the present operations).
- 868 • identify which resource record to use based on the service fields, then order fields, then preference fields of the
869 result set.
- 870 • obtain the document(s) at the provided location(s) as required by the application

871 4.1.1. Parsing the ProviderID

872 To initiate the resolution of the location of the target metadata elements, it will be necessary in some cases to
873 decompose the `ProviderID` (expressed as a URI) into one or more atomic elements.

874 The following regular expression should be used when initiating the decomposition process:

```
875 ^([\^:/?#\+:\ ])?/*([\^:/?#\+:\ ]*)?(((([\^:/?#\+:\ ])*\.)*([\^:/?#\+:\ ]+))\.[([\^:/?#\+:\ ]+)):([\d+)?([\^?#\ ]*)(\?[^\#]*)?(\#\.*)?&
```

| | | | | | | | | |
|-----|----|---|----|----|---|---|---|----|
| 876 | 1 | 2 | 34 | 56 | 7 | 8 | 9 | 10 |
| 877 | | | | | | | | |
| 878 | | | | | | | | |
| 879 | 11 | | | | | | | |

880 Subexpression 3 **MUST** result in a Fully Qualified Domain Name (FQDN), which will be the basis for retrieving
881 metadata locations from this zone.

882 4.1.2. Obtaining Metadata via the DNS

883 Upon completion of the parsing of the `providerID`, the application then performs a DNS query for the resulting
884 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses. Applications **MAY**
885 exclude from the result set any service definitions which do not concern the present request operations. Should the
886 DNS not produce a valid response, the consumer **MUST ALWAYS** attempt direct dereferencing of the `providerID`.

887 Resolving applications **MUST** subsequently order the result set according to the order field, and **MAY** order the result
888 set based on the preference set. Resolvers are **NOT REQUIRED** to follow the ordering of the preferences field.

889 The resulting NAPTR resource record(s) are operated on iteratively (based on the order flag) until a terminal NAPTR
890 resource record is reached.

891 The result will be a well formed, fully qualified URL, which will then be used to retrieve the metadata document.

892 **4.1.2.1. Post Processing Operations**

893 When service specific metadata is sought, resolvers MAY filter the NAPTR result set based on more specific resource
894 records with service identifiers which match the service(s) sought.

895 **4.1.3. Obtaining Metadata via the "Well-Known Location Method"**

896 Consumers of published metadata MAY attempt retrieval via the well-known-location method by directly dereferenc-
897 ing the providerID. Other forms of well-known location MAY be agreed upon by a group of Liberty entities, however,
898 it is STRONGLY SUGGESTED that publication in the DNS be employed as well, to allow for interactions with other
899 Liberty implementations. The resulting XML document MUST describe the metadata for the providerID entity
900 only. If other entities need to be described, the AdditionalMetaLocation MUST be used. There may be only
901 one location, although this document MAY point to other document locations using the AdditionalMetaLocation
902 element.

903 **5. Post Processing of the Metadata Document**

904 **5.1. Processing of ds:Signature and General Trust Processing**

905 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and for the trust
906 ascribed to the entity described by such metadata:

- 907 • Trust derived from the signature of the zone from which the metadata location URI was resolved, ensuring accuracy
908 of the metadata document location(s)
- 909 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of the XML
910 document
- 911 • Trust derived from the SSL/TLS negotiation of the metadata delivery URI, ensuring the identity of the publisher
912 of the metadata.

913 Post processing of the metadata document **MUST** include the signature processing at the XML-document level
914 and **MAY** include one of the other two processes. Specifically, the relying party **MAY** choose to trust any of the
915 cited authorities in the resolution and parsing process. Publishers of metadata **MUST** employ a document-integrity
916 mechanism and **MAY** employ any of the other two processing profiles to establish trust of the subject of the metadata
917 document, governed by implementation policies.

918 **5.1.1. Processing Signed DNS Zones**

919 Verification of zone signature **SHOULD** be processed, if present, as described in [\[RFC2535\]](#)

920 **5.1.2. Processing Signed Documents and Fragments**

921 Published metadata documents **SHOULD** be signed, as described in [\[XMLDsig\]](#), either by a certificate issued to the
922 subject of the document, or by another trusted party. Publishers **MAY** consider signatures of other parties as a means
923 of trust conveyance.

924 Consumers **MUST** validate signatures, when present, on the metadata document on initial retrieval as described by
925 [\[XMLDsig\]](#).

926 **5.1.3. Processing Server Authentication in Metadata Retrieval via TLS/SSL**

927 It is **STRONGLY RECOMMENDED** that publishers implement TLS URL's; therefore, consumers **SHOULD** consider
928 the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not always be located in
929 the domain of the provider of the subject of the metadata document; therefore, consumers **SHOULD NOT** expect
930 certificates whose subject is the provider, as it may be hosted at another trusted party.

931 As the basis of this trust may not be available against a cached document, other mechanisms **SHOULD** be used under
932 such circumstances.

933 **5.2. Metadata Location and Document Caching**

934 Location caching based on DNS profiles **MUST NOT** exceed the TTL of the DNS zone from which the location was
935 derived. Resolvers **MUST** obtain a fresh copy of the Metadata location upon reaching the expiration of the TTL of the
936 zone.

937 A publishers of Metadata documents should carefully consider the TTL of the zone when making updates to its
938 metadata document location. Should such a location change occur, a publisher **MUST** either keep the document at
939 both the old and new location until all conforming resolvers are certain to have the updated location (e.g.: time of zone
940 change + TTL), or provide an HTTP `Redirect` [\[RFC2616\]](#) to the new location.

941 Document caching MUST NOT exceed the `validUntil` attribute of the subject element(s) and the `cacheDuration`
942 attribute. If fragments have parents which contain caching policies, the parent fragment ALWAYS takes precedence.

943 To properly process the `cacheDuration` attributes on fragments and documents consumers MUST retain the
944 `dateTime` when the document was retrieved.

945 When a document or fragment has expired the consumer MUST retrieve a fresh copy, which may require a refresh of
946 the document location(s). Consumers SHOULD process document cache processing according to [\[RFC2616\]](#) section
947 13, and MAY request the Last-Modified `dateTime` from the HTTPS server. Publishers SHOULD ensure acceptable
948 cache processing as described in [\[RFC2616\]](#) (Section 10.3.5 304 Not Modified).

949 **5.3. Handling of HTTPS Redirects**

950 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, or 307 Temporary Redirect) [\[RFC2616\]](#), and user
951 agents MUST follow the specified URL in the Redirect response. Redirects SHOULD be to a TLS/SSL protected
952 resource, and SHOULD be of the same protocol as the initial request.

953 **6. Security Considerations**

954 **6.1. Trust Establishment**

955 Cryptographic signatures are used to establish identity and tamper evidence in several locations within the metadata
956 specification. While valid signatures convey some level of trust in the resulting document, extreme care should be
957 taken as to the validity of the URIs described within the document itself. Relying parties should carefully inspect
958 agreements and statements made by the signing authorities of the subject certificates or keys.

959 7. Metadata XSD

```
960 <?xml version="1.0" encoding="UTF-8"?>
961 <xs:schema targetNamespace="urn:liberty:metadata:2003-08" xmlns="urn:liberty:metadata:2003-08"
962 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:saml="urn:oasis:names:tc:SAML:1.0:as
963 ertion" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attrib
964 uteFormDefault="unqualified" version="1.0">
965   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="http://www.w3.org/TR/xmldsi
966   g-core/xmldsig-core-schema.xsd"/>
967   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="oasis-sstc-saml-schema-asserti
968   on-1.1.xsd"/>
969   <xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http://www.w3.org/2001/xm
970   l.xsd"/>
971   <xs:include schemaLocation="liberty-utility-v1.0.xsd"/>
972   <xs:annotation>
973     <xs:documentation>XML Schema for Metadata description and discovery protocols</xs:docum
974     entation>
975     <xs:documentation>The source code in this XSD file was excerpted verbatim from:
976     Liberty Metadata Description and Discovery Specification
977     Version 1.0
978     12th November 2003
979     Copyright (c) 2003 Liberty Alliance participants, see
980     https://www.projectliberty.org/specs/idff_copyrights.html
981   </xs:documentation>
982   </xs:annotation>
983   <xs:simpleType name="entityIDType">
984     <xs:restriction base="xs:anyURI">
985       <xs:maxLength value="1024" id="maxlengthid"/>
986     </xs:restriction>
987   </xs:simpleType>
988   <xs:attribute name="libertyPrincipalIdentifier" type="entityIDType"/>
989   <xs:attribute name="providerID" type="entityIDType"/>
990   <xs:attribute name="validUntil" type="xs:dateTime"/>
991   <xs:attribute name="cacheDuration" type="xs:duration"/>
992   <xs:complexType name="additionalMetadataLocationType">
993     <xs:simpleContent>
994       <xs:extension base="xs:anyURI">
995         <xs:attribute name="namespace" type="xs:anyURI"/>
996       </xs:extension>
997     </xs:simpleContent>
1000   </xs:complexType>
1001   <xs:complexType name="organizationType">
1002     <xs:sequence>
1003       <xs:element name="OrganizationName" type="organizationNameType" maxOccurs="unbounded"/>
1004       <xs:element name="OrganizationDisplayName" type="organizationDisplayNameType" max
1005       Occurs="unbounded"/>
1006       <xs:element name="OrganizationURL" type="localizedURIType" maxOccurs="unbounded"/>
1007       <xs:element ref="Extension" minOccurs="0"/>
1008     </xs:sequence>
1009   </xs:complexType>
1010   <xs:complexType name="organizationNameType">
1011     <xs:simpleContent>
1012       <xs:extension base="xs:string">
1013         <xs:attribute ref="xml:lang"/>
1014       </xs:extension>
1015     </xs:simpleContent>
1016   </xs:complexType>
1017   <xs:complexType name="organizationDisplayNameType">
1018     <xs:simpleContent>
1019       <xs:extension base="xs:string">
1020         <xs:attribute ref="xml:lang" use="required"/>
1021       </xs:extension>
1022     </xs:simpleContent>
1023   </xs:complexType>
```

```

1025 <xs:complexType name="localizedURIType">
1026   <xs:simpleContent>
1027     <xs:extension base="xs:anyURI">
1028       <xs:attribute ref="xml:lang" use="required"/>
1029     </xs:extension>
1030   </xs:simpleContent>
1031 </xs:complexType>
1032 <xs:complexType name="contactType">
1033   <xs:sequence>
1034     <xs:element name="Company" type="xs:string" minOccurs="0"/>
1035     <xs:element name="GivenName" type="xs:string" minOccurs="0"/>
1036     <xs:element name="SurName" type="xs:string" minOccurs="0"/>
1037     <xs:element name="EmailAddress" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
1038     <xs:element name="TelephoneNumber" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
1039     <xs:element ref="Extension" minOccurs="0"/>
1040   </xs:sequence>
1041   <xs:attribute ref="libertyPrincipalIdentifier" use="optional"/>
1042   <xs:attribute name="contactType" type="attr.contactType" use="required"/>
1043 </xs:complexType>
1044 <xs:simpleType name="attr.contactType">
1045   <xs:restriction base="xs:string">
1046     <xs:enumeration value="technical"/>
1047     <xs:enumeration value="administrative"/>
1048     <xs:enumeration value="billing"/>
1049     <xs:enumeration value="other"/>
1050   </xs:restriction>
1051 </xs:simpleType>
1052 <xs:simpleType name="keyTypes">
1053   <xs:restriction base="xs:string">
1054     <xs:enumeration value="encryption"/>
1055     <xs:enumeration value="signing"/>
1056   </xs:restriction>
1057 </xs:simpleType>
1058 <xs:complexType name="providerDescriptorType">
1059   <xs:sequence>
1060     <xs:element name="KeyDescriptor" type="keyDescriptorType" minOccurs="0" maxOccurs="unbounded"/>
1061
1062     <xs:element name="SoapEndpoint" type="xs:anyURI" minOccurs="0"/>
1063     <xs:element name="SingleLogoutServiceURL" type="xs:anyURI" minOccurs="0"/>
1064     <xs:element name="SingleLogoutServiceReturnURL" type="xs:anyURI" minOccurs="0"/>
1065     <xs:element name="FederationTerminationServiceURL" type="xs:anyURI" minOccurs="0"/>
1066     <xs:element name="FederationTerminationServiceReturnURL" type="xs:anyURI" minOccurs="0"/>
1067     <xs:element name="FederationTerminationNotificationProtocolProfile" type="xs:anyURI" ↵
1068 minOccurs="0" maxOccurs="unbounded"/>
1069     <xs:element name="SingleLogoutProtocolProfile" type="xs:anyURI" minOccurs="0" maxOccurs="unbound
1070 ded"/>
1071     <xs:element name="RegisterNameIdentifierProtocolProfile" type="xs:anyURI" minOccurs="0" ↵
1072 maxOccurs="unbounded"/>
1073     <xs:element name="RegisterNameIdentifierServiceURL" type="xs:anyURI" minOccurs="0"/>
1074     <xs:element name="RegisterNameIdentifierServiceReturnURL" type="xs:anyURI" minOccurs="0"/>
1075     <xs:element name="RelationshipTerminationNotificationProtocolProfile" type="xs:anyURI" minOcc
1076 urs="0" maxOccurs="unbounded"/>
1077     <xs:element name="NameIdentifierMappingBinding" type="saml:AuthorityBindingType" ↵
1078 minOccurs="0" maxOccurs="unbounded"/>
1079     <xs:element name="Organization" type="organizationType" minOccurs="0"/>
1080     <xs:element name="ContactPerson" type="contactType" minOccurs="0" maxOccurs="unbounded"/>
1081     <xs:element name="AdditionalMetaLocation" type="additionalMetadataLocationType" ↵
1082 minOccurs="0" maxOccurs="unbounded"/>
1083     <xs:element ref="Extension" minOccurs="0"/>
1084     <xs:element ref="ds:Signature" minOccurs="0"/>
1085   </xs:sequence>
1086   <!--xs:attribute ref="providerID" use="required"/-->
1087   <xs:attribute name="protocolSupportEnumeration" type="xs:NMTOKENS" use="required"/>
1088   <xs:attribute name="id" type="xs:ID" use="optional"/>
1089   <xs:attribute ref="validUntil" use="optional"/>
1090   <xs:attribute ref="cacheDuration" use="optional"/>
1091 </xs:complexType>

```

```

1092 <!-- added-->
1093 <xs:element name="KeyDescriptor" type="keyDescriptorType"/>
1094 <xs:complexType name="keyDescriptorType">
1095   <xs:sequence>
1096     <xs:element name="EncryptionMethod" type="xs:anyURI" minOccurs="0"/>
1097     <xs:element name="KeySize" type="xs:integer" minOccurs="0"/>
1098     <xs:element ref="ds:KeyInfo" minOccurs="0"/>
1099     <xs:element ref="Extension" minOccurs="0"/>
1100   </xs:sequence>
1101   <xs:attribute name="use" type="keyTypes" use="optional"/>
1102 </xs:complexType>
1103 <!-- -->
1104 <xs:element name="EntityDescriptor" type="entityDescriptorType"/>
1105 <xs:group name="providerGroup">
1106   <xs:sequence>
1107     <xs:element name="IDPDescriptor" type="IDPDescriptorType" minOccurs="0" maxOccurs="unbound
1108 ded"/>
1109     <xs:element name="SPDescriptor" type="SPDescriptorType" minOccurs="0" maxOccurs="unbounded"/>
1110   </xs:sequence>
1111 </xs:group>
1112 <xs:complexType name="entityDescriptorType">
1113   <xs:sequence>
1114     <xs:choice>
1115       <xs:group ref="providerGroup"/>
1116       <xs:element name="AffiliationDescriptor" type="affiliationDescriptorType"/>
1117     </xs:choice>
1118     <xs:element name="ContactPerson" type="contactType" minOccurs="0"/>
1119     <xs:element name="Organization" type="organizationType" minOccurs="0"/>
1120     <xs:element ref="Extension" minOccurs="0"/>
1121     <xs:element ref="ds:Signature" minOccurs="0"/>
1122   </xs:sequence>
1123   <xs:attribute ref="providerID" use="required"/>
1124   <xs:attribute name="id" type="xs:ID" use="optional"/>
1125   <xs:attribute ref="validUntil" use="optional"/>
1126   <xs:attribute ref="cacheDuration" use="optional"/>
1127 </xs:complexType>
1128 <xs:complexType name="SPDescriptorType">
1129   <xs:complexContent>
1130     <xs:extension base="providerDescriptorType">
1131       <xs:sequence>
1132         <xs:element name="AssertionConsumerServiceURL" maxOccurs="unbounded">
1133           <xs:complexType>
1134             <xs:simpleContent>
1135               <xs:extension base="xs:anyURI">
1136                 <xs:attribute name="id" type="xs:ID" use="required"/>
1137                 <xs:attribute name="isDefault" type="xs:boolean" default="false"/>
1138               </xs:extension>
1139             </xs:simpleContent>
1140           </xs:complexType>
1141         </xs:element>
1142         <xs:element name="AuthnRequestsSigned" type="xs:boolean"/>
1143       </xs:sequence>
1144     </xs:extension>
1145   </xs:complexContent>
1146 </xs:complexType>
1147 <xs:complexType name="IDPDescriptorType">
1148   <xs:complexContent>
1149     <xs:extension base="providerDescriptorType">
1150       <xs:sequence>
1151         <xs:element name="SingleSignOnServiceURL" type="xs:anyURI"/>
1152         <xs:element name="SingleSignOnProtocolProfile" type="xs:anyURI" maxOccurs="unbounded"/>
1153         <xs:element name="IntroductionNotificationProtocolProfile" type="xs:anyURI" minOccu
1154 rs="0" maxOccurs="unbounded"/>
1155       </xs:sequence>
1156     </xs:extension>
1157   </xs:complexContent>
1158 </xs:complexType>

```

```
1159 <xs:element name="EntitiesDescriptor" type="entitiesDescriptorType"/>
1160 <xs:complexType name="entitiesDescriptorType">
1161   <xs:sequence>
1162     <xs:element ref="EntityDescriptor" minOccurs="2" maxOccurs="unbounded"/>
1163   </xs:sequence>
1164 </xs:complexType>
1165 <xs:complexType name="affiliationDescriptorType">
1166   <xs:sequence>
1167     <xs:element name="AffiliateMember" type="entityIDType" maxOccurs="unbounded"/>
1168     <xs:element ref="Extension" minOccurs="0"/>
1169     <xs:element name="KeyDescriptor" type="keyDescriptorType" minOccurs="0" maxOccurs="unb
1170 ounded"/>
1171     <xs:element ref="ds:Signature" minOccurs="0"/>
1172   </xs:sequence>
1173   <xs:attribute name="affiliationID" type="entityIDType" use="required"/>
1174   <xs:attribute name="affiliationOwnerID" type="entityIDType" use="required"/>
1175   <xs:attribute ref="validUntil" use="optional"/>
1176   <xs:attribute ref="cacheDuration" use="optional"/>
1177   <xs:attribute name="id" type="xs:ID" use="optional"/>
1178 </xs:complexType>
1179 </xs:schema>
1180
1181
```

References

1182

Normative

1183

- 1184 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
1185 1.2, Liberty Alliance Project (12 November 2003). <http://www.projectliberty.org/specs>
- 1186 [LibertyProtSchema1.1] Beatty, John, Kemp, John, eds. "Liberty Protocols and Schema Specification," Version 1.1,
1187 Liberty Alliance Project (January 2003). <http://www.projectliberty/specs>
- 1188 [LibertyBindProf] Cantor, Scott, Kemp, John , eds. "Liberty ID-FF Bindings and Profiles Specification," Version 1.2,
1189 Liberty Alliance Project (12 November 2003). <http://www.projectliberty.org/specs>
- 1190 [RFC1034] Mockapetris, P., eds. (November 1987). "Domain names - concepts and facilities," RFC 1034 / STD 0013,
1191 Internet Engineering Task Force <http://www.rfc-editor.org/rfc/rfc1034.txt>
- 1192 [RFC2119] Bradner, S., eds. "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
1193 Engineering Task Force (March 1997). <ftp://ftp.rfc-editor.org/in-notes/rfc2119.txt>
- 1194 [RFC2246] Dierks, T., Allen, C., , , , eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet
1195 Engineering Task Force <http://www.rfc-editor.org/rfc/rfc2246.txt> [20 December 2002].
- 1196 [RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI):
1197 Generic Syntax," RFC 2396, The Internet Engineering Task Force <http://www.rfc-editor.org/rfc/rfc2396.txt>
1198 [18 December 2002].
- 1199 [RFC2535] Eastlake, D., eds. "Domain Name System Security Extensions," RFC 2535, The Internet Engineering Task
1200 Force (March 1999). <ftp://ftp.rfc-editor.org/in-notes/rfc2535.txt>
- 1201 [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999).
1202 "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force [http://www.rfc-
editor.org/rfc/rfc2616.txt](http://www.rfc-
1203 editor.org/rfc/rfc2616.txt) [18 December 2002].
- 1204 [RFC2732] Hinden, R., Carpenter, B., Masinter, L., eds. (December 1999). "Format for Literal IPv6 Addresses in
1205 URL's ," RFC 2732, The Internet Engineering Task Force <http://www.rfc-editor.org/rfc/rfc2732.txt>
- 1206 [RFC2915] Mealling, M., Daniel, R., eds. (September 2000). "The Naming Authority Pointer (NAPTR) DNS
1207 Resource Record," RFC 2915, Internet Engineering Task Force <http://www.rfc-editor.org/rfc/rfc2915.txt>
1208 [October 2003].
- 1209 [RFC3401] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) Part
1210 One: The Comprehensive DDDS," RFC 3401, The Internet Engineering Task Force [http://www.rfc-
editor.org/rfc/rfc3401.txt](http://www.rfc-
1211 editor.org/rfc/rfc3401.txt)
- 1212 [RFC3403] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) Part Three: The
1213 Domain Name System (DNS) Database ," RFC 3403, The Internet Engineering Task Force [http://www.rfc-
editor.org/rfc/rfc3403.txt](http://www.rfc-
1214 editor.org/rfc/rfc3403.txt)
- 1215 [RFC3404] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) Part Four: The
1216 Uniform Resource Identifiers (URI) ," RFC 3404, The Internet Engineering Task Force [http://www.rfc-
editor.org/rfc/rfc3404.txt](http://www.rfc-
1217 editor.org/rfc/rfc3404.txt)
- 1218 [RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T., eds. (June 2003). "Transport
1219 Layer Security (TLS) Extensions ," RFC 3546, The Internet Engineering Task Force [http://www.rfc-
editor.org/rfc/rfc3546.txt](http://www.rfc-
1220 editor.org/rfc/rfc3546.txt)

- 1221 [SAMLCore11] Maler, E., Mishra, P., Philpott, R., eds. (27 May 2003). "Assertions and Protocol for the
1222 OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, ver-
1223 sion 1.1, Organization for the Advancement of Structured Information Standards [http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security](http://www.oasis-
1224 open.org/committees/documents.php?wg_abbrev=security)
- 1225 [Schema1] Thompson, H.S., Beech, D., Maloney, M., Mendleson, N., eds. (May 2002). "XML Schema Part 1:
1226 Structures," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlschema-1/>
- 1227 [Schema2] Biron, P.V., Malhotra, A., eds. (May 2002). "XML Schema Part 2: Datatypes," Recommendation, World
1228 Wide Web Consortium <http://www.w3.org/TR/xmlschema-2/>
- 1229 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith,
1230 Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001).
1231 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315> [<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>]
- 1232 [XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, Eve, eds. (Oct 2000). "Extensible
1233 Markup Language (XML) 1.0 (Second Edition)," Recommendation, World Wide Web Consortium
1234 <http://www.w3.org/TR/2000/REC-xml-20001006>
- 1235 [XMLDsig] Eastlake, D., Reagle, J., Solo, D., eds. (12 Feb 20002). "XML-Signature Syntax and Processing,"
1236 Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 1237 [xmlesc-core] Eastlake, Donald, Reagle, Joseph, eds. (December 2002). "XML Encryption Syntax and Processing,"
1238 W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlesc-core/>
- 1239 **Informative**
- 1240 [LibertyIDFFOverview] Wason, Thomas, eds. "Liberty ID-FF Architecture Overview," Version 1.2, Liberty Alliance
1241 Project (12 November 2003). <http://www.projectliberty.org/specs>