



Liberty ID-WSF Discovery Service Specification

Version:

2.0-errata-v1.0

Editors:

Conor

Cahill

, Intel Corporation

Jeff

Hodges

, NeuStar, Inc.

Contributors:

Robert

Aarts

, NokiaSun Microsystems, Corporation

John

Beatty

, Sun Microsystems, Inc.

Carolina Canales-Valenzuela, Ericsson

Darryl Champagne, IEEE-ISTO

Gary Ellison, Sun Microsystems, Inc.

Jukka Kainulainen, Nokia Corporation

John Kemp, Nokia Corporation

Rob

Lockhart

, IEEE-ISTO

Paul Madsen, NTT, formerly Entrust, Inc.

Jonathan

Sergent

, Sun Microsystems, Inc.

Greg Whitehead, Hewlett-Packard

Emily Xu, Sun Microsystems, Inc.

Abstract:

This specification defines mechanisms for describing and discovering identity web services.

Filename: liberty-idwsf-disco-svc-2.0-diff-v1.0.pdf

1 **Notice**

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

6 Implementation of certain elements of this document may require licenses under third party intellectual property rights,
7 including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not and
8 shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual
9 property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any
10 warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringe-
11 ment of third party intellectual property rights, and fitness for a particular purpose.** Implementers of this
12 Specification are advised to review the Liberty Alliance Project's website (<http://www.projectliberty.org/>) for infor-
13 mation concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance
14 Management Board.

15 Copyright © 2007 2FA Technology; Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.; Amer-
16 ican Express Company; Amsoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Axalto; Bank of
17 America Corporation; Beta Systems Software AG; BIPAC; British Telecommunications plc; Computer Associates
18 International, Inc.; Credentica; DataPower Technology, Inc.; Deutsche Telekom AG, T-Com; Diamelle Technologies,
19 Inc.; Diversinet Corp.; Drummond Group Inc.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Falkin Systems
20 LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le développement
21 de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens Ltd.; GSA Office of Governmentwide Policy;
22 Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke & Devrient GmbH; Hewlett-Packard GSA Office
23 Company; Hochhauser & Co., Policy; Hewlett-Packard LLC; IBM Corporation; Intel Corporation; Intuit Inc.; Kant-
24 ega; Kayak Interactive; Livo Technologies; Luminance Consulting Services; MasterCard International; MedCommons
25 Inc.; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; NTT DoCoMo, Inc.; Netegrity, Inc.; Neustar, Inc.;
26 New Zealand Government State Services Commission; Nippon Telegraph and Telephone Corporation; Nokia Corpo-
27 ration; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation; RSA Security
28 Inc.; Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
29 Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
30 Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security; Trusted Network Technologies;
31 UNINETT AS; UTI; VeriSign, Inc.; Vodafone Group Plc.; Wave Systems Corp. All rights reserved.

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

38 Contents

39	1. Introduction.....	8
40	1.1. Conceptual Model and Terminology	8
41	1.2. Scope.....	9
42	1.3. Notation and Conventions.....	9
43	1.3.1. XML Namespaces	9
44	2. Discovery Service Information Model	11
45	2.1. Overview of Discovery Service Information Model	11
46	2.2. Versioning in ID-WSF	12
47	2.3. ID-WSF Endpoint References (EPRs)	13
48	2.3.1. EPR Profiling Attributes	14
49	2.3.2. EPR Profiling Elements	15
50	2.3.3. ID-WSF Web Services Addressing EPR Profile	19
51	2.3.4. Effective Web Services Addressing EPR	23
52	2.3.5. Example Liberty ID-WSF EPRs	24
53	2.4. Service Metadata	25
54	2.4.1. Service Metadata element.....	25
55	2.4.2. Minting ID-WSF EPRs based upon Service Metadata.....	28
56	2.4.3. Service Metadata Example	28
57	3. Discovery Service.....	32
58	3.1. Service URIs	33
59	3.2. Status Codes	33
60	3.3. Operation: <i>DiscoveryQuery</i>	34
61	3.3.1. wsa: Action values for <i>DiscoveryQuery</i> Messages	34
62	3.3.2. <Query> Message	34
63	3.3.3. QueryResponse	38
64	3.3.4. DiscoveryQuery Processing Rules	40
65	3.4. Operation: <i>MDAssociationAdd</i>	40
66	3.4.1. wsa: Action values for <i>MDAssociationAdd</i> Messages	41
67	3.4.2. SvcMDAssociationAdd Message	41
68	3.4.3. SvcMDAssociationAddResponse Message	41
69	3.4.4. MDAssociation Add Processing Rules	42
70	3.5. Operation: <i>MDAssociationQuery</i>	43
71	3.5.1. wsa: Action values for <i>MDAssociationQuery</i> Messages	43
72	3.5.2. SvcMDAssociationQuery Message	43
73	3.5.3. SvcMDAssociationQueryResponse Message	43
74	3.5.4. MDAssociation Query Processing Rules	44
75	3.6. Operation: <i>MDAssociationDelete</i>	44
76	3.6.1. wsa: Action values for <i>MDAssociationDelete</i> Messages	44
77	3.6.2. SvcMDAssociationDelete Message	44
78	3.6.3. SvcMDAssociationDeleteResponse Message	45
79	3.6.4. MDAssociation Delete Processing Rules	45
80	3.7. Operation: <i>MetadataRegister</i>	46
81	3.7.1. wsa: Action values for <i>MetadataRegister</i> Messages	46
82	3.7.2. SvcMDRegister Message	46
83	3.7.3. SvcMDRegisterResponse Message	47
84	3.7.4. Metadata Register Processing Rules	48
85	3.8. Operation: <i>MetadataQuery</i>	49
86	3.8.1. wsa: Action values for <i>MetadataQuery</i> Messages	49
87	3.8.2. SvcMDQuery Message	49
88	3.8.3. SvcMDQueryResponse Message	49
89	3.8.4. Metadata Query Processing Rules	50

90	3.9. Operation: <i>MetadataReplace</i>	51
91	3.9.1. wsa:Action values for MetadataReplace Messages	51
92	3.9.2. SvcMDReplace Message	51
93	3.9.3. SvcMDReplaceResponse Message	51
94	3.9.4. Metadata Replace Processing Rules	52
95	3.10. Operation: <i>MetadataDelete</i>	53
96	3.10.1. wsa:Action values for MetadataDelete Messages	53
97	3.10.2. SvcMDDelete Message	53
98	3.10.3. SvcMDDeleteResponse Message	53
99	3.10.4. Metadata Delete Processing Rules	54
100	3.11. Option Value for Response Authentication	55
101	3.12. Including Keys in the <i>SvcMDRegisterResponse</i> Message	55
102	3.13. Discovery Service Example Messages (NON-Normative)	56
103	3.13.1. Query People Service	56
104	3.13.2. Query provider	58
105	3.13.3. Query (empty)	59
106	3.13.4. Query People Service and Provider	61
107	3.13.5. SvcMDQuery (empty)	61
108	3.13.6. SvcMDQuery w/Bad SvcMDID	62
109	3.13.7. SvcMDRegister w/single SvcMD	63
110	3.13.8. SvcMDQuery w/Good SvcMDID	63
111	3.13.9. SvcMDDelete w/Good SvcMDID	64
112	3.13.10. SvcMDDelete w/Already Deleted SvcMDID	64
113	3.13.11. SvcMDRegister w/Complex SvcMD	65
114	3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD	66
115	3.13.13. SvcMDReplace of existing SvcMD	67
116	3.13.14. SvcMDQuery of Replaced SvcMDID	67
117	3.13.15. SvcMDDelete of replaced SvcMD	68
118	3.13.16. SvcMDRegister w/multiple SvcMDs	68
119	3.13.17. Query Calendar Service	70
120	3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD	71
121	3.13.19. SvcMDAssociationQuery w/SvcMDID	71
122	3.13.20. SvcMDAssociationQuery w/o SvcMDID	71
123	3.13.21. Query Calendar Service (again)	72
124	3.13.22. Query Calendar Service w/Action	73
125	3.13.23. Query Calendar Service w/resultsType=all	74
126	3.13.24. Query for all Calendar Service EPRs	75
127	3.13.25. Query for one Calendar Service EPRs	77
128	3.13.26. Query specific version of Calendar Service	78
129	3.13.27. SvcMDReplace Calendar Service	79
130	3.13.28. Query old Calendar Service	80
131	3.13.29. Query for all Calendar Service EPRs	80
132	3.13.30. SvcMDAssociationAdd the ATM Service SvcMD	81
133	3.13.31. Query ATM Service w/resultsType=best	81
134	3.13.32. Query ATM Service w/Withdraw Action	83
135	3.13.33. Query ATM Service w/Option	84
136	3.13.34. Query ATM Service w/unknown Option	85
137	3.13.35. Query ATM Service w/good and bad Option	86
138	4. Discovery Service ID-WSF EPR conveyed via a Security Token	87
139	4.1. EPR Generation Rules	87
140	4.2. SAML 2.0 Security Tokens	88
141	4.3. SAML 1.x (Liberty ID-FF) Security Tokens	88
142	5. ID-WSF 1.x Resource Offering conveyed in an EPR	90

143 6. Acknowledgments 92
144 References..... 93
145 A. Discovery Service Version 2.0 XSD 95
146 B. Discovery Service WSDL 102

147 1. Introduction

148 This specification defines a framework for describing and discovering web services in general and *identity web serv-*
149 *ices* in particular. The conceptual model and terminology is first provided to set the context for the rest of the
150 specification. Next, the data types for the information maintained by a Discovery Service are specified. Then the
151 Discovery Service itself is specified.

152 1.1. Conceptual Model and Terminology

153 An *identity web service* is defined as a type of web service whose operations are indexed by *identity*. Such services
154 maintain information about, or on behalf of, *Principals* — as represented by their *identities* — and/or perform actions
155 on behalf of Principals. They are also sometimes referred to as simply *identity services*.

156 There are various types of services, each of which is assigned a unique *service type* identifier, encoded as a *URI* (Uniform
157 Resource Identifier). This *service type URI* maps to exactly one version of an *abstract WSDL* definition of a service,
158 which contains the `<wsdl:types>`, `<wsdl:message>`, and `<wsdl:portType>` elements of a WSDL 1.1 description
159 [WSDLv1.1].

160 An example of a type of identity web service is a Principal's "calendar service," which could be identified by a URI
161 such as `urn:example:services:calendar:2006-12`. Note the use of the year/month in the service type to identify the
162 version of the abstract WSDL.

163 A *service instance* is a deployed physical instantiation of a particular type of service. A *service provider* may deploy
164 one or more concrete service instances in the act of deploying a service.

165 A service instance may be described by a *concrete WSDL* document (including at least the `<wsdl:binding>`,
166 `<wsdl:service>`, and `<wsdl:port>` elements) which contains the *protocol endpoint* and additional information
167 necessary for a client to communicate with a particular service instance. An example of such "additional information"
168 is communication security policy information.

169 A service instance is hosted by some *provider*, identified by a URI. An example of a service instance is a SOAP-over-
170 HTTP endpoint offering a calendar service, being hosted by some provider.

171 Thus, a service instance exposes a protocol interface to a set of logical resources, nominally indexed by Principal. A
172 *resource* in this specification is either data related to some *Principal's identity* or a service acting on behalf of some
173 Principal. An example of a resource is a calendar containing appointments for a particular Principal. When a client
174 sends a request message to a service instance, information in the message serves to implicitly identify the resource
175 being acted upon. This is accomplished in one of the following fashions:

- 176 • Implicitly (e.g., PAOS exchange [LibertyPAOS]).
- 177 • Via a `<TargetIdentity>` header block [LibertySOAPBinding].
- 178 • Via supplied security token: it is presumed that a resource of the security token subject, i.e., the Principal itself, is
179 to be accessed.
- 180 • Via the endpoint. A service may choose to offer different endpoints for every resource. The simplest case of this
181 is to represent the resource as a part of the query string.

182 Caution should be exercised when using this unique endpoint solution as the use of unique endpoints for every
183 resource can release enough information to allow collusion across providers as to the identity of a principal (if
184 multiple providers get the same unique endpoint reference for their local principal, they can figure out that the local
185 principal on their respective environment is the same principal).

186 A resource commonly has access control policies associated with it. These access control policies are typically under
187 the purview of the entity or entities associated with the resource (in common language, the entity or entities could be

188 said to "own", or "manage", the resource). The access control policies associated with a resource must be enforced by
189 the service instance.

190 The Discovery Service defined here is not intended to be exclusive. Some identity services meeting the conceptual
191 model may be exposed via other discovery mechanisms. For example, [LibertyPAOS] defines an equivalent discovery
192 mechanism.

193 1.2. Scope

194 This specification:

- 195 • Specifies service instance endpoint description and enumeration via a profile of W3C Web Services Addressing
196 [WSAv1.0].
- 197 • Specifies a Discovery Service facilitating discovery and invocation of service instances.
- 198 • A SAML (see [SAMLCore2]) <Attribute> element defined such that an Endpoint Reference (EPR) for the
199 Discovery Service itself can be conveyed via SAML assertions. This is known as a *Discovery EPR* or *DS EPR* and
200 also colloquially as the *discovery bootstrap*.

201 1.3. Notation and Conventions

202 This specification uses schema documents conforming to W3C XML Schema (see [Schema1-2]) and normative text
203 to describe the syntax and semantics of XML-encoded messages.

204 The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT,"
205 "RECOMMENDED," "MAY," and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

209 1.3.1. XML Namespaces

210 The following XML namespaces are referred to in this document:

- 211 • The prefix *ds*: represents the Discovery Service namespace. This namespace is the default for instance fragments,
212 type names, and element names in this document. In schema listings, and in examples of Discovery Service mes-
213 sages and fragments thereof, this is the default namespace *when* no prefix is shown:

214 *urn:liberty:disco:2006-08*

- 215 • The prefix *saml2*: stands for the SAMLv2 assertion namespace [SAMLCore2]:

216 *urn:oasis:names:tc:SAML:2.0:assertion*

- 217 • The prefix *samlp2*: stands for the SAMLv2 protocol namespace [SAMLCore2]:

218 *urn:oasis:names:tc:SAML:2.0:protocol*

- 219 • The prefix *sb*: stands for the Liberty Soap Bindings namespace [LibertySOAPBinding]:

220 *urn:liberty:sb:2006-08*

- 221 • The prefix *sbf*: stands for the Liberty Soap Bindings Framework namespace [LibertySOAPBinding]:

222 *urn:liberty:sb*

- 223 • The prefix *sec:* stands for the Liberty Security Mechanisms namespace [[LibertySecMech](#)]:

224 *urn:liberty:security:2006-08*

- 225 • The prefix *wsa:* stands for the W3C Web Services Addressing (WSA) namespace [[WSAv1.0](#)]:

226 *http://www.w3.org/@@@/@/@/addressing*

- 227 • The prefix *wSDL:* stands for the primary WSDL v1.1 namespace [[WSDLv1.1](#)]:

228 *http://schemas.xmlsoap.org/wSDL/*

- 229 • The prefix *wSDLsoap:* stands for the namespace of the WSDL-SOAP binding [[WSDLv1.1](#)]:

230 *http://schemas.xmlsoap.org/wSDL/soap/*

- 231 • The prefix *xs:* stands for the W3C XML schema namespace [[Schema1-2](#)]:

232 *http://www.w3.org/2001/XMLSchema*

- 233 • The prefix *xsi:* stands for the W3C XML schema instance namespace:

234 *http://www.w3.org/2001/XMLSchema-instance*

235 2. Discovery Service Information Model

236 This section describes the Discovery Service information model. This model comprises the various data types, and
237 thus information, that are maintained and managed by the Discovery Service, as well as the manner and format in which
238 this information is exchanged between the Discovery Service and its clients.

239 First, there is a brief non-normative overview describing how *service instances* are referenced, as well as the interactions
240 between the Discovery Service and the various other roles donned by system entities in the ID-WSF framework. Next
241 are the normative definitions of the various elements defined in this specification and used in referencing service
242 instances. Lastly is the Discovery Service WSA Profile, which normatively defines WSA EPRs profiled for use in
243 referencing ID-WSF service instances.

244 2.1. Overview of Discovery Service Information Model

245 A *service instance* is a web service at a distinct protocol endpoint. Information about service instances needs to be
246 communicated in various contexts. This specification defines a profile of WSA Endpoint References (EPRs)
247 [WSAv1.0][WSAv1.0-SOAP] such that they can be used to convey service instance information needed by entities
248 wishing to communicate with said service instances. Such "profiled EPRs" are termed "ID-WSF EPRs" in the remainder
249 of this specification.

250 The general model for ID-WSF system entity interactions from a Principal's perspective is as follows:

- 251 • A Principal wielding some *user agent* interacts with some *service provider* and is authenticated in some Liberty-
252 compliant fashion, such that the service provider obtains possession of a *discovery bootstrap* assertion for the
253 Principal. This assertion contains a pointer to the Principal's Discovery Service instance in the form of an ID-WSF
254 EPR.
- 255 • Now, the service provider, acting as a *web service consumer* (WSC) and using the ID-WSF EPR obtained above,
256 queries the Principal's Discovery Service for a pointer to some other desired service of the **Principal—e.g., the**
257 **Principal's Profile Service or Calendar Service.**
- 258 • The Discovery Service returns one or more ID-WSF EPRs to the querying WSC, pointing to the Principal's service
259 instance(s), of the requested type, if any.
- 260 • The WSC now employs the returned ID-WSF EPR(s) to interact with the identified service instance(s), which
261 themselves will be acting in the role of a *web service provider*. The WSC returns results as appropriate to the
262 Principal's user agent.

263 There are various permutations of this general interaction model. For example, the Principal's user agent may itself
264 act in the role of a WSC. Or, a Principal may not be actively involved in a given interaction—a WSC is simply
265 interacting with a WSP on a Principal's behalf. For example, it may be renewing some contract, such as a magazine
266 subscription.

267 In order to enable the above Principal's-perspective model, there is a parallel model from the *web service provider's*
268 (WSP) perspective, which is as follows:

- 269 • A service instance(s), acting as a WSP, is deployed at some addressable endpoint(s). In this example, the WSP is
270 providing some service(s) on behalf of one or more Principals, **e.g., a profile or calendar service.**
- 271 • The WSP registers itself with the Discovery Service by inserting Service Metadata into the DS using the Service
272 Metadata maintenance operations (defined later in this specification). The Service Metadata describes the WSP's
273 service instance(s) such that the Discovery Service has the necessary information to mint ID-WSF EPRs for a WSC
274 to invoke that WSP.

- 275 • The Service Metadata, using appropriate Discovery Service protocol operations (defined later in this specification),
276 is then "associated" with a principal'.
- 277 • The above Principal's-perspective model is now enabled.
- 278 There are various permutations of this general WSP-perspective service instance registration model. For example,
279 the same administrative entity may be deploying the both the Discovery Service and the other services and so may
280 employ alternative means, e.g., bulk configuration, to effect service metadata registration with their Discovery
281 Service.

282 2.2. Versioning in ID-WSF

283 Versioning applies to both the communications framework and the service itself within Liberty. The Discovery Service
284 is at the center of versioning in Liberty because it is the entity that matches the version capabilities of the WSC to that
285 of the WSP.

286 The specific areas of versioning include:

- 287 • **Service Versioning** — the version of the Service APIs that are available from a service instance.

288 The service version is implicitly part of the `<ServiceType>` URI which identifies a specific version of a specific
289 logical service (e.g., a Profile service or a Discovery Service). For example, the service type URI: `urn:
290 liberty:disco:2006-08` represents a specific version of the logical service "Discovery Service" (in this case,
291 a version identified by the date 2006-08).

292 There are times when the Discovery Service may need to know the logical type of service. In the case of Liberty
293 defined services, we have adopted the convention of using a colon separated URN in the "liberty" namespace where
294 the last element is the version identifier. In this case, the Discovery service could extract the logical type of service
295 from the service type. However, that is just the Liberty convention and does not necessarily apply to service type
296 definitions outside of the Liberty URN namespace.

297 In non-Liberty-defined `<ServiceType>` URIs, the Discovery Service may understand the convention used in that
298 particular namespace, or it may have some configuration defined knowledge of which URIs match to which logical
299 type of service (perhaps via a user configurable parameter). If the DS cannot separate the logical type of service
300 from the service version, the DS SHOULD treat the entire `<ServiceType>` URI as the logical type of service.

301 The `<ServiceType>` URI is also typically used as the Namespace identifier for the XML schema for the service,
302 so the version identifier typically shows up there as well -- although this is NOT a normative requirement.

- 303 • **Framework Versioning** — the version of the communications framework used for ID-WSF messaging. Each ID-
304 WSF message has a potential collection of SOAP headers defined by the various ID-WSF specifications which are
305 tied together by the [LibertyIDWSF20SCR]. The [LibertySOAPBinding] specification defines the
306 `<Framework>` element which carries a description of the framework. As of this release that consists primarily of
307 a `version` attribute. [LibertyIDWSF20SCR] defines a particular version string to represent each concrete version
308 of the specifications.

309 The `Framework description` is included in ID-WSF messages, ID-WSF minted EPRs and in Discovery Service
310 `<Query>` operations (in other words, the framework description is actively specified at each stage of the ID-WSF
311 interaction model).

312 To ensure that the WSC communicates appropriately (from a versioning point of view) with the WSP, the WSC specifies
313 both the service and framework versions that it supports during discovery and the Discovery Service matches the WSC
314 capabilities with the appropriate registered service instances in order to return an EPR that the WSC can use.

315 2.3. ID-WSF Endpoint References (EPRs)

316 The general form of an EPR is illustrated in [Example 1](#).

```
317 <wsa:EndpointReference ...>
318   <wsa:Address>...some URI here...</wsa:Address>
319   <wsa:ReferenceParameters>.....</wsa:ReferenceParameters>
320   <wsa:Metadata>...some metadata here... </wsa:Metadata>
321 </wsa:EndpointReference>
```

326 **Example 1. General Form of an EPR**

327 The EPRs are profiled, as specified below in [Section 2.3.3](#), by placing Liberty-specific attributes and elements into the
328 EPR. Specifically, a few attributes on the EPR itself and some sub-elements within `<wsa:Metadata>` element of the
329 EPR. These Liberty-specific components are defined below in [Section 2.3.1: EPR Profiling Attributes](#) and [Section 2.3.2: EPR Profiling Elements](#). These profiled EPRs are referred to as "ID-WSF EPRs", [Example 2](#) illustrates an
330 ID-WSF EPR.
331 ID-WSF EPR.

332 **Note**

333 The use of these profiled EPRs does not necessarily replace WSDL; rather, they may be used either in con-
334 junction with WSDL or without. This is also described in [Section 2.3.3](#).

```

335
336 <wsa:EndpointReference
337     notOnOrAfter="2005-08-15T23:18:56Z"
338     ...>
339   <wsa:Address>
340     https://profile-provider.com/profiles/someFoobarProfile
341   </wsa:Address>
342
343   <wsa:Metadata>
344     <ds:Abstract>
345       This is a personal profile containing common name information.
346     </ds:Abstract>
347
348     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
349
350     <ds:ServiceType>&PS1Namespace;</ds:ServiceType>
351
352     <ds:Framework version="2.0" />
353
354     <ds:SecurityContext>
355       <ds:SecurityMechID>
356         urn:liberty:security:2005-02:TLS:SAML
357       </ds:SecurityMechID>
358
359       <sec:Token>
360         <!-- some security token goes here -->
361       </sec:Token>
362     </ds:SecurityContext>
363
364     <ds:Options>
365       <ds:Option>urn:liberty:id-sis-pp</ds:Option>
366       <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
367       <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
368       <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
369     </ds:Options>
370   </wsa:Metadata>
371 </wsa:EndpointReference>
372

```

Example 2. An Instantiated ID-WSF EPR

2.3.1. EPR Profiling Attributes

This section defines the attributes that are used to profile EPRs as defined below in [Section 2.3.3: ID-WSF Web Services Addressing EPR Profile](#). The full Discovery Service schema is given in [Appendix A: Discovery Service Version 2.0 XSD](#).

2.3.1.1. wsu:Id — unique identifier for xml references to an EPR.

The `wsu:Id` attribute ([Figure 1](#)) is used when other elements in the XML document (e.g., message) need to refer to this EPR (for example, when this element is referenced in an XML signature).

```

381
382
383   <!-- wsu:Id attribute for EPR - for xml document references to EPR -->
384
385   <xs:attribute ref="wsu:Id" use="optional" />

```

Figure 1. wsu:Id — Schema Fragment

387 2.3.1.2. reqRef — request reference

388 The reqRef attribute (Figure 2) identifies which <RequestedServiceType> element in the Discovery Service
389 <Query> request that this EPR was minted in response to. In other words this is used to associate the EPR in the
390 <QueryResponse> with the <RequestedServiceType> in the <Query> request.

```
391  
392     <!-- Request Reference - to id which Request generated EPR -->  
393  
394     <xs:attribute name="reqRef" type="xs:string" use="optional"/>
```

395 **Figure 2. reqRef — Schema Fragment**

396 2.3.1.3. notOnOrAfter

397 The notOnOrAfter attribute states the expiration timestamp for the EPR with which it is associated (Figure 3). See
398 Example 2, above, for an instantiated EPR example.

399 Values of the notOnOrAfter attribute MUST be expressed in accordance with Liberty ID-WSF time value restrictions.

400 Liberty system entities SHOULD NOT rely on time resolution finer than milliseconds. Implementations MUST NOT
401 generate time instants that specify leap seconds.

```
402  
403     <!-- EPR Expiration Timestamp -->  
404  
405     <xs:attribute name="notOnOrAfter" type="xs:dateTime"/>
```

408 **Figure 3. notOnOrAfter — Schema Fragment**

409 2.3.2. EPR Profiling Elements

410 This section defines the elements that are used to profile EPRs as defined below in Section 2.3.3: ID-WSF Web
411 Services Addressing EPR Profile . The full Discovery Service schema is given in Appendix A: Discovery Service
412 Version 2.0 XSD .

413 2.3.2.1. Abstract

414 The <Abstract> element (Figure 4) is used for conveying a textual, natural language description of the service
415 instance.

```
416  
417     <!-- Abstract: natural-language description of service -->  
418  
419     <xs:element name="Abstract" type="xs:string"/>
```

421 **Figure 4. <Abstract> — Schema Fragment**

422 2.3.2.2. Provider ID

423 The <ProviderID> element (Figure 5) contains the URI of the provider of this service instance.

```
424
425 <!-- Provider ID -->
426
427 <xs:element name="ProviderID" type="xs:anyURI"/>
428
```

Figure 5. <ProviderID> — Schema Fragment

2.3.2.3. Service Type

The <ServiceType> element (Figure 6) is used to identify a service type and version. This URI needs be constant across all implementations of a service to enable interoperability. Therefore, it is RECOMMENDED that this URI be the same as the targetNamespace URI of the abstract WSDL description for the service.

```
434
435 <!-- Service Type -->
436
437 <xs:element name="ServiceType" type="xs:anyURI"/>
438
439
```

Figure 6. <ServiceType> — Schema Fragment

Some examples of possible ServiceType URIs:

```
442 urn:liberty:disco:2006-08
443 urn:liberty:id-sis-pp:2003-08
444 http://myservices.com/gaming/1.0
```

2.3.2.4. securityContext

The <SecurityContext> element (Figure 7) is a container in which <SecurityMechID> elements and <sec:Token> elements are placed and thus associated with an ID-WSF EPR. The <sec:Token> element is used to either directly contain, or reference, security tokens and/or identity tokens.

Therefore, the <SecurityContext> element serves to denote the invocation context necessary for interacting with the service instance represented by the containing ID-WSF EPR.

NOTE: in some cases the DS will **not** be able to generate the necessary tokens to complete the security context. This will usually happen when a context needs a security token from a provider other than the DS (such as a non-related IdP). In such cases, the DS will include an empty token element with the *ref* attribute set to the following URI:

- urn:liberty:disco:tokenref:ObtainFromIDP

In such cases, the WSC receiving the EPR MUST communicate with the invoking principal's IdP's SSO Service (see [LibertyAuthn]) in order to obtain the necessary security token.

The value of the security mechanism in the security context will identify the type of security token that the WSC should request from the IdP. For example, if the security mechanism was "urn:liberty:....SAMLV2," the WSC would know they needed a SAML 2.0 token with a subject confirmation of "....holder-of-key" and would indicate so on the SSO Service request.

An ID-WSF EPR MAY contain more than one <SecurityContext> element. This serves to denote mutually-exclusive groupings of <SecurityMechID>s and <sec:Token>s, and thus different security contexts.

See Section 2.3.3: ID-WSF Web Services Addressing EPR Profile, below, for the precise specification of the mapping of <SecurityContext>, and its contents, to ID-WSF EPRs.

```

465
466 <!-- Security Context Container -->
467
468 <xs:element name="SecurityContext">
469   <xs:complexType>
470     <xs:sequence>
471       <xs:element ref="SecurityMechID"
472         minOccurs="1"
473         maxOccurs="unbounded"/>
474
475       <xs:element ref="sec:Token"
476         minOccurs="0"
477         maxOccurs="unbounded"/>
478     </xs:sequence>
479   </xs:complexType>
480 </xs:element>
481
482

```

483 **Figure 7. <SecurityContext> — Schema Fragment**

484 See [Example 2](#), above, for an instantiated ID-WSF EPR example, containing a <SecurityContext> element, itself
 485 containing <SecurityMechID> and <sec:Token> elements.

486 **2.3.2.5. SecurityMechID**

487 The <SecurityMechID> element ([Figure 8](#)) specifies the security mechanism(s) supported by the service instance
 488 represented by the ID-WSF EPR. These security mechanisms are represented as URIs, and are defined in [[Liberty-](#)
 489 [SecMech](#)].

490 The <SecurityMechID> element is used within the <SecurityContext> element described above. This is detailed
 491 in the [Section 2.3.3: ID-WSF Web Services Addressing EPR Profile](#) section below.

```

492
493 <!-- Security Mechanism ID -->
494
495 <xs:element name="SecurityMechID" type="xs:anyURI"/>
496
497

```

498 **Figure 8. <SecurityMechID> — Schema Fragment**

499 Some examples of possible SecurityMechID URI values (from [[LibertySecMech](#)]):

```

500   urn:liberty:security:2006-08:ClientTLS:SAMLV2
501   urn:liberty:security:2003-08:ClientTLS:SAML
502   urn:liberty:security:2006-08:TLS:SAMLV2

```

503 See [Example 2](#), above, for an instantiated ID-WSF EPR example, containing a <SecurityContext> element con-
 504 taining <SecurityMechID> and <sec:Token> elements.

505 **2.3.2.6. Framework**

506 The <Framework> element ([Figure 9](#)) identifies the Liberty ID-WSF framework supported by the service instance at
 507 this endpoint. There MUST be at least one <Framework> element within an EPR.

508 Multiple <Framework> elements indicate that the service instance supports any of the specified ID-WSF versions at
 509 this same endpoint.

510 The structure and content of this element is defined in [[LibertySOAPBinding](#)].

```

511
512 <!-- Framework Description -->
513
514 <xs:element ref="sb:Framework" maxOccurs="unbounded" />
515
516

```

517 **Figure 9. <Framework> — Schema Fragment**

518 2.3.2.7. Action

519 The optional multi-occurrence <Action> element (Figure 10) is used to identify the set of interfaces exposed by the
520 provider at this endpoint.

521 Each <Action> element contains a URI that MUST match one of the <wsa:Action> URIs defined for the service.

522 When there are no <Action> elements in an EPR, the EPR can be used to invoke **all** of the interfaces for the defined
523 service type.

524 This element is typically only included when the service instance specified in the EPR can only address a sub-set of
525 the service's interfaces. A service instance may do this to scale their resources across different interfaces. For example,
526 a service instance of the personal profile service may support the Query interface on a large cluster of systems, but
527 require that the less frequently called, modify operations take place on some dedicated hardware.

```

528
529 <!-- Action(s) - the interfaces available at this service -->
530
531 <xs:element name="Action" type="xs:anyURI" />
532

```

533 **Figure 10. <Action> — Schema Fragment**

534 2.3.2.8. Options

535 The <Options> element (Figure 11) expresses the "options" supported by a service instance. Thus they provide hints
536 to a potential requester whether certain data or operations may be available with a particular service instance.

537 For example, an option may be provided stating that home contact information is available. If no Options element is
538 present, it means only that the service instance does not advertise whether any options are available. Options may, in
539 fact, be employed by the service instance. For example, it may be a simple service that is not capable of updating its
540 entry in the Discovery Service when the available options change, so it avoids listing them at all. If the Options element
541 is present, but is empty, it means that the service instance explicitly advertises that no options are available.

```

542
543 <!-- Options -->
544
545 <xs:element name="Options" type="OptionsType"/>
546
547 <xs:element name="Option" type="xs:anyURI" />
548
549 <xs:complexType name="OptionsType">
550   <xs:sequence>
551     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded" />
552   </xs:sequence>
553 </xs:complexType>
554
555

```

556 **Figure 11. <Options> — Schema Fragment**

557 The <Options> element contains zero or more <Option> elements, each of which contains a URI identifying a
558 particular option. The set of possible URIs for an <Option> element should be defined by the service type. For example,

559 a person profile service specification would specify a set of options particular to its own domain. However, one common
 560 <Option> flag related to security, and thus common to ID-WSF services, is defined in Section 3.11: [Option Value](#)
 561 [for Response Authentication](#) .

562 2.3.3. ID-WSF Web Services Addressing EPR Profile

563 This section specifies the profile of WSA Endpoint References (EPRs). Profiling an EPR, yielding an ID-WSF EPR,
 564 is accomplished by placing various of the elements defined in Section 2.3.2: [EPR Profiling Elements](#) , above, into the
 565 EPR's <wsa:Metadata> element according to the rules defined below. All ID-WSF EPRs must adhere to the per-
 566 element rules in Section 2.3.2, and thereupon adhere to the rules defined in the following sections, depending upon the
 567 intended usage scenario for the ID-WSF EPR being minted.

568 For reference, the general form of an instantiated EPR is illustrated above in [Example 1](#), and the <wsa:
 569 EndpointReference> schema fragment [WSAv1.0-SOAP] is illustrated below in [Figure 12](#).

570 An ID-WSF EPR is normatively defined as a <wsa:EndpointReference> profiled as per this section.

571 Note

572 Except for the <wsa:Address> and <wsa:ReferenceParameters> elements, all elements discussed in
 573 the below sections are denoted as either being "absent" or "present" as content of the <wsa:Metadata>
 574 element of the ID-WSF EPR being minted.

```

575
576 <xs:element name="EndpointReference" type="tns:EndpointReferenceType"/>
577
578 <xs:complexType name="EndpointReferenceType">
579   <xs:sequence>
580     <xs:element name="Address"
581               type="tns:AttributedURIType"/>
582
583     <xs:element name="ReferenceParameters"
584               type="tns:ReferenceParametersType"
585               minOccurs="0"/>
586
587     <xs:element ref="tns:Metadata"
588               minOccurs="0"/>
589
590     <xs:any namespace="##other"
591             processContents="lax"
592             minOccurs="0"
593             maxOccurs="unbounded"/>
594   </xs:sequence>
595
596   <xs:anyAttribute namespace="##other" processContents="lax"/>
597 </xs:complexType>
598
599
600
```

601 **Figure 12.** <wsa:EndpointReference> — Schema Fragment

602 2.3.3.1. ID-WSF EPR Minting Rules

603 ID-WSF EPRs are minted by both the Discovery Service (in response to <Query> requests) and by system entities
 604 acting as in a WSC or WSP role for inclusion in SOAP message header blocks such as the <wsa:ReplyTo> and the
 605 <wsa:FaultTo>, as discussed in [\[LibertySOAPBinding\]](#). This section refers to these different parties collectively as
 606 "issuers."

607 The following rules MUST be observed by issuers when constructing an ID-WSF EPR:

- 608 1. A `notOnOrAfter` attribute MAY be present in each ID-WSF EPR. If absent, or if it has a value of
609 `1970-01-01T00:00:00Z`, it means the issuer is not stipulating an expiration time for this ID-WSF EPR, and that
610 its wielder is obliged to follow its own local policy for refreshing any cached copies. If present, the value should
611 be set by the issuer according to local policy.
- 612 2. The value of the `<wsa:Address>` element MUST contain the endpoint address of the service instance being
613 described by this EPR. This literally-addressed form of ID-WSF EPR is useful in order to ease the burden of
614 WSCs from having to retrieve and parse WSDL in common cases. Additionally, the rules specified in [Section 2.3.3.2: ID-WSF EPR Specifics](#) MUST be adhered to.
615
- 616 3. A `wsu:Id` attribute MAY be present on the EPR root element.
- 617 4. A `reqRef` attribute MAY be present on the EPR root element.
- 618 5. Exactly one `<Abstract>` element MAY be present in the EPR `<Metadata>` element.
- 619 6. Exactly one `<ProviderID>` element MUST be present in the EPR `<Metadata>` element.
- 620 7. One or more `<ServiceType>` elements MUST be present in the EPR `<Metadata>` element.
- 621 8. One or more `<Framework>` elements MUST be present in the EPR `<Metadata>` element.
- 622 9. Optionally, one or more `<Options>` element(s). These are discussed in detail above, in [Section 2.3.2.8](#).
- 623 10. Optionally, one or more `<Action>` element(s). These are discussed in detail above, in [Section 2.3.2.7](#).
- 624 11. One or more `<SecurityContext>` elements SHOULD be present in each ID-WSF EPR. If so they, and their
625 content, MUST adhere to the rules below, as well as the additional specific rules in [Section 2.3.3.3: Security](#)
626 [Mechanism Specifics](#) :
- 627 a. If no security or identity tokens are to be embedded, then place all the supported security mechanisms, denoted
628 by `<SecurityMechID>` elements, in a single `<SecurityContext>` element.
- 629 b. Else, if security and/or identity tokens are to be embedded or referenced (via `<sec:Token>` elements), then
630 one MUST group corresponding `<SecurityMechID>` and `<sec:Token>` elements into the same
631 `<SecurityContext>` element. In other words, all security and identity tokens within a
632 `<SecurityContext>` element MUST apply to ALL of the security mechanisms in the same context.
- 633 c. A security and/or identity token embedded in a `<sec:Token>` in a given ID-WSF EPR's
634 `<SecurityContext>` element MAY be referenced from other `<SecurityContext>` elements, whether
635 the other `<SecurityContext>` elements are contained within the given ID-WSF EPR or whether they are
636 in another ID-WSF EPR in the list of ID-WSF EPRs being constructed.
- 637 Such referencing is accomplished by using the `ref` attribute of a `<sec:Token>` element. When constructing
638 such a reference, the referencing `<sec:Token>` MUST reference the `<sec:Token>` element containing the
639 target embedded security token, as specified in [[LibertySecMech](#)].
- 640 d. All `<sec:Token>` elements included in the `<SecurityContext>` element MUST have the `usage` attribute
641 set to the appropriate value (as documented in [[LibertySecMech](#)]) indicating their intended purpose.
- 642 e. If the issuer is unable to generate a necessary token, it MUST include an empty `<sec:Token>` element with
643 the `ref` attribute set to the value `urn:liberty:disco:tokenref:ObtainFromIDP`

644 2.3.3.2. ID-WSF EPR Specifics

645 The information contained in an ID-WSF EPR is sufficient for making invocations for service instances. In other words,
646 the information contained in this group together with the abstract WSDL specified by the ServiceType URI is sufficient
647 to logically compute concrete WSDL with the rule set specified below.

648 The `<wsa:Address>` element of the ID-WSF EPR contains the URI of the endpoint. For SOAP-over-HTTP endpoints,
649 the URI scheme MUST be "http" or "https."

650 Use of this addressing form implies `<wsdl:binding>` and `<wsdl:service>` elements according to the following
651 rules (i.e., the concrete WSDL can be logically computed given the abstract WSDL and an ID-WSF EPR):

- 652 • The `<wsdl:binding>` contains a `<wsdlsoap:binding>` element. This specifies that the SOAP binding for
653 WSDL is being used.
- 654 • The `style` attribute of the `<wsdlsoap:binding>` element is "document".
- 655 • The `transport` attribute of the `<wsdlsoap:binding>` element is `http://schemas.xmlsoap.org/soap/http`.
- 656 • The abstract WSDL corresponding to the `<ServiceType>` MUST contain a single `<portType>` element. The
657 `<wsdl:binding>` element provides bindings for the operations specified in this `<wsdl:portType>`. Each op-
658 eration binding includes an input element and an output element, each containing a single `<wsdlsoap:body>`
659 element. The `use` attribute of the `<wsdlsoap:body>` elements is "literal".
- 660 • The `location` attribute of `<wsdlsoap:address>` is equal to `<wsa:Address>`.
- 661 • All other optional elements and attributes are not specified and thus default to the SOAP binding of WSDL.

662 2.3.3.3. Security Mechanism Specifics

663 With respect to `<SecurityMechID>` URIs: these URIs denote the security mechanisms supported by the service
664 instance described by the ID-WSF EPR. Other specifications, such as [LibertySecMech] define the actual security
665 mechanisms along with their identifying URIs. These security mechanisms refer to the way a WSC authenticates to a
666 WSP ("peer-entity authentication") and/or provides message security ("data-origin authentication").

667 An ID-WSF EPR SHOULD list all of the security mechanisms that the service instance supports in order of preference.
668 I.e. the most preferred security mechanism is first in the list, the next is the second-most preferred, and so on.

669 In the case that the set of supported security mechanisms varies with respect to endpoint address(es) and/or WSDL
670 binding, the system entity constructing the ID-WSF EPRs MUST construct multiple ID-WSF EPRs with each ID-WSF
671 EPR separately representing each supported mapping.

672 Also, any single `<SecurityMechID>` URI MUST NOT appear in more than one of the `<SecurityContext>` ele-
673 ments of any of the ID-WSF EPRs so constructed. In other words, each service instance may only specify one WSDL
674 binding per supported security mechanism. If a sequence of ID-WSF EPRs is constructed, then the ID-WSF EPRs
675 SHOULD appear in the order of the constructor's preference, and the `<SecurityContext>` elements within each
676 should be in order of preference, as should the `<SecurityMechID>` elements within them—with the most preferred
677 item listed first in each case.

678 For example: many web servers will require a different endpoint URI to be used for SOAP/HTTP clients authenticating
679 using client TLS certificates than for clients which authenticate in some other fashion. See Example 4.

680 2.3.3.4. Action Specifics

681 With respect to `<Action>` URIs: these URIs denote the interfaces supported by the service instance described by the
682 ID-WSF EPR. The service specific specifications, such as this document, define the actual interfaces along with their
683 identifying URIs.

684 An ID-WSF EPR SHOULD NOT list actions unless the service instance at this endpoint does not support the complete
685 set of service interfaces. In such a case, the ID-WSF EPR SHOULD list all of the available interfaces.

686 There is no preference or other significance to the ordering of the <Action> URIs.

687 2.3.3.5. Identity Invocation Context specifics

688 The invocation of an ID-WSF service can carry several identities as documented in [LibertySOAPBinding]. These
689 identities include the `Sender`, the `InvocationIdentity`, the `TargetIdentity`, and the `Recipient`.

690 The Discovery Service, when minting ID-WSF EPRs, works to maintain the same identity invocation context that was
691 used to invoke it such that the same logical `Sender`, `InvocationIdentity` and `TargetIdentity` are carried forth
692 in messages invoked through the minted EPR. Of course, the `Recipient` of the subsequent invocation will be different
693 as it will be the WSP to which this EPR points.

694 The Discovery Service generates security and/or identity tokens to convey these identities in the minted ID-WSF EPR.
695 These tokens are placed into the <sec:Token> elements within <SecurityContext> element.

696 In preparing the necessary tokens to carry forth these identities, the Discovery Service may have to perform identity
697 translations to obtain pseudonymous identifiers for the interested parties at the intended `Recipient`.

698 The rules for when and how the tokens are generated when the ID-WSF EPR is minted by the Discovery Service (in
699 response to a `DiscoveryQuery` operation, see [Section 3.3](#)), are as follows:

- 700 • If the Principal, whose discovery resource is being queried, is the same as the invocation identity of the Discov-
701 eryQuery operation — *i.e.*, there is not a <sb:TargetIdentity> header block on the <Query> message — then
702 the same effective invocation identity MUST be expressed by the Discovery Service's resultant selected security
703 tokens for the invocation identity (which are embedded in <sec:Token> element(s) in the
704 <SecurityContext> element in the ID-WSF EPR's <wsa:Metadata> element)

705 **Note**

706 Since the security tokens usually carry the identity of the `Sender` and that of the
707 `InvocationIdentity` it is possible that a single <SecurityContext> may include multiple security
708 tokens identifying each of the parties.

- 709 • Else, if the Principal, whose discovery resource is being queried, is not the same as the invocation identity of the
710 `DiscoveryQuery` operation — *i.e.*, a <sb:TargetIdentity> header block appears in the header of the
711 <Query> message — then the invocation identity to be conveyed in the ID-WSF EPR is expressed as denoted in
712 the bullet item above, and additionally, a identity token denoting the target identity (per [LibertySecMech] and
713 [LibertySecMech20SAML]) is also embedded in a <sec:Token> element in the <SecurityContext> element
714 in the ID-WSF EPR's <wsa:Metadata> element.

715 The rules for when and how the above identity tokens are included as above when the ID-WSF EPR is minted by a
716 WSC or WSP (refer to [Section 2.3.3.1](#), above, for context), are as follows:

- 717 • If the intended target identity is to be the same as that of the intended invocation identity, then the intended invocation
718 identity MUST be expressed in the minted ID-WSF EPR as detailed in the rules above (first bullet item).
- 719 • If the intended target identity is to be different than the intended invocation identity, then the intended invocation
720 identity and the intended target identity both MUST be expressed in the minted ID-WSF EPR as detailed in the
721 rules above (second bullet item).

722 The recipient of an ID-WSF EPR distinguishes between the various tokens contained within a <sec:Token> element
723 via the `usage` attribute as follows:

- 724 • A token with the usage attribute set to urn:liberty:security:tokenusage:2006-02:SecurityToken
 725 contains a security token that MUST be placed into the <wsse:Security> header block (according to [Liberty-
 726 SecMech] and its related profiles) when a message is generated for the target of the ID-WSF EPR.
- 727 If multiple <sec:Token>s are included in a single <ds:SecurityContext>, they MUST ALL be placed into
 728 the same <wsse:Security> header block.
- 729 • A token with the usage attribute set to urn:liberty:security:tokenusage:2006-08:
 730 TargetIdentity contains an identity token that MUST be placed into the <sb:TargetIdentity> header block
 731 (according to [LibertySOAPBinding]) when a message is generated for the target of the ID-WSF EPR.

732 2.3.4. Effective Web Services Addressing EPR

733 The net effect of the ID-WSF profile of the EPR is as if the EndpointReferenceType were defined with the schema
 734 fragment below. There are several things to note about this schema including:

- 735 • There is no normative XML schema defined as such, this is just an approximation of what the schema could look
 736 like.
- 737 • While the elements within the <Metadata> element appear to be ordered, they can all appear in any order and can
 738 have other elements appear between the listed elements. This is why they are contained within a multi-occurrence
 739 <xs:choice>.
- 740 • Four attributes have been added to the EPR element itself: the notOnOrAfter timestamp and three different IDs
 741 (for use in different circumstances).
- 742 • Seven sub-elements were added to the <Metadata> element.
- 743 • The <Metadata> sub-element <disco:ProviderID> MUST appear exactly once in an ID-WSF EPR, even
 744 though the schema below does not enforce that requirement (because of a limitation in XML Schema -- or perhaps
 745 in the author's understanding of XML schema).

746 The <Metadata> sub-elements: <sbf:Framework>, and <disco:ServiceType> MUST appear at least once
 747 in an ID-WSF EPR, even though the schema below does not enforce that requirement (because of a limitation in
 748 XML Schema -- or perhaps in the author's understanding of XML schema).

```

749 ...
750 <xs:element name="EndpointReference" type="tns:EndpointReferenceType"/>
751 <xs:complexType name="EndpointReferenceType" mixed="false">
752   <xs:sequence>
753     <xs:element name="Address" type="tns:AttributedURIType"/>
754     <xs:element name="ReferenceParameters"
755       type="tns:ReferenceParametersType" minOccurs="0"/>
756     <xs:element ref="tns:Metadata" minOccurs="0"/>
757     <xs:any namespace="##other" processContents="lax"
758       minOccurs="0" maxOccurs="unbounded"/>
759   </xs:sequence>
760   <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional" />
761   <xs:attribute ref="wsu:Id" use="optional" />
762   <xs:attribute name="reqRef" type="xs:string" use="optional" />
763   <xs:anyAttribute namespace="##other" processContents="lax" />
764 </xs:complexType>
765
766 <xs:complexType name="ReferenceParametersType" mixed="false">
767   <xs:sequence>
768     <xs:any namespace="##any" processContents="lax"
769       minOccurs="0" maxOccurs="unbounded"/>
770   </xs:sequence>
771   <xs:anyAttribute namespace="##other" processContents="lax" />
772 
```

```

773 </xs:complexType>
774
775 <xs:element name="Metadata" type="tns:MetadataType"/>
776 <xs:complexType name="MetadataType" mixed="false">
777   <xs:sequence>
778     <xs:choice minOccurs="0" maxOccurs="unbounded">
779       <xs:element ref="disco:Abstract" minOccurs="0" />
780       <xs:element ref="sbf:Framework" maxOccurs="unbounded" />
781       <xs:element ref="disco:ProviderID" />
782       <xs:element ref="disco:ServiceType" maxOccurs="unbounded" />
783       <xs:element ref="disco:SecurityContext" minOccurs="0" maxOccurs="unbounded" />
784       <xs:element ref="disco:Options" minOccurs="0" maxOccurs="unbounded" />
785       <xs:element ref="disco:Action" minOccurs="0" maxOccurs="unbounded" />
786       <xs:any namespace="##other" processContents="lax"
787         minOccurs="0" maxOccurs="unbounded" />
788     </xs:choice>
789   </xs:sequence>
790   <xs:anyAttribute namespace="##other" processContents="lax" />
791 </xs:complexType>
792

```

793 Example 3. Effective ID-WSF EPR Schema

794 2.3.5. Example Liberty ID-WSF EPRs

```

795
796 <wsa:EndpointReference
797   notOnOrAfter="2005-08-15T23:18:56Z"
798   ...>
799   <wsa:Address>
800     http://profile-provider.com/profiles/someFoobarProfileAddr
801   </wsa:Address>
802
803   <wsa:Metadata>
804     <ds:Abstract>
805       This is a personal profile containing common name information.
806     </ds:Abstract>
807
808     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
809
810     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
811
812     <sbf:Framework version="2.0" />
813
814     <ds:SecurityContext>
815       <ds:SecurityMechID>
816         urn:liberty:security:2006-08:ClientTLS:SAMLV2
817       </ds:SecurityMechID>
818
819       <ds:SecurityMechID>
820         urn:liberty:security:2005-02:ClientTLS:SAML
821       </ds:SecurityMechID>
822
823       <sec:Token wsu:id="_10"
824         usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
825         <!-- some security token goes here -->
826       </sec:Token>
827     </ds:SecurityContext>
828
829     <ds:SecurityContext >
830       <ds:SecurityMechID>
831         urn:liberty:security:2005-02:ClientTLS:X509
832       </ds:SecurityMechID>
833
834       <sec:Token wsu:id="_20"
835         usage="urn:liberty:security:tokenusage:2006-08:InvocationIdentity">
836         <!-- Identity Token goes here -->

```

```

837         </sec:Token>
838     </ds:SecurityContext>
839
840     <ds:Options>
841         <ds:Option>urn:liberty:id-sis-pp</ds:Option>
842         <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
843         <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
844         <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
845     </ds:Options>
846 </wsa:Metadata>
847 </wsa:EndpointReference>
848
849 <wsa:EndpointReference
850     notOnOrAfter="2005-08-15T23:18:56Z"
851     ...>
852 <wsa:Address>
853     http://profile-provider.com/profiles/anotherFoobarProfileEndpointAddr
854 </wsa:Address>
855
856 <wsa:Metadata>
857     <ds:Abstract>
858         This is a personal profile containing common name information.
859     </ds:Abstract>
860
861     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
862
863     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
864
865     <sb:Framework version="2.0" />
866
867     <ds:SecurityContext>
868         <ds:SecurityMechID>
869             urn:liberty:security:2006-08:TLS:SAMLV2
870         </ds:SecurityMechID>
871
872         <sec:Token ref="_10" usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" />
873     </ds:SecurityContext>
874
875     <ds:Options>
876         <ds:Option>urn:liberty:id-sis-pp</ds:Option>
877         <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
878         <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
879         <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
880     </ds:Options>
881 </wsa:Metadata>
882 </wsa:EndpointReference>
883

```

884 **Example 4. Instantiated List of ID-WSF EPRs Illustrating Multiple <SecurityContext> Elements with both Embedded**
885 **and Referenced <sec:Token> Elements**

886 2.4. Service Metadata

887 The discovery Service mints the ID-WSF EPRs described in the previous section using information provided by the
888 WSP in the WSP's registered Service Metadata.

889 2.4.1. Service Metadata element

890 The Service Metadata is used to describe a single instance of a service hosted by a WSP as it applies to all principals
891 (i.e., the principal independent information related to an instance).

892 This single instance can include multiple endpoints, multiple security mechanisms, and even multiple service types.
893 Multiple service types SHOULD only be included in a single Service Metadata element if the WSP considers those
894 service types to be different versions of the same service (for example, *urn:liberty:disco:2006-08* and *urn:liberty:*
895 *disco:2003-08* are two different versions of the Liberty ID-WSF Discovery Service).

896 Most of the fields present in the Service Metadata have the same purpose and meaning as the elements of the same
 897 name in the ID-WSF EPR (as this is where the Discovery service gets those elements for the ID-WSF EPR).
 898 When fields permit multiple values, the order of entries in the SvcMD is significant with higher preference items coming
 899 first. This comes into plan should the WSC request a subset of the possible results when querying the Discovery Service
 900 (in which case the entries with the higher preference -- those listed first -- would be used to mint the ID-WSF EPRs in
 901 the response).

```

902
903 <!-- Service Metadata (SvcMD) - metadata about service instance -->
904
905 <xs:element name="SvcMD" type="SvcMetadataType"/>
906 <xs:complexType name="SvcMetadataType">
907   <xs:sequence>
908     <xs:element ref="Abstract" />
909     <xs:element ref="ProviderID" />
910     <xs:element ref="ServiceContext" maxOccurs="unbounded" />
911   </xs:sequence>
912   <xs:attribute name="svcMDID" type="xs:string" use="optional" />
913 </xs:complexType>
914
915 <!-- ServiceContext - describes service type/option/endpoint context -->
916 <xs:element name="ServiceContext" type="ServiceContextType"/>
917 <xs:complexType name="ServiceContextType">
918   <xs:sequence>
919     <xs:element ref="ServiceType" maxOccurs="unbounded" />
920     <xs:element ref="Options" minOccurs="0" />
921     <xs:element ref="EndpointContext" maxOccurs="unbounded" />
922   </xs:sequence>
923 </xs:complexType>
924
925 <!-- EndpointContext - describes endpoints used to access service -->
926 <xs:element name="EndpointContext" type="EndpointContextType" />
927 <xs:complexType name="EndpointContextType">
928   <xs:sequence>
929     <xs:element ref="Address" maxOccurs="unbounded" />
930     <xs:element ref="sbf:Framework" maxOccurs="unbounded" />
931     <xs:element ref="SecurityMechID" maxOccurs="unbounded" />
932     <xs:element ref="Action" minOccurs="0" />
933     <xs:element ref="Action" maxOccurs="unbounded" />
934   </xs:sequence>
935 </xs:complexType>
936
937 <!-- SvcMD ID element used to refer to Service Metadata elements -->
938 <xs:element name="SvcMDID" type="xs:string" />
939

```

940 **Figure 13. Service Metadata — Schema Fragment**

941 **2.4.1.1. svcMDID**

942 The `svcMDID` attribute is a unique identifier assigned by the Discovery Service during service metadata registration
 943 and used on later principal registrations.

944 The value of the identifier **MUST** be unique across all registered service metadata for the registering WSP at the DS
 945 and **MAY** be unique across all WSPs.

946 **2.4.1.2. Abstract**

947 A text description of the service.

948 **2.4.1.3. ProviderID**

949 The URI of the provider of this service instance.

950 **2.4.1.4. ServiceContext**

951 The <ServiceContext> describes the set of service versions and options that are available at a particular set of
952 endpoints. A Service Metadata description may have multiple <ServiceContext>s when they support a particular
953 version (or set of options) of the service at one set of endpoints and another version at a different set of endpoints.

954 The elements contained within a <ServiceContext>s are discussed below:

955 **2.4.1.4.1. ServiceType**

956 The URI of which defines the type of service.

957 Note that there may be multiple service types defined in a service metadata indicating that multiple distinct services
958 are available at the same endpoint. This typically occurs when multiple versions of the same general type of service
959 are available at the same endpoint although it is possible that very different services could be at the same endpoint.

960 **2.4.1.4.2. Option**

961 The Option(s) supported by this service instance.

962 Multiple options may be specified indicating that this service instance supports all of the listed options.

963 **2.4.1.4.3. EndpointContext**

964 While not explicitly in an ID-WSF EPR, the contents of this element show up in various locations within the IPR and/
965 or guide the generation of the contents of the EPR.

966 Multiple <EndpointContext> elements may appear if the same service is available via different, incompatible com-
967 binations of the contents (such as a TLS and a non-TLS endpoint at different addresses).

968 The sub-elements include:

969 **2.4.1.4.3.1. Address**

970 A URI describing the address to which messages should be sent to communicate with this provider.

971 If multiple addresses are specified they are all considered equally valid addresses for this same service (such that if a
972 Discovery Service were to mint all of the possible EPRs for this case, there would be a separate EPR for each address
973 specified since an EPR can only include a single address).

974 In the case where the Discovery service has been asked to mint a subset of the possible EPRs (see [Section 3.3](#)), the
975 Discovery service is free to select any of the specified addresses using whatever local policy it chooses.

976 **2.4.1.4.3.2. Framework**

977 The SOAP Bindings ([[LibertySOAPBinding](#)]) <sbf:Framework> element describing the version of the ID-WSF
978 framework supported at this endpoint..

979 Multiple <Framework> elements may be specified if they can be used at each of the <Address> URIs within this
980 <EndpointContext>.

981 **2.4.1.4.3.3. SecurityMechID**

982 The Security Mechanism URI(s) (defined in [[LibertySecMech](#)] and its related profiles) supported by this endpoint.

983 Multiple <SecurityMechID> elements may be specified indicating that any of these mechanisms can be used at this
984 endpoint.

985 Note that while a particular security mechanism may need a particular form of a security token, the registering WSP
986 cannot provide such tokens. It is up to the Discovery service to mint the necessary token, or indicate to the WSC that
987 they need to obtain the token from their IdP.

988 **2.4.1.4.3.4. Action**

989 The URI indicating the supported service action at this endpoint. This is typically used when only a sub-set of the entire
990 service's operations are available at this endpoint.

991 Multiple <Action> elements may be specified to indicate that there are multiple operations available at this endpoint.

992 If no <Action> element is specified, all service operations are available at this endpoint.

993 **2.4.2. Minting ID-WSF EPRs based upon Service Metadata**

994 Service Metadata is stored in the Discovery Service in order to guide the minting of ID-WSF EPRs by the Discovery
995 Service in response to queries from WSCs.

996 One can visualize that the entire set of elements within a single Service Metadata can result in a large number of possible
997 EPRs based upon the possible combinations of those elements.

998 The Discovery Service MUST mint ID-WSF EPRs as if the following process took place (there is NOT a normative
999 requirement to implement this exact process, just a requirement that the results generated by whatever process is used
1000 by the DS MUST result in the set of data that would result from this process).

- 1001 1. Eliminate portions of the Service Metadata that do not conform to the search requirements (such as unsupported
1002 (by the WSC) security mechanisms or framework versions, or undesired service types).
- 1003 2. If an <EndpointContext> element had all occurrences of a given sub-element (such as <Framework>) elimi-
1004 nated, eliminate the context.
- 1005 3. For each remaining <Address> element within a remaining <EndpointContext> element, an EPR SHOULD
1006 be minted.
- 1007 4. For each EPR, assign one <Address> element to the <wsa:Address> element in the EPR and use the rest of
1008 the <EndpointContext> that contained this address to build the necessary <Metadata> sub-elements for the
1009 ID-WSF EPR (. <SecurityContext>(s), <Action>(s), <Framework>(s), etc.).
- 1010 5. Fill out the rest of the ID-WSF EPR using the service wide elements (<Abstract>, <ProviderID>,
1011 <ServiceType>(s), etc.).
- 1012 6. If necessary, generate any security and/or identity tokens and place them into the appropriate
1013 <SecurityContext> element(s).

1014 The set of EPRs generated by this process may be further restricted by the request parameters on the DiscoveryQuery
1015 operation, see [Section 3.3](#)).

1016 **2.4.3. Service Metadata Example**

1017 Some examples to help show how service metadata works.

1018 **2.4.3.1. A simple service**

1019 This is an example of a simple service that has a single endpoint, supports a single framework version (2.0), and only
1020 supports a single security mechanism.

```
1021
1022     <ds:svcMD svcMDID="1234">
1023         <ds:Abstract>This is a simple service metadata definition</ds:abstract>
1024         <ds:ProviderID>http://simpler.providers.com</ds:ProviderID>
1025         <ds:ServiceContext>
1026             <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1027             <ds:EndpointContext>
1028                 <ds:Address>https://simple.providers.com/PP</ds:Address>
1029                 <sb:Framework version="2.0"/>version="2.0"/>
1030             <ds:SecurityMechID>
1031                 urn:liberty:security:2003-08:TLS:Bearer
1032             </ds:SecurityMechID>
1033         </ds:EndpointContext>
1034     </ds:ServiceContext>
1035 </ds:SvcMD>
```

1036 **Figure 14. Service Metadata example: A simple service**

1037 2.4.3.2. A complex service

1038 This is an example of a service metadata definition with a number of complex attributes including:

- 1039 • Multiple service versions **and** multiple framework versions on the same endpoint.
- 1040 • There are two service contexts, one for one version of the service and one for a different version of the service. So,
1041 for example, the 2003-08 version of the service is only available at the URL *https://old.providers.com/PP* and only
1042 for framework version *1.1*
- 1043 • Multiple interfaces on different endpoints with different security mechanisms
- 1044 • There are multiple, redundant, addresses for the TLS endpoint for the *2007-11* version of the service.

```

1045
1046 <!-- Service Metadata Example: A complex service -->
1047 <ds:SvcMD svcMDID="4567">
1048   <ds:Abstract>This is a complex service metadata definition</ds:abstract>
1049   <ds:ProviderID>http://complex.providers.com</ds:ProviderID>
1050   <ds:ServiceContext>>
1051     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1052     <ds:EndpointContext>
1053       <ds:Address>https://old.providers.com/PP</ds:Address>
1054       <sb:Framework version="1.1" />
1055       <ds:SecurityMechID>
1056         urn:liberty:security:2003-08:TLS:Bearer
1057       </ds:SecurityMechID>
1058     </ds:EndpointContext>
1059   </ds:ServiceContext>>
1060   <ds:ServiceContext>>
1061     <ds:ServiceType>urn:liberty:pp:2007-11</ds:ServiceType>
1062     <ds:EndpointContext>
1063       <ds:Address>https://complex.providers.com/PP</ds:Address>
1064       <ds:Address>https://backup.complex.providers.com/PP</ds:Address>
1065       <sb:Framework version="2.0" />
1066       <ds:SecurityMechID>
1067         urn:liberty:security:2003-08:TLS:Bearer
1068       </ds:SecurityMechID>
1069     </ds:EndpointContext>
1070     <ds:EndpointContext>
1071       <ds:Address>http://complex.providers.com/PP</ds:Address>
1072       <sb:Framework version="2.0" />
1073       <ds:SecurityMechID>
1074         urn:liberty:security:2003-08:null:SAMLV2
1075       </ds:SecurityMechID>
1076     </ds:EndpointContext>
1077   </ds:ServiceContext>>
1078 </ds:SvcMD>

```

Figure 15. Service Metadata example: A complex service

1080 2.4.3.3. Another complex service

1081 This is an example of a service metadata definition where the service has some of its operations at one endpoint and
 1082 others at a different endpoint (thus splitting the service operations across different instances).

1083 This service is still defined with a single service context since the endpoints all expose the same service type.

```
1084
1085 <!-- Service Metadata Example: A simple service -->
1086 <ds:SvcMD svcMDID="8901">
1087   <ds:Abstract>Another example complex service</ds:abstract>
1088   <ds:ProviderID>http://split.providers.com</ds:ProviderID>
1089   <ds:ServiceContext>>
1090     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1091     <ds:EndpointContext>
1092       <ds:Address>https://cluster1.split.providers.com/PP</ds:Address>
1093       <sb:Framework version="2.0" />
1094       <ds:SecurityMechID>
1095         urn:liberty:security:2003-08:TLS:Bearer
1096       </ds:SecurityMechID>
1097       <ds:Action>urn:liberty:pp:2003-08:Query</ds:Action>
1098     </ds:EndpointContext>
1099     <ds:EndpointContext>
1100       <ds:Address>https://writer.split.providers.com/PP</ds:Address>
1101       <sb:Framework version="2.0" />
1102       <ds:SecurityMechID>
1103         urn:liberty:security:2003-08:TLS:Bearer
1104       </ds:SecurityMechID>
1105       <ds:Action>urn:liberty:pp:2003-08:Modify</ds:Action>
1106     </ds:EndpointContext>
1107   </ds:ServiceContext>>
1108 </ds:SvcMD>
```

1109

Figure 16. Service Metadata example: Another complex service

1110 3. Discovery Service

1111 A Discovery Service is a web service providing both identity based and non-identity based operations.

1112 The identity based Discovery Service interfaces facilitate requesters' discovery of identity service instances on a per-
1113 identity basis, and acquisition of ID-WSF Endpoint References (ID-WSF EPRs) "pointing" to the discovered service
1114 instances. These ID-WSF EPRs provide requesters with the information necessary to invoke discovered service in-
1115 stances.

1116 The non-identity based Discovery Service interfaces provide a WSP with principal-independent management of their
1117 metadata stored at the Discovery Service (which is, through an identity-based interface, associated with a principal).

1118 Thus in an abstract sense, the Discovery Service is essentially a web service interface to per-identity "discovery re-
1119 sources", each of which can be viewed as a registry of ID-WSF EPRs. The notion of "discovery resources" is an abstract
1120 way of referring to what are concretely "identity-indexed Discovery Service instances".

1121 The Discovery Service can also be used as a non-identity service to discover and obtain ID-WSF EPRs for non-identity
1122 services. For example, the Discovery Service could be used to locate the available Authentication Services before a
1123 principal identity has been established.

1124 Entities can register ID-WSF EPRs, pointing to their identity services, with a discovery resource, and this will allow
1125 other entities to discover them. A common use case is that a Principal places references (aka ID-WSF EPRs) to his or
1126 her personal profile, calendar, and so on, in a discovery resource so that they may be discovered by other entities, e.g., |
1127 web service providers who wish to provide the Principal with value-added services.

1128 When invoked as an identity service, the act of discovering service instances is implicitly on a per-identity basis. This
1129 occurs in a number of fashions in ID-WSF including:

- 1130 • When a Principal authenticates to a service provider using a SAMLv2 profile (or similarly via
1131 ID-FF), the identity provider conveys, within the authentication assertion, an ID-WSF EPR
1132 pointing explicitly to the *Principal's* discovery service resource, which the SP may then use to
1133 discover the Principal's various services.
- 1134 • A Principal's (LUAD-)WSC authenticates via the Authentication Service (see [LibertyAuthn]),
1135 which will likely return an ID-WSF EPR for the Principal's Discovery Service resource.
- 1136 • Any Identity Token (see [LibertySecMech]), or security token may contain a Discovery Service
1137 bootstrap ID-WSF EPR (see Section 4: Discovery Service ID-WSF EPR conveyed via a Security
1138 Token) which contains the necessary information to access the Principal's Discovery Service
1139 resource.

1140 The Discovery service is identified by ID-WSF EPRs, which themselves have been crafted (typically by an identity
1141 provider) such that they identify the discovery service resource (aka Discovery Service instance) mapped to the Prin-
1142 cipal in question.

1143 The Discovery Service is intended to be used in conjunction with other ID-WSF specifications. For example, security
1144 mechanisms are not specified here, because they are defined in [LibertySecMech]. At the same time, the Discovery
1145 Service is specified such that it could be used with other security mechanisms, not yet defined.

1146 The Discovery Service is designed to be describable by WSDL [WSDLv1.1], and an abstract WSDL definition is
1147 included in this document, see Appendix B: Discovery Service WSDL . This WSDL document defines two "WSDL
1148 operations" for the Discovery Service. The first is the *DiscoveryQuery* operation. This operation returns an enumeration
1149 of ID-WSF EPRs for a given search criteria.

1150 To enforce access control policies, security tokens may need to be presented by the client when interacting with a
1151 Discovery Service instance. While the definition of these security tokens is outside the scope of this specification, it

1152 is common for the same provider that is hosting the Discovery Service to also be the entity that generates the security
 1153 tokens necessary to access the service. To avoid extra network round-trips, arrangements are made here so that security
 1154 tokens may be provided as part of the Discovery Service lookup response.

1155 3.1. Service URIs

1156 **Table 1. Discovery Service URIs**

Use	URI
Service Type	<i>urn:liberty:disco:2006-08</i>
Query wsa: Action	<i>urn:liberty:disco:2006-08:Query</i>
QueryResponse wsa: Action	<i>urn:liberty:disco:2006-08:QueryResponse</i>
SvcMDAssociationAdd wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationAdd</i>
SvcMDAssociationAddResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationAddResponse</i>
SvcMDAssociationQuery wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationQuery</i>
SvcMDAssociationQueryResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationQueryResponse</i>
SvcMDAssociationDelete wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationDelete</i>
SvcMDAssociationDeleteResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDAssociationDeleteResponse</i>
SvcMDQuery wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDQuery</i>
SvcMDQueryResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDQueryResponse</i>
SvcMDRegister wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDRegister</i>
SvcMDRegisterResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDRegisterResponse</i>
SvcMDReplace wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDReplace</i>
SvcMDReplaceResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDReplaceResponse</i>
SvcMDDelete wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDDelete</i>
SvcMDDeleteResponse wsa: Action	<i>urn:liberty:disco:2006-08:SvcMDDeleteResponse</i>

1157 3.2. Status Codes

1158 The following status code strings are defined:

- 1159 • *OK*: message processing succeeded

- 1160 • *Failed*: general failure code
- 1161 • *Forbidden*: the request was denied based on policy
- 1162 • *Duplicate*: the request was denied because it would result in duplicate data in the service
- 1163 • *LogicalDuplicate*: the request was denied because it would result in logically duplicate data in the service
- 1164 • *NoResults*: the query had no matching results
- 1165 • *NotFound*: the specified item(s) were not found

1166 These strings are expected to appear in the "code" attribute of <Status> elements used in SOAP-bound Discovery
1167 Service protocol messages [LibertySOAPBinding]. Specific uses for the status codes are defined in the processing rules
1168 for individual messages. The "ref" attribute on the <Status> element is not used in this specification, so it MUST
1169 NOT appear on Status elements in Discovery Service protocol messages. The contents of the comment attribute are
1170 not defined by this specification, but it may be used for additional descriptive text intended for human consumption
1171 (for example, to carry information that will aid debugging).

1172 3.3. Operation: *DiscoveryQuery*

1173 The *DiscoveryQuery* WSDL operation enables a requester to obtain an enumeration of ID-WSF EPRs (see [Section 2:](#)
1174 [Discovery Service Information Model](#)) — the requester sends a <Query> message and receives a
1175 <QueryResponse> message in return. Also, because a provider hosting a Discovery Service may also be playing other
1176 roles on behalf of Principals (such as a *Policy Decision Point* or an *Authentication Authority*), the *DiscoveryQuery*
1177 operation can also function as a security token service, providing the requester with an efficient means of obtaining
1178 security tokens that may be necessary to invoke service instances described in the <QueryResponse>.

1179 3.3.1. wsa:Action values for DiscoveryQuery Messages

1180 <Query> messages MUST include a <wsa:Action> SOAP header with the value of urn:liberty:disco:
1181 2006-08:Query.

1182 <QueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of urn:liberty:
1183 disco:2006-08:QueryResponse.

1184 3.3.2. <Query> Message

1185 A <Query> request is an attempt to retrieve ID-WSF EPRs suitable for use in the same identity context that was used
1186 to make the request. In particular, the Target Identity (See [LibertySOAPBinding]), if applicable, is used to restrict the
1187 results to just those for the specified principal. The Invocation Identity will be verified against an access control list to
1188 ensure that they have access to the requested results.

1189 A <Query> request message is empty in the minimal case. Such a request indicates the requester is requesting all
1190 available ID-WSF EPRs, regardless of security mechanisms or service types. The result set is dependant upon the local
1191 access control policies of the discovery service instance.

1192 Alternatively, a request can be qualified with a set of <RequestedService> elements, which enables the requester
1193 to specify that all ID-WSF EPRs returned must be offered via one or more service instances complying with the specified
1194 search criteria. For each <RequestedService> specified, the requester specifies the search criteria for the DS to use
1195 in determining if there is a matching instance. The search criteria includes zero or more of any of the following:

- 1196 • <ServiceType> the requested type of service. Service Type URIs are defined by the individual service specifi-
1197 cations and contain both service class and service versioning information.

- 1198 Multiple <ServiceType>s MAY be specified in a single <RequestedService>s element in order to allow the
1199 WSC to specify what the WSC considers to be different versions of the same service.
- 1200 When multiple entries are listed, the order of such entries is an indication of the preference as to which
1201 <ServiceType> the WSC would prefer to see in the results, with the first being the most preferred. This typically
1202 only impacts a request where the WSC indicates that they only want a subset of the results returned (see
1203 resultsType below).
- 1204 When a request results in multiple ID-WSF EPRs in a response, the preference order specified by the WSC on the
1205 request MAY have no impact on the order of results returned by the Discovery Services. The Discovery Service is
1206 free to return the results of the request in whatever order it chooses.
- 1207 If not specified, then any service instance would be considered a match for this criteria.
- 1208 A service instance must support at least one of the specified service types in order to be considered a match for this
1209 criteria.
- 1210 • <ProviderID> the requested provider ID(s). This is used when the WSC wants to communicate with a particular
1211 WSP. Frequently such requests are made without specifying a <ServiceType> element in the request, but doing
1212 so is not prohibited.
- 1213 If not specified, then any service instance would be considered a match for this criteria.
- 1214 A service instance must contain at least one of the specified providerIDs in order to be considered a match for this
1215 criteria.
- 1216 The order of the <ProviderID> elements is an indication as to the preference of the requester with the first such
1217 element being the most desired (declining preference order). The Discovery Services is free to return the results of
1218 the request in whatever order it chooses.
- 1219 • <Options> — an optional multi-occurrence element defining options SETs desired for the service.
- 1220 If not specified, any service instance will be considered a match for this criteria.
- 1221 An option SET is defined within each <Options> element and contains a list of the desired options. The service
1222 instance MUST support ALL of the options within the option SET in order to be considered a match for this request.
- 1223 If more than one <Options> element is specified (thus defining multiple option SETs), service instances that match
1224 ANY of the SETS are considered a match for this request. As noted above, to match a SET, you have to match
1225 ALL of the entries within the SET.
- 1226 Service instance EPRs registered without an <Options> element are always considered a match from the point of
1227 view of any possible <Options> search criteria.
- 1228 The order of the <options> elements is an indication as to the preference of the requester with the first such
1229 element being the most desired (declining preference order). The Discovery Services is free to return the results of
1230 the request in whatever order it chooses.
- 1231 • <SecurityMechID> - an optional multi-occurrence element specifying the security mechanism identifier(s) (see
1232 [LibertySecMech]) that the WSC is willing to use to invoke the WSP. If not specified, any security mechanism
1233 registered for a service will be considered a match for this criteria.
- 1234 A service instance MUST support at least one of the requested security mechanisms in order to be considered a
1235 match for this request.

- 1236 The order of the <SecurityMechID> elements is an indication as to the preference of the requester with the first
1237 such element being the most desired (declining preference order). The Discovery Services is free to return the
1238 results of the request in whatever order it chooses.
- 1239 • <Framework> — an optional multi-occurrence element specifying the framework description(s) supported by the
1240 WSC.
- 1241 If not specified, the Discovery Service SHOULD use the value of the framework description used in the ID-WSF
1242 framework layer for the current request (e.g., if the call to the Discovery Service was made using an ID-WSF version
1243 2.0 message the request SHOULD be treated as if a <disco:Framework> element was present and contained the
1244 value specified in the <sbf:Framework> SOAP header.
- 1245 Multiple <disco:Framework> elements MAY be specified, indicating that the WSC has the capability to support
1246 ANY of the specified versions. The order of elements in such a case indicates the WSC's preference with the most
1247 preferred coming first.
- 1248 Note that while both the <disco:Framework> and the <sbf:Framework> elements are of the same type
1249 (<sbf:FrameworkType>), the elements themselves are in different namespaces. The element within the
1250 <RequestedService> is in the Discover Service Namespace, while the element within any ID-WSF EPRs and
1251 the SOAP header block on an ID-WSF message are in the SOAP Bindings namespace.
- 1252 • <Action> — an optional multi-occurrence element specifying the wsa:Action value(s) for the interfaces of the
1253 service that the WSC intends to make use of.
- 1254 If not specified, the Discovery Service SHOULD treat this request as a request for all of the interfaces at the
1255 requested specified instance.
- 1256 Unlike the other sub-elements of the <RequestedService> element, if multiple <Action> elements are specified
1257 it indicates that the WSC intends to invoke **all** of the specified interfaces and the Discovery Service SHOULD
1258 return the set of EPRs that are necessary to reach the complete set of specified interfaces.
- 1259 Services registered without an <Action> element (which is the norm) are treated as exposing **all** of the interfaces
1260 defined for that type of service.
- 1261 The Discovery Service will return the set of EPRs for service instances that intersect with the search criteria specified
1262 in the <RequestedService> element. This may result in a single EPR in the response or it may result in a multitude
1263 of EPRs, depending upon the search criteria and the service instance definitions available (see [Section 2.3.3: ID-WSF
1264 Web Services Addressing EPR Profile](#)).
- 1265 The result set of EPRs generated in response to a particular <RequestedService> element can be further controlled
1266 using the following attributes:
- 1267 • reqID — an optional attribute identifying this <RequestedService> request. Typically only used when multiple
1268 <RequestedService> elements are included in a single Discovery Service <Query>.
- 1269 If present the value of this attribute will be placed into the reqRef attribute in any EPRs that result from this
1270 <RequestedService> element (see [Section 2.3.1.2](#) above).
- 1271 The value of reqID SHOULD be different for all <RequestedService> elements in a given <Query>.
- 1272 • resultsType — an optional attribute describing the results desired by the requestor. This value may be set to:
- 1273 • **best** — the Discovery Service SHOULD return the smallest set of EPRs while still meeting the minimum
1274 requirements of the request. This will typically be a single EPR.
- 1275 • **all** — the Discovery Service SHOULD return all of the matching entries for the given search criteria.

1276 This would typically be used when the client wants to choose which EPRs within the DS database it should
1277 use.

1278 This option should be used with caution as it can cause the DS to perform substantial work in order to mint all
1279 of the matching EPRs and the necessary security tokens for those EPRs.

1280 • **only-one** — a restricted version of **best** which further restricts the resulting output to **exactly one** EPR, even
1281 if the minimum requirements of the request would require multiple EPRs. A client would typically specify this
1282 option if it was going to ignore anything other than the first EPR returned.

1283 If `resultsType` is not specified the Discovery Service may make its own determination (under local policy) as
1284 to which set of results to return.

1285 Requestors **SHOULD** include at least one `<ServiceType>` or `<ProviderID>` element, and **MAY** include any number
1286 of both of them.

1287 Requesters **SHOULD** construct a `Query` to be as qualified as possible, as the Discovery Service instance may have to
1288 perform significant work for each item in the result set, especially if security tokens will be generated.

```

1289 <!-- Query Message Element & Type -->
1290
1291 <xs:element name="Query" type="QueryType" />
1292
1293 <xs:complexType name="QueryType">
1294   <xs:sequence>
1295     <xs:element name="RequestedService"
1296               type="RequestedServiceType"
1297               minOccurs="0"
1298               maxOccurs="unbounded" />
1299   </xs:sequence>
1300
1301   <xs:anyAttribute namespace="##other" processContents="lax" />
1302 </xs:complexType>
1303
1304 <xs:complexType name="RequestedServiceType">
1305   <xs:sequence>
1306     <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
1307
1308     <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
1309
1310     <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
1311
1312     <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
1313
1314     <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
1315
1316     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
1317
1318     <xs:any namespace="##other"
1319           processContents="lax"
1320           minOccurs="0"
1321           maxOccurs="unbounded" />
1322   </xs:sequence>
1323
1324   <xs:attribute name="reqID" type="xs:string" use="optional" />
1325   <xs:attribute name="resultsType" type="xs:string" use="optional" />
1326 </xs:complexType>
1327
1328
1329
1330
1331

```

1332 **Figure 17. Query Message — Schema Fragment**

```

1333
1334 <soap:Envelope
1335     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
1336
1337     <soap:Header>
1338         ...
1339     </soap:Header>
1340
1341     <soap:Body>
1342         <Query xmlns="&DS2Namespace;">
1343             <RequestedService>
1344                 <ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1345
1346                 <SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</SecurityMechID>
1347                 <SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</SecurityMechID>
1348                 <SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</SecurityMechID>
1349                 <Framework version="2.0" />
1350
1351             </RequestedService>
1352         </Query>
1353     </soap:Body>
1354 </soap:Envelope>
1355

```

1356 **Example 5. SOAP message containing a Query**

1357 3.3.3. QueryResponse

1358 A <QueryResponse> message conveys the results of the query as a set of ID-WSF EPRs, *i.e.*, profiled <wsa: |
1359 EndpointReference> elements (see [Section 2.3.3: ID-WSF Web Services Addressing EPR Profile](#)).

1360 As specified in [Section 2.3.3](#), security tokens, appropriate for subsequent invocation(s) of the service instances repre-
1361 sented by the returned ID-WSF EPRs, MAY be provided within the ID-WSF EPRs in the response.

1362 A status code is also included in the response.

```

1363
1364 <!-- QueryResponse Message Element & Type -->
1365
1366 <xs:element name="QueryResponse" type="QueryResponseType"/>
1367
1368 <xs:complexType name="QueryResponseType">
1369     <xs:sequence>
1370         <xs:element ref="lu:Status"/>
1371
1372         <xs:element ref="wsa:EndpointReference"
1373             minOccurs="0"
1374             maxOccurs="unbounded"/>
1375     </xs:sequence>
1376     <xs:anyAttribute namespace="##other" processContents="lax"/>
1377 </xs:complexType>
1378
1379

```

1380 **Figure 18. <QueryResponse> — Schema Fragment**

1381 An example SOAP message containing a <QueryResponse> message is illustrated in [Example 6](#). This example
1382 includes a security token embedded in the returned ID-WSF EPR. Parts of the security token have been omitted due
1383 to size.

```

1384
1385 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
1386
1387     <soap:Header>

```

```

1388     ...
1389 </soap:Header>
1390
1391 <soap:Body>
1392   <QueryResponse xmlns="&DS2Namespace;">
1393     <Status code="OK"/>
1394
1395     <wsa:EndpointReference
1396       notOnOrAfter="2005-08-15T23:18:56Z" >
1397       <wsa:Address>http://example.com/pip/bob</wsa:Address>
1398
1399       <wsa:Metadata>
1400         <ds:Abstract>
1401           Bob's personal profile
1402         </ds:Abstract>
1403
1404         <ds:ProviderID>http://example.com/</ds:ProviderID>
1405
1406         <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1407
1408         <ds:Framework Version="2.0" />
1409
1410         <ds:SecurityContext>
1411           <ds:SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</ds:SecurityMechID>
1412           <ds:SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</ds:SecurityMechID>
1413
1414           <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" >
1415             <saml2:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1416               ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
1417               Issuer="idp.example.com"
1418               IssueInstant="2003-09-09T16:58:33.173Z">
1419               <ds:Signature>...</ds:Signature>
1420             <saml2:Subject>
1421               <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1422                 http://serviceprovider.com/
1423               </saml2:NameID>
1424
1425               <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1426                 <saml2:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
1427                   <ds:KeyInfo>
1428                     <ds:KeyName>
1429                       CN=serviceprovider.com,
1430                       OU=Services R US,O=Service Nation,...
1431                     </ds:KeyName>
1432                   </ds:KeyInfo>
1433                 </saml2:SubjectConfirmationData>
1434               </saml2:SubjectConfirmation>
1435             </saml2:Subject>
1436
1437             <saml2:AuthnStatement AuthnInstant="2003-09-09T16:57:30.000Z"
1438               SessionIndex="..."
1439               SessionNotOnOrAfter="..."
1440             >
1441               <saml2:AuthnContext
1442                 ...
1443               </saml2:AuthnContext>
1444             </saml2:AuthnStatement>
1445
1446           </saml2:Assertion>
1447         </sec:Token>
1448       </ds:SecurityContext>
1449     </wsa:Metadata>
1450   </wsa:EndpointReference>
1451 </QueryResponse>
1452 </soap:Body>

```

1453 </soap:Envelope>
1454

1455 **Example 6. SOAP-bound <QueryResponse> Message with Embedded Security Token**

1456 3.3.4. DiscoveryQuery Processing Rules

1457 The discovery Service returns entries based on the requester's search criteria (interpreted as described above in [Section 3.3.2: <Query> Message](#)), the policies of the discovery resource, and the contents of the discovery resource.

1459 For each <RequestedService> element in a <Query> message, the matching rules MUST applied independently
1460 (as if the other <RequestedService> elements were not present (potentially returning equivalent EPRs in response
1461 to different <RequestedService> elements).

1462 When building the results for a given <RequestedService> element, the Discovery Service SHOULD return the
1463 data in as few EPRs as possible (within the constraints of the EPR) especially with respect to data originating from the
1464 same SvcMD.

1465 The Discovery Service SHOULD, when possible, provide the security tokens necessary for the security mechanism(s)
1466 identified in the ID-WSF EPRs in the response. If the Discovery Service is not able to generate the necessary security
1467 token, it should indicate so by including an empty <sec:Token> element with the ref attribute set to the value:

1468 urn:liberty:disco:tokenref:ObtainFromIDP

1469 The Discovery Service SHOULD mint new EPRs such that they carry the same identity context that was used to invoke
1470 the Discovery Service in the invocation context for the targeted WSP.

1471 When minting EPRs in response to a request, the Discovery Service:

- 1472 • MUST include in the EPR the matching (or all, if none were specified on the request) <Option> values contained
1473 within the <ServiceContext> of the <SvcMD> that matched the request criteria.
- 1474 • SHOULD NOT include any data from the <SvcMD> that was not specified in the request **and** is not necessary to
1475 create a useful EPR. For example, if the request specified a single <SecurityMechID> value, the resulting EPRs
1476 should not include other <SecurityMechID> elements, even if they are present in the <SvcMD> and otherwise
1477 could be specified in the same EPR.

1478 The Discovery Service MAY order <wsa:EndpointReference> elements as it sees fit. If the Discovery Service is
1479 rank ordering the entries, it MUST use descending rank order. This enables the requester to assume that if the results
1480 were ordered, the first result is the most relevant.

1481 The following rules specify the status code in the response:

- 1482 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code MUST
1483 be *Failed*.
- 1484 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NoResults* as a second-level
1485 status code.

1486 The service may not wish to reveal the reason for failure, in which case no second-level status code will appear.

1487 3.4. Operation: *MDAssociationAdd*

1488 The *MDAssociationAdd* operation is used by the WSP to add an association of the principal to the specified metadata.

1489 **3.4.1. wsa:Action values for MDAssociationAdd Messages**

1490 <SvcMDAssociationAdd> messages MUST include a <wsa:Action> SOAP header with the value of "urn:
1491 liberty:disco:2006-08:SvcMDAssociationAdd."

1492 <SvcMDAssociationAddResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1493 "urn:liberty:disco:2006-08:SvcMDAssociationAddResponse."

1494 **3.4.2. SvcMDAssociationAdd Message**

1495 The <SvcMDAssociationAdd> is called with one or more <SvcMDID> elements to add associations to these service
1496 metadata descriptions for the principal.

1497 A WSP SHOULD NOT associate the same <SvcMD> (or different SvcMD element that carry metadata for the "same"
1498 service) to a principal multiple times without first removing the previous entry.

1499 The values in the <SvcMDID> element(s) must have been obtained via one of the service metadata operations discussed
1500 later in this specification.

```
1501  
1502 <!-- SvcMDAssociationAdd operation -->  
1503  
1504 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType"/>  
1505  
1506 <xs:complexType name="SvcMDAssociationAddType">  
1507   <xs:sequence>  
1508     <xs:element ref="SvcMDID" maxOccurs="unbounded" />  
1509   </xs:sequence>  
1510   <xs:anyAttribute namespace="##other" processContents="lax"/>  
1511 </xs:complexType>  
1512
```

1513 **Figure 19. <SvcMDAssociationAdd> — Schema Fragment**

1514 An example message body containing a <SvcMDAssociationAdd> message follows. This request adds a new asso-
1515 ciation for the current principal (note that the identity of the principal is carried in the invocation context and not in the
1516 body of the message).

```
1517 <ds:SvcMDAssociationAdd>  
1518   <ds:SvcMDID>2323872</ds:SvcMDID>  
1520 </ds:SvcMDAssociationAdd>
```

1521 **Example 7. <SvcMDAssociationAdd> Message**

1522 **3.4.3. SvcMDAssociationAddResponse Message**

1523 This response to the <SvcMDAssociationAdd> request contains the following elements and attributes.

- 1524 • <lu:Status>: Contains status code; see processing rules.

```

1525
1526 <!-- Response for SvcMDAssociationAdd operation -->
1527
1528 <xs:element name="SvcMDAssociationAddResponse"
1529           type="SvcMDAssociationAddResponseType" />
1530
1531 <xs:complexType name="SvcMDAssociationAddResponseType">
1532   <xs:sequence>
1533     <xs:element ref="lu:Status" />
1534   </xs:sequence>
1535   <xs:anyAttribute namespace="##other" processContents="lax" />
1536 </xs:complexType>
1537

```

1538 **Figure 20. <SvcMDAssociationAddResponse> — Schema Fragment**

```

1539
1540 <ds:SvcMDAssociationAddResponse>
1541   <lu:Status code="OK" />
1542 </ds:SvcMDAssociationAddResponse>

```

1543 **Example 8. <SvcMDAssociationAddResponse> Message**

1544 3.4.4. MDAssociation Add Processing Rules

- 1545 • Once the association is added by the WSP, the Discovery Service **MUST** consider this metadata (subject to local
1546 policy) when responding to subsequent DiscoveryQuery operations and should the associated metadata meet the
1547 requirements of the query, mint the necessary ID-WSF EPRs based upon the requirements of the WSC and the
1548 WSP.
- 1549 • The Discovery Service **SHOULD** reject attempts to associate a <SvcMDID> that has already been associated with
1550 the principal by this WSP. In such cases the Discovery service **MAY** set the second level status code in the response
1551 to *Duplicate*.
- 1552 • The Discovery Service **MAY** similarly reject attempts to associate a <SvcMDID> that references the same service
1553 type and WSP that is in one of the already associated service metadata descriptions. In such cases the Discovery
1554 service **MAY** set the second level status code in the response to *LogicalDuplicate*.
- 1555 • The Discovery Service **MUST** reject attempts to associate a <SvcMDID> that does not exist or is not owned by the
1556 WSP invoking the call. In such cases the Discovery service **MAY** set the second level status code in the response
1557 to *NotFound*.
- 1558 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code **MUST**
1559 be *Failed*.
- 1560 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden*, *Duplicate*, *LogicalDuplicate*, or
1561 *NotFound* as a second-level status code. The Discovery Service instance may not wish to reveal the reason for
1562 failure, in which case no second-level status code will appear.
- 1563 • A Discovery Service **MAY** provide some programmatic or browser based interface which allows the principal to
1564 manage the service associations that have been added to their resource at the Discovery Service. A principal may
1565 be able to use such interfaces to change or even remove service associations made by the WSP without the WSP's
1566 permission (it is the principal's resource) and perhaps, even without notification to the WSP.
- 1567 Such interfaces are out-of-scope for this specification, but are mentioned here to remind the WSP that they may
1568 exist.

1569 **3.5. Operation: *MDAssociationQuery***

1570 The *MDAssociationQuery* operation is used by the WSP to query the Discovery Service for any previously added
1571 associations related to the principal.

1572 **3.5.1. wsa: Action values for *MDAssociationQuery* Messages**

1573 <SvcMDAssociationQuery> messages MUST include a <wsa: Action> SOAP header with the value of "urn:
1574 liberty:disco:2006-08:SvcMDAssociationQuery."

1575 <SvcMDAssociationQueryResponse> messages MUST include a <wsa: Action> SOAP header with the value of
1576 "urn:liberty:disco:2006-08:SvcMDAssociationQueryResponse."

1577 **3.5.2. SvcMDAssociationQuery Message**

1578 The <SvcMDAssociationQuery> is called with zero or more <SvcMDID> elements to query associations to these
1579 service metadata descriptions. If no <SvcMDID> elements are specified, ALL associations between the WSP's service
1580 metadata and the principal are returned.

```
1581 <!-- SvcMDAssociationQuery operation -->
1582
1583 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType"/>
1584
1585 <xs:complexType name="SvcMDAssociationQueryType">
1586   <xs:sequence>
1587     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1588   </xs:sequence>
1589   <xs:anyAttribute namespace="##other" processContents="lax"/>
1590 </xs:complexType>
```

1592 **Figure 21. <SvcMDAssociationQuery> — Schema Fragment**

1593 An example message body containing a <SvcMDAssociationQuery> message follows. This request asks for all
1594 associations.

```
1595 <ds:SvcMDAssociationQuery />
```

1597 **Example 9. <SvcMDAssociationQuery> Message**

1598 **3.5.3. SvcMDAssociationQueryResponse Message**

1599 This response to the <SvcMDAssociationQuery> request contains the following elements and attributes.

- 1600 • <lu: Status>: Contains status code; see processing rules.
- 1601 • <SvcMDID>: the associated service metadata ID(s). If <SvcMDID>s were specified on the
1602 <SvcMDAssociationQuery> the response will be limited to at most those IDs (if they have been associated with
1603 the principal).

```

1604
1605 <!-- Response for SvcMDAssociationQuery operation -->
1606
1607 <xs:element name="SvcMDAssociationQueryResponse"
1608           type="SvcMDAssociationQueryResponseType" />
1609
1610 <xs:complexType name="SvcMDAssociationQueryResponseType">
1611   <xs:sequence>
1612     <xs:element ref="lu:Status" />
1613     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1614   </xs:sequence>
1615   <xs:anyAttribute namespace="##other" processContents="lax" />
1616 </xs:complexType>
1617

```

Figure 22. <SvcMDAssociationQueryResponse> — Schema Fragment

```

1619
1620 <ds: SvcMDAssociationQueryResponse>
1621   <lu: Status code="OK" />
1622   <ds: SvcMDID>2323872</ds: SvcMDID>
1623 </ds: SvcMDAssociationQueryResponse>

```

Example 10. <SvcMDAssociationQueryResponse> Message

3.5.4. MDAssociation Query Processing Rules

- The Discovery Service MUST limit the operation to only those associations added by the WSP to the current principal's resource (a WSP MUST NOT be able to query associations added at the same Discovery Service by other WSPs or associations added to a different principal). There MUST NOT be any indication on the response as to whether or not other such elements exist.
- If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code MUST be *Failed*.
- If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NotFound* as a second-level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code will appear.

3.6. Operation: *MDAssociationDelete*

The *MDAssociationDelete* operation is used by the WSP to delete a previously added association of the principal to the specified metadata.

3.6.1. wsa: Action values for MDAssociationDelete Messages

<SvcMDAssociationDelete> messages MUST include a <wsa: Action> SOAP header with the value of "urn:liberty:disco:2006-08: SvcMDAssociationDelete."

<SvcMDAssociationDeleteResponse> messages MUST include a <wsa: Action> SOAP header with the value of "urn:liberty:disco:2006-08: SvcMDAssociationDeleteResponse."

3.6.2. SvcMDAssociationDelete Message

The <SvcMDAssociationDelete> is called with one or more <SvcMDID> elements to delete associations to these service metadata descriptions.

Note that the service metadata description is not impacted by this call. Only the principal's association with the metadata is impacted.

```

1648
1649 <!-- SvcMDAssociationDelete operation -->
1650
1651 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType"/>
1652
1653 <xs:complexType name="SvcMDAssociationDeleteType">
1654   <xs:sequence>
1655     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
1656   </xs:sequence>
1657   <xs:anyAttribute namespace="##other" processContents="lax" />
1658 </xs:complexType>

```

1659 **Figure 23. <SvcMDAssociationDelete> — Schema Fragment**

1660 An example message body containing a <SvcMDAssociationDelete> message follows. This request deletes a single
1661 association for the current principal (note that the identity of the principal is carried in the invocation context and not
1662 in the body of the message).

```

1663
1664 <ds: SvcMDAssociationDelete>
1665   <ds: SvcMDID>2323872</ds: SvcMDID>
1666 </ds: SvcMDAssociationDelete>

```

1667 **Example 11. <SvcMDAssociationDelete> Message**

1668 3.6.3. SvcMDAssociationDeleteResponse Message

1669 This response to the <SvcMDAssociationDelete> request contains the following elements and attributes.

- 1670 • <lu:Status>: Contains status code; see processing rules.

```

1671
1672 <!-- Response for SvcMDAssociationDelete operation -->
1673
1674 <xs:element name="SvcMDAssociationDeleteResponse"
1675   type="SvcMDAssociationDeleteResponseType"/>
1676
1677 <xs:complexType name="SvcMDAssociationDeleteResponseType">
1678   <xs:sequence>
1679     <xs:element ref="lu:Status" />
1680   </xs:sequence>
1681   <xs:anyAttribute namespace="##other" processContents="lax" />
1682 </xs:complexType>
1683

```

1684 **Figure 24. <SvcMDAssociationDeleteResponse> — Schema Fragment**

```

1685
1686 <ds: SvcMDAssociationDeleteResponse>
1687   <lu:Status code="OK" />
1688 </ds: SvcMDAssociationDeleteResponse>

```

1689 **Example 12. <SvcMDAssociationDeleteResponse> Message**

1690 3.6.4. MDAssociation Delete Processing Rules

- 1691 • Once deleted, the association MUST NOT be subsequently used by the DS to mint ID-WSF EPRs in response to
1692 queries relative to this principal. However, WSPs should be prepared to receive requests from WSCs from clients
1693 who previously obtained ID-WSF EPRs minted from the associaton which haven't expired.
- 1694 • The Discovery Service MUST limit the operation to only those associations added by the WSP to the current
1695 principal's resource (a WSP MUST NOT be able to delete associations added at the same Discovery Service by

1696 other WSPs or associations added to a different principal). There MUST NOT be any indication on the response
 1697 as to whether or not other such elements exist.

1698 • The Discovery Service MUST treat attempts to delete non-existent associations as a successful no-op. This applies
 1699 whether or not there are other existing associations being deleted in the same request (so a request to delete a single
 1700 association that doesn't exist will succeed, even though the Discovery Service does not have to actually delete the
 1701 record).

1702 • This operation MUST be atomic and if successful, all portions of the request MUST have succeeded. If any portion
 1703 of the request fails, the entire request must fail.

1704 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code MUST
 1705 be *Failed*.

1706 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* as a second-level status code. The
 1707 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code
 1708 will appear.

1709 **3.7. Operation: *MetadataRegister***

1710 The *MetadataRegister* operation is used to register a new service metadata description with the Discovery Service.

1711 **3.7.1. *wsa:Action* values for *MetadataRegister* Messages**

1712 <SvcMDRegister> messages MUST include a <wsa:Action> SOAP header with the value of "urn:liberty:
 1713 disco:2006-08:SvcMDRegister."

1714 <SvcMDRegisterResponse> messages MUST include a <wsa:Action> SOAP header with the value of "urn:
 1715 liberty:disco:2006-08:SvcMDRegisterResponse."

1716 **3.7.2. *SvcMDRegister* Message**

1717 The <SvcMDRegister> is called with one or more service metadata descriptions to be registered at the Discovery
 1718 Service on behalf of the WSP.

```

1719
1720 <!-- Register operation for Service Metadata -->
1721
1722 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
1723
1724 <xs:complexType name="SvcMDRegisterType">
1725   <xs:sequence>
1726     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1727   </xs:sequence>
1728   <xs:anyAttribute namespace="##other" processContents="lax" />
1729 </xs:complexType>
1730
    
```

1731 **Figure 25. <SvcMDRegister> — Schema Fragment**

1732 An example message body containing a <SvcMDRegister> message follows. This request registers a new service
 1733 metadata description. Note that the WSP has not set the *svcMDID* attribute on the <SvcMD> element -- this will be
 1734 assigned by the DS and returned in the response to the WSP.

```

1735
1736     <ds: SvcMDRegister>
1737         <ds: SvcMD>
1738             <ds: Abstract>Profile Service</ds: abstract>
1739             <ds: ProviderID>http://profile.com</ds: ProviderID>
1740             <ds: ServiceContext>
1741                 <ds: ServiceType>urn:liberty:pp:2003-08</ds: ServiceType>
1742                 <ds: EndpointContext>
1743                     <ds: Address>https://profile.com/</ds: Address>
1744                     <sb: Framework version="2.0" />
1745                     <ds: SecurityMechID>
1746                         urn:liberty:security:2003-08:TLS:Bearer
1747                     </ds: SecurityMechID>
1748                 </ds: EndpointContext>
1749             </ds: ServiceContext>
1750         </ds: SvcMD>
1751     </ds: SvcMDRegister>

```

1752 **Example 13. <SvcMDRegister> Message**

1753 3.7.3. SvcMDRegisterResponse Message

1754 This response to the <SvcMDRegister> request contains the following elements and attributes.

- 1755 • <lu: Status>: Contains status code; see processing rules.
- 1756 • One or more <SvcMDID> if the call was successful (status code is OK). One SvcMDID is returned for each service
1757 metadata element registered.
- 1758 • <Keys>: Contains the key descriptors for the keys used by the Discovery Service to sign security tokens (see
1759 [Section 3.12](#) for a description of when and why this may be necessary).

```

1760
1761 <!-- Response for SvcMDRegister operation -->
1762
1763 <xs: element name="SvcMDRegisterResponse"
1764             type="SvcMDRegisterResponseType" />
1765
1766 <xs: complexType name="SvcMDRegisterResponseType">
1767     <xs: sequence>
1768
1769         <xs: element ref="lu: Status" />
1770         <xs: element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1771         <xs: element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
1772
1773     </xs: sequence>
1774     <xs: anyAttribute namespace="##other" processContents="lax" />
1775 </xs: complexType>
1776
1777

```

1778 **Figure 26. <SvcMDRegisterResponse> — Schema Fragment**

```

1779
1780     <ds: SvcMDRegisterResp>
1781         <lu: Status code="OK" /><code="OK" />
1782         <ds: SvcMDID>2323872</ds: SvcMDID>
1783     </ds: SvcMDRegister>

```

1784 **Example 14. <SvcMDRegisterResponse> Message**

1785 3.7.4. Metadata Register Processing Rules

1786 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
1787 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

1788 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use the
1789 Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the operation
1790 if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

1791 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The
1792 Discovery Service **MUST** enforce this atomicity.

1793 • For each <SvcMD> element, the Discovery Service instance **MAY** store the metadata provided such that it can be
1794 used (subject to policy) to mint ID-WSF EPRs in response to future *DiscoveryQuery* operations should that service
1795 metadata be associated with a principal's resource at the Discovery Service.

1796 If the Discovery Service instance does not store the metadata, it **MUST** return a *Failed* status code for the operation,
1797 and therefore not register any of the other entries provided.

1798 If the Discovery Service does store the metadata, it **MUST** assign a permanent identifier for the metadata usable
1799 by the WSP to subsequently reference the metadata. This identifier **MUST** be unique across all metadata objects
1800 stored by a WSP and **MAY** be unique across all metadata objects stored by all WSPs at that Discovery Service.
1801 This identifier is provided to the WSP in the response and can be subsequently used by the WSP to associate this
1802 metadata with a principal or to manage the metadata using one of the other metadata operations.

1803 • A WSP **MAY** register multiple service metadata descriptions that for all intents and purposes, appear to be fully
1804 equal. The Discovery Service **MUST NOT** generate an error solely because it thinks the descriptions are equal.
1805 The Discovery Service **MUST** treat these records as independent registrations and assign the associated unique
1806 SvcMDID values.

1807 • The Discovery Service **MAY** have some policy driven limit on the number of service metadata descriptions that it
1808 will allow a WSP to register. If a WSP attempts to register a new service metadata description that would exceed
1809 such a limit, the DS **SHOULD** include a secondary-level status code of *LimitExceeded*.

1810 A WSP should exercise care to only register new service metadata descriptions when an existing, registered, de-
1811 scription that meets the WSP's needs is not available.

1812 • The Discovery Service **SHOULD** validate that a given SvcMD only contains entries for a single logical service
1813 (*i.e.*, allow for different versions, but **not** allow differences in the basic service type). For example, a single SvcMD
1814 **SHOULD** not contain data for both a contact book service and a calendar service.

1815 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.

1816 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the registration request and **MAY**
1817 include a secondary-level status code of *Invalid*.

1818 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code **MUST**
1819 be *Failed*.

1820 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden*, *Invalid*, or *OverLimit* as a second-
1821 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
1822 second-level status code will appear.

1823 **3.8. Operation: *MetadataQuery***

1824 The *MetadataQuery* operation is used to query the Discovery Service for existing, registered, service metadata de-
1825 scriptions.

1826 **3.8.1. wsa:Action values for MetadataQuery Messages**

1827 <SvcMDQuery> messages MUST set the value of the <wsa:Action> header to "urn:liberty:disco:2006-08:
1828 SvcMDQuery."

1829 <SvcMDQueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of "urn:
1830 liberty:disco:2006-08:SvcMDQueryResponse."

1831 **3.8.2. SvcMDQuery Message**

1832 The <SvcMDQuery> is called with zero or more <SvcMDID> elements to retrieve the specified list of service metadata
1833 descriptions. If no <SvcMDID>s are specified, ALL of the metadata stored at the Discovery service by the invoking
1834 WSP will be returned.

```
1835
1836 <!-- Query operation on Service Metadata -->
1837
1838 <xs:element name="SvcMDQuery" type="SvcMDQueryType" />
1839
1840 <xs:complexType name="SvcMDQueryType">
1841   <xs:sequence>
1842     <xs:element ref="SvcMDID"
1843               minOccurs="0"
1844               maxOccurs="unbounded" />
1845   </xs:sequence>
1846   <xs:anyAttribute namespace="##other" processContents="lax" />
1847 </xs:complexType>
1848
```

1849 **Figure 27. <SvcMDQuery> — Schema Fragment**

1850 An example message body containing a <SvcMDQuery> message follows. This request queries for a specific service
1851 metadata description by providing the ID of the desired metadata in the <SvcMDID> element.

```
1852
1853 <ds: SvcMDQuery>
1854   <ds: SvcMDID>2323872</ds: SvcMDID>
1855 </ds: SvcMDQuery>
```

1856 **Example 15. <SvcMDQuery> Message**

1857 **3.8.3. SvcMDQueryResponse Message**

1858 This response to the <SvcMDQuery> request contains the following elements and attributes.

- 1859 • <lu:Status>: Contains status code; see processing rules.
- 1860 • One or more <SvcMD> elements if the call was successful (status code is OK).

```

1861
1862 <!-- Response for Query operation on Service Metadata -->
1863
1864 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType"/>
1865
1866 <xs:complexType name="SvcMDQueryResponseType">
1867   <xs:sequence>
1868     <xs:element ref="lu:Status" />
1869     <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
1870   </xs:sequence>
1871   <xs:anyAttribute namespace="##other" processContents="lax" />
1872 </xs:complexType>
1873
1874

```

Figure 28. <SvcMDQueryResponse> — Schema Fragment

```

1876
1877 <ds:SvcMDQueryResponse>
1878   <lu:Status code="OK" />
1879   <ds:SvcMD svcMDID="2323872" >
1880     <ds:Abstract>Profile Service</ds:Abstract>
1881     <ds:ProviderID>http://profile.com</ds:ProviderID>
1882     <ds:ServiceContext>
1883       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1884       <ds:EndpointContext>
1885         <ds:Address>https://profile.com/</ds:Address>
1886         <sb:Framework version="2.0" />
1887         <ds:SecurityMechID>
1888           urn:liberty:security:2003-08:TLS:Bearer
1889         </ds:SecurityMechID>
1890       </ds:EndpointContext>
1891     </ds:ServiceContext>
1892   </ds:SvcMD>
1893 </ds:SvcMDQueryResponse>

```

Example 16. <SvcMDQueryResponse> Message

3.8.4. Metadata Query Processing Rules

- 1896 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
1897 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

- 1898 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use the
1899 Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the operation
1900 if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

- 1901 • The Discovery Service **MUST** limit the results to only those metadata elements stored by the WSP (a WSP **MUST**
1902 **NOT** be able to retrieve metadata elements stored at the same Discovery Service by other WSPs). There **MUST**
1903 **NOT** be any indication on the response as to whether or not other such elements exist.

- 1904 • The Discovery Service **SHOULD** treat a request that matches a subset of the `svcMDID` values specified in the
1905 request as a successful request returning the entries that were found and nothing for the missing entries. The WSC
1906 will be able to distinguish which entries were found by examining the `svcMDID` attribute on the `<svcMD>` element
1907 (s) in the response.

- 1908 • If request processing succeeded **AND** results are returned, the top-level status code **MUST** be *OK*. Otherwise, the
1909 top-level status code **MUST** be *Failed*.

- 1910 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* or *NoResults* as a second-level
1911 status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no second-
1912 level status code will appear.

1913 3.9. Operation: *MetadataReplace*

1914 The *MetadataReplace* operation is used by a WSP to replace previously stored metadata in the Discovery Service. This
1915 is how the WSP updates their metadata without having to reassociate with a principal.

1916 3.9.1. *wsa:Action* values for *MetadataReplace* Messages

1917 <SvcMDReplace> messages MUST include a <wsa:Action> SOAP header with the value of "urn:liberty:
1918 disco:2006-08:SvcMDReplace."

1919 <SvcMDReplaceResponse> messages MUST include a <wsa:Action> SOAP header with the value of "urn:
1920 liberty:disco:2006-08:SvcMDReplaceResponse."

1921 3.9.2. *SvcMDReplace* Message

1922 The <SvcMDReplace> is called with one or more replacement <SvcMD> elements each of which must include the
1923 svcMDID attribute set to the ID of the respective metadata element they are to replace.

```
1924 <!-- Replace operation on Service Metadata -->
1925
1926 <xs:element name="SvcMDReplace" type="SvcMDReplaceType"/>
1927
1928 <xs:complexType name="SvcMDReplaceType">
1929   <xs:sequence>
1930     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1931   </xs:sequence>
1932   <xs:anyAttribute namespace="##other" processContents="lax"/>
1933 </xs:complexType>
1934
1935
```

1936 **Figure 29. <SvcMDReplace> — Schema Fragment**

1937 An example message body containing a <SvcMDReplace> message follows. This request replaces an existing metadata
1938 element to update the endpoint for the service.

```
1939 <ds:SvcMDReplace>
1940   <ds:SvcMD svcMDID="2323872" >
1941     <ds:Abstract>Profile Service</ds:abstract>
1942     <ds:ProviderID>http://profile.com</ds:ProviderID>
1943     <ds:ServiceContext>
1944       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1945       <ds:EndpointContext>
1946         <ds:Address>https://newaddr.com/</ds:Address>
1947         <sb:Framework version="2.0"/>version="2.0"/>
1948         <ds:SecurityMechID>
1949           urn:liberty:security:2003-08:TLS:Bearer
1950         </ds:SecurityMechID>
1951       </ds:EndpointContext>
1952     </ds:ServiceContext>
1953   </ds:SvcMD>
1954 </ds:SvcMDReplace>
```

1956 **Example 17. <SvcMDReplace> Message**

1957 3.9.3. *SvcMDReplaceResponse* Message

1958 This response to the <SvcMDReplace> request contains the following elements and attributes.

- 1959 • <lu:Status>: Contains status code; see processing rules.

```

1960
1961 <!-- Response for SvcMDReplace operation -->
1962
1963 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType" />
1964
1965 <xs:complexType name="SvcMDReplaceResponseType">
1966   <xs:sequence>
1967     <xs:element ref="lu:Status" />
1968   </xs:sequence>
1969   <xs:anyAttribute namespace="##other" processContents="lax" />
1970 </xs:complexType>
1971
1972

```

Figure 30. <SvcMDReplaceResponse> — Schema Fragment

```

1973
1974
1975 <ds: SvcMDReplaceResponse>
1976   <lu: Status code="OK" />
1977 </ds: SvcMDReplaceResponse>

```

Example 18. <SvcMDReplaceResponse> Message

3.9.4. Metadata Replace Processing Rules

- 1980 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
- 1981 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

- 1982 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use the
- 1983 Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the operation
- 1984 if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

- 1985 • The Discovery Service **MUST** limit the operation to only those metadata elements stored by the WSP (a WSP
- 1986 **MUST NOT** be able to replace metadata elements stored at the same Discovery Service by other WSPs). There
- 1987 **MUST NOT** be any indication on the response as to whether or not other such elements exist.

- 1988 • The Discovery Service **SHOULD** validate that the replacement SvcMD contains the same logical service as the
- 1989 original SvcMD. By "logical service" we mean to allow for different versions, but **not** allow differences in the basic
- 1990 service type. For example, a calendar service SvcMD **SHOULD** not be allowed to replace a contact book service
- 1991 SvcMD.

- 1992 The Discovery Service **SHOULD** also validate that the replacement SvcMD only contains entries for a single logical
- 1993 service (as described above). For example, a single SvcMD **SHOULD** not contain data for both a contact book
- 1994 service and a calendar service.

- 1995 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.

- 1996 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the replacement request and **MAY**
- 1997 include a secondary-level status code of *Invalid*.

- 1998 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The
- 1999 Discovery Service **MUST** enforce this atomicity.

- 2000 • Once replaced, the previous service metadata element **MUST NOT** be subsequently used by the DS to mint ID-
- 2001 WSF EPRs. However, WSPs should be prepared to receive requests from WSCs from clients who previously
- 2002 obtained ID-WSF EPRs minted from the prior metadata which haven't expired.

- 2003 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code **MUST**
- 2004 be *Failed*.

- 2005 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Invalid*, or *NotFound* as a second-
 2006 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
 2007 second-level status code will appear.

2008 **3.10. Operation: *MetadataDelete***

2009 The *MetadataDelete* operation is used by the WSP to delete previously registered metadata elements in the Discovery
 2010 Service.

2011 **3.10.1. wsa: Action values for *MetadataDelete* Messages**

2012 <SvcMDelete> messages MUST include a <wsa:Action> SOAP header with the value of "urn:liberty:
 2013 disco:2006-08:SvcMDelete."

2014 <SvcMDeleteResponse> messages MUST include a <wsa:Action> SOAP header with the value of "urn:
 2015 liberty:disco:2006-08:SvcMDeleteResponse."

2016 **3.10.2. SvcMDelete Message**

2017 The <SvcMDelete> is called with one or more <SvcMDID> elements to delete the specified list of service metadata
 2018 descriptions.

```

2019
2020 <!-- Delete operation on Service Metadata -->
2021
2022 <xs:element name="SvcMDelete" type="SvcMDeleteType"/>
2023
2024 <xs:complexType name="SvcMDeleteType">
2025   <xs:sequence>
2026     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
2027   </xs:sequence>
2028   <xs:anyAttribute namespace="##other" processContents="lax"/>
2029 </xs:complexType>
2030
    
```

2031 **Figure 31. <SvcMDelete> — Schema Fragment**

2032 An example message body containing a <SvcMDelete> message follows. This request deletes a single service met-
 2033 adata description.

```

2034
2035 <ds: SvcMDelete>
2036   <ds: SvcMDID>2323872</ds: SvcMDID>
2037 </ds: SvcMDelete>
    
```

2038 **Example 19. <SvcMDelete> Message**

2039 **3.10.3. SvcMDeleteResponse Message**

2040 This response to the <SvcMDelete> request contains the following elements and attributes.

- 2041 • <lu:Status>: Contains status code; see processing rules.

```

2042
2043 <!-- Response for delete operation on Service Metadata -->
2044
2045 <xs:element name="SvcMDDeleteResponse" type="SvcMDDeleteResponseType" />
2046
2047 <xs:complexType name="SvcMDDeleteResponseType">
2048   <xs:sequence>
2049     <xs:element ref="lu:Status" />
2050   </xs:sequence>
2051   <xs:anyAttribute namespace="##other" processContents="lax" />
2052 </xs:complexType>
2053
2054

```

2055 **Figure 32. <SvcMDDeleteResponse> — Schema Fragment**

```

2056
2057   <ds: SvcMDDeleteResponse>
2058     <lu:Status code="OK" />
2059 </ds: SvcMDDeleteResponse>

```

2060 **Example 20. <SvcMDDeleteResponse> Message**

2061 3.10.4. Metadata Delete Processing Rules

2062 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
 2063 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

2064 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use the
 2065 Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the operation
 2066 if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

2067 • If the service metadata being deleted is still associated with one or more principals, the Discovery Service **MUST**
 2068 automatically remove such associations (*i.e.*, the delete of metadata cascades to delete the associations).

2069 • Once deleted, the service metadata element **MUST NOT** be subsequently used by the DS to mint ID-WSF EPRs.
 2070 However, WSPs should be prepared to receive requests from WSCs from clients who previously obtained ID-WSF
 2071 EPRs minted from the metadata which haven't expired.

2072 • The Discovery Service **MUST** limit the operation to only those metadata elements stored by the WSP (a WSP
 2073 **MUST NOT** be able to delete metadata elements stored at the same Discovery Service by other WSPs). There
 2074 **MUST NOT** be any indication on the response as to whether or not other such elements exist.

2075 • The Discovery Service **MUST** treat attempts to delete non-existent metadata elements as a successful no-op. This
 2076 applies whether or not there are other existing metadata elements being deleted in the same request (so a request
 2077 to delete a single metadata element that doesn't exist will succeed, even though the Discovery Service does not
 2078 have to actually delete the record).

2079 • This operation **MUST** be atomic and if successful, all portions of the request **MUST** have succeeded. If any portion
 2080 of the request fails, the entire request must fail.

2081 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code **MUST**
 2082 be *Failed*.

2083 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* as a second-level status code. The
 2084 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code
 2085 will appear.

2086 3.11. option Value for Response Authentication

2087 The ID-WSF EPR <SecurityContext> element provides a way for services to indicate to clients what mechanisms
2088 are necessary for the client to authenticate itself to the service via the <SecurityMechID> element. The
2089 <SecurityMechID> values defined by [LibertySecMech] also indicate whether the service uses peer entity authen-
2090 tication (for example, server-side SSL/TLS). However, a web service client may need to know whether the service will
2091 use message authentication (that is, whether the service will sign the response message) and may not be willing to use
2092 a service which does not sign its responses.

2093 To avoid situations where a client requests data and then discovers it does not trust it because it is not signed, an
2094 <Option> value is defined:

2095 *urn:liberty:disco:2006-08:options:security-response-x509*

2096 If a service instance always authenticates its response messages according to the "X.509 v3 Certificate Message Au-
2097 thentication" mechanism in [LibertySecMech], registrations of ID-WSF EPRs describing the service instance
2098 SHOULD include this option value. Otherwise, its registered ID-WSF EPRs MUST NOT include this option value.
2099 Clients MAY include this option value in <Query> messages in order to locate only services which always authenticate
2100 their response messages. A service MAY authenticate its response messages even if this option value was not included
2101 in its description at the Discovery Service instance.

2102 In case the service also supports a previous version of the security mechanism specification [LibertySecMech11], it
2103 should be able to register two different endpoints at the Discovery Service, each of them with different Options values
2104 —one according to [LibertySecMech], the other one according to [LibertySecMech11]. This information will aid the
2105 client in determining which version of the WSS-SMS specification ([wss-sms-draft] and/or [wss-sms]) is supported
2106 by the service, and the service will act accordingly, depending on the ID-WSF EPR used by the client. Note that this
2107 behavior only applies to the case when the client's request does not use message authentication mechanisms.

2108 Otherwise, it should be possible for the service to determine the version of the WSS-SMS specification supported by
2109 the client by simply analyzing the <wsse:Security> header present in the request.

2110 In general, it is recommended that services do not sign their responses unless they positively know that clients are able
2111 to perform message authentication and are aware of the version of the WSS-SMS spec used by that client.

2112 3.12. Including Keys in the **SvcMDRegisterResponse** Message

2113 The Discovery Service instance may need to generate signed security tokens in <QueryResponse> messages for the
2114 ID-WSF EPRs in question (which are later included in a message to a WSP). The WSP which receives the signed
2115 security tokens from a client needs to be able to verify the Discovery service instance's signature on the security tokens.
2116 Typically the metadata (see [SAMLMeta2]) for the Discovery service instance is sufficient for such information. In
2117 certain situations, such as when the Discovery service instance is hosted on a LUAD (see [LibertyClientProfiles]), it
2118 may not be feasible to assign the LUAD a ProviderID with which to obtain metadata. However, the key material still
2119 needs to be made available to service instances which register ID-WSF EPRs with the Discovery Service which include
2120 security mechanisms requiring such tokens.

2121 The Discovery Service instance may include a <Keys> element in the **<SvcMDRegisterResponse>** in order to pro-
2122 vide such keys.

2123 The Discovery Service instance SHOULD ONLY include the <Keys> element in **<SvcMDRegisterResponse>**
2124 messages if it has no <ProviderID> and the **<SvcMDRegister>** message included ~~an~~ **Service Metadata that relies**
2125 **upon signed security tokens for one or more of its signed security mechanisms.**

```

2126
2127 <!-- Keys Element - For use in ModifyResponse -->
2128
2129 <xs:element name="Keys" type="KeysType"/>
2130
2131 <xs:complexType name="KeysType">
2132   <xs:sequence>
2133     <xs:element ref="md:KeyDescriptor"
2134       minOccurs="1"
2135       maxOccurs="unbounded"/>
2136   </xs:sequence>
2137 </xs:complexType>
2138
2139

```

Figure 33. <Keys> — Schema Fragment

2141 The <Keys> element appears as a child of the <SvcMDRegisterResponse> element. It contains one or more
 2142 <KeyDescriptor> elements.

2143 3.13. Discovery Service Example Messages (NON-Normative)

2144 This section walks through a series of messages to show examples of inputs and outputs. The information in this section
 2145 is **NOT** normative with respect to the specification (in cases where the other normative section(s) of the specification
 2146 conflict with this section, the normative section(s) will prevail).

2147 Notes about this sequence:

- 2148 • It is an actual network capture of a real session between two independent liberty implementations.
- 2149 • The messages are from a **test sequence** which exercises the features of the Discovery Service. We do **not** expect
 2150 that any real world situation would result in this sequence of messages (or anything similar to this sequence of
 2151 messages).
- 2152 • All of these messages were invoked using the same SAML Assertion to establish the invocation context with the
 2153 principal as the subject and the WSP in the subject confirmation. Note that some of the processing rules for the
 2154 Discovery Service interfaces (those that manage the service metadata) require that such an invocation context be
 2155 interpreted as a WSP invoker invocation context.
- 2156 • The initial state is that the principal has a single service associated with their identity (an instance of the ID-WSF
 2157 People Service).
- 2158 • The messages **are** part of a sequence that was invoked in this order and one request (such as a new registration)
 2159 can and usually does impact the results of subsequent requests.
- 2160 • The first request and response messages are shown as a full SOAP-bound ID-* message (with all of the SOAP
 2161 headers). The remaining request and response messages are shown with just the ID-* message component (**i.e.,** |
 2162 only the contents within the <S:Body> element are shown).
- 2163 • The messages have been formatted for easier reading (pretty-printing) and data has been ellided (such as the contents
 2164 of SAML assertions) for brevity.
- 2165 • Many of the service types used in these example messages do **not** exist as real ID-* services and are only used here
 2166 as testing input to exercise the Discovery Service.

2167 3.13.1. Query People Service

2168 Query for the People Service

```

2169 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
2170     xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd"
2171     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2172     xmlns:sb2="urn:liberty:sb:2006-08"
2173     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2174     xmlns:wsa="http://www.w3.org/2005/08/addressing">
2175   <S:Header>
2176     <wsa:MessageID S:mustUnderstand="true"
2177       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="msgHdrID">
2178       uuid:asdqwer-238asf-44353608-000b8c14
2179     </wsa:MessageID>
2180     <wsa:To S:mustUnderstand="true"
2181       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaToID">
2182       https://s-ds.liberty-iop.org:8681/DISCO-S
2183     </wsa:To>
2184     <wsa:Action S:mustUnderstand="true"
2185       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaActionID">
2186       urn:liberty:disco:2006-08:Query
2187     </wsa:Action>
2188     <wsse:Security S:mustUnderstand="true"
2189       S:actor="http://schemas.xmlsoap.org/soap/actor/next">
2190       <sa:Assertion
2191         xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2192         ID="CRED6q6HqDRRCAPsq3L8d_sh"
2193         IssueInstant="2006-04-06T15:39:12Z"
2194         Version="2.0">
2195         ... assertion data was here ...
2196       </sa:Assertion>
2197       <wsu:Timestamp wsu:Id="WsuTimestampID">
2198         <wsu:Created>2006-04-06T15:38:48Z</wsu:Created>
2199       </wsu:Timestamp>
2200     </wsse:Security>
2201   </S:Header>
2202   <S:Body wsu:Id="msgBodyID">
2203     <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2204       <disco:RequestedService>
2205         <disco:ServiceType>urn:liberty:ps:2006-08</disco:ServiceType>
2206         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2207         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2208       </disco:RequestedService>
2209     </disco:Query>
2210   </S:Body>
2211 </S:Envelope>

```

2212 Things to note about this query:

- 2213 • it is a query of a service type (in this case, a particular version of the ID-WSF People Service)
- 2214 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for
- 2215 communicating with the specified service

2216 The response from the Discovery Service:

```

2217 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
2218     xmlns:lib="urn:liberty:iff:2003-08">
2219   <soap:Header>
2220     <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing" id="MID">
2221       uuid:IKvqaaaE0dZ5HhglIQZl
2222     </wsa:MessageID>
2223     <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
2224       uuid:asdqwer-238asf-44353608-000b8c14
2225     </wsa:RelatesTo>
2226     <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
2227       urn:liberty:disco:2006-08:QueryResponse
2228     </wsa:Action>
2229     <wsa:ReplyTo xmlns:wsa="http://www.w3.org/2005/08/addressing">

```

```

2230     <wsa:Address>
2231         http://www.w3.org/2005/03/addressing/role/anonymous
2232     </wsa:Address>
2233 </wsa:ReplyTo>
2234 <wsse:Security xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd">
2235     <wsu:Timestamp xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd">
2236         <wsu:Created>2006-04-06T15:39:13Z</wsu:Created>
2237     </wsu:Timestamp>
2238 </wsse:Security>
2239 <sb2:Sender xmlns:sb2="urn:liberty:sb:2006-08"
2240     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2241     providerID="https://s-ds.liberty-iop.org:8681/idp.xml"
2242     soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
2243     wsu:Id="PRV"/>
2244 <sb2:Framework
2245     xmlns:sb2="urn:liberty:sb:2006-08"
2246     version="2.0"
2247     soap:actor="http://schemas.xmlsoap.org/soap/actor/next"/>
2248 </soap:Header>
2249 <soap:Body>
2250 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2251     <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2252     <wsa:EndpointReference
2253         xmlns:wsa="http://www.w3.org/2005/08/addressing"
2254         xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2255         notOnOrAfter="2006-04-06T17:39:14Z"
2256         wsu:Id="EPRIDIo61485WDEA701qHi4Ey">
2257     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2258     <wsa:Metadata>
2259         <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2260         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2261         <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2262         <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2263         <disco:SecurityContext>
2264             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2265             <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2266                 usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2267                 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2268                     ID="CREDI4cINqS4SV1HPm_xOoGh"
2269                     IssueInstant="2006-04-06T15:39:14Z"
2270                     Version="2.0">
2271                     ... assertion data was here ...
2272                 </sa:Assertion>
2273             </sec:Token>
2274         </disco:SecurityContext>
2275     </wsa:Metadata>
2276 </wsa:EndpointReference>
2277 </disco:QueryResponse>
2278 </soap:Body>
2279 </soap:Envelope>

```

2280 Things to note about this response:

- 2281 • the query was successful (status code is OK).
- 2282 • there is a single ID-WSF EPR in the response for the service type specified in the request
- 2283 • the details within the assertion were edited out to save space

2284 3.13.2. Query provider

2285 Query for the same service using the ProviderID of the WSP.

```

2286 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2287     <disco:RequestedService>
2288         <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>

```

```

2289     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2290     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2291     </disco:RequestedService>
2292 </disco:Query>

```

2293 Things to note about this query:

- 2294 • it is a query of a ProviderID, so all services provided by that ProviderID would be returned (and depending upon
- 2295 the invocation context of the ID-WSF framework, this may be all services provided by that provider for a particular
- 2296 principal or just all services).

- 2297 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for
- 2298 communicating with the specified service

2299 Server Response:

```

2300 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2301   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2302   <wsa:EndpointReference
2303     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2304     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2305     notOnOrAfter="2006-04-06T17:39:18Z"
2306     wsu:Id="EPRIDyGxcYpQ8Gace5y81Dm7Z">
2307     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2308     <wsa:Metadata>
2309       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2310       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2311       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2312       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2313       <disco:SecurityContext>
2314         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2315         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2316           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2317           <sa:Assertion
2318             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2319             ID="CREDOEV7U7-mJk02pEVJAn--"
2320             IssueInstant="2006-04-06T15:39:18Z"
2321             Version="2.0">
2322             ... assertion data was here ...
2323           </sa:Assertion>
2324         </sec:Token>
2325       </disco:SecurityContext>
2326     </wsa:Metadata>
2327   </wsa:EndpointReference>
2328 </disco:QueryResponse>

```

2329 Things to note about this response:

- 2330 • the query was successful (status code is OK).

- 2331 • there is a single ID-WSF EPR in the response denoting the single service offered by the provider specified in the
- 2332 request for this user.

- 2333 • The ID-WSF EPR is **almost** identical to the ID-WSF EPR returned in the previous request. The differences are
- 2334 only in the timestamps and the element IDs.

2335 3.13.3. Query (empty)

2336 A Query without specifying any search criteria (which should return all services available to the user).

```

2337 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq"/>

```

2338 Things to note about this query:

- 2339 • you still need to include the <disco:Query> element in the request, just that its contents are empty.

2340 Server Response:

```

2341 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2342   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2343   <wsa:EndpointReference
2344     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2345     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2346     notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDt3MOE1DHPL0EBD8whEvw">
2347     <wsa:Address>https://s-ds.liberty-iop.org:8681/DISCO-S</wsa:Address>
2348     <wsa:Metadata>
2349       <disco:Abstract>SYMfiam Discovery Service</disco:Abstract>
2350       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2351       <disco:ProviderID>https://s-ds.liberty-iop.org:8681/idp.xml</disco:ProviderID>
2352       <disco:ServiceType>urn:liberty:disco:2006-08</disco:ServiceType>
2353       <disco:SecurityContext>
2354         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2355         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2356           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2357           <sa:Assertion
2358             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2359             ID="CREDr2tN6rxYICAXxD6CtenC"
2360             IssueInstant="2006-04-06T15:39:21Z" Version="2.0">
2361             ... assertion data was here ...
2362           </sa:Assertion>
2363         </sec:Token>
2364       </disco:SecurityContext>
2365     </wsa:Metadata>
2366   </wsa:EndpointReference>
2367   <wsa:EndpointReference
2368     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2369     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2370     notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDQBqOfWHDp3GKFSqqnZyj">
2371     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2372     <wsa:Metadata>
2373       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2374       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2375       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2376       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2377       <disco:SecurityContext>
2378         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2379         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2380           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2381           <sa:Assertion
2382             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2383             ID="CREDAsw9RdingIu3m4HtUxX5"
2384             IssueInstant="2006-04-06T15:39:22Z" Version="2.0">
2385             ... assertion data was here ...
2386           </sa:Assertion>
2387         </sec:Token>
2388       </disco:SecurityContext>
2389     </wsa:Metadata>
2390   </wsa:EndpointReference>
2391 </disco:QueryResponse>

```

2392 Things to note about this response:

- 2393 • the query was successful (status code is OK).
- 2394 • two ID-WSF EPRs were returned, one for the Discovery Service and one for the People Service.
- 2395 • Discovery Service instances will usually expose ID-WSF EPRs which point to themselves in this way in order to
- 2396 expose alternative invocation methods and/or allow the client to obtain newer credentials.

2397 3.13.4. Query People Service and Provider

2398 A discovery query specifying both the Provider ID and the Service Type (so that only services of that type by that
2399 provider are returned)

```
2400 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2401   <disco:RequestedService>
2402     <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2403     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2404     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2405     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2406   </disco:RequestedService>
2407 </disco:Query>
```

2408 Things to note about this query:

- 2409 • both the <disco:ServiceType> element and the <disco:ProviderID> element are included.
- 2410 • The provider and service type are the same ones we've been using so we should get the same response we had
2411 on earlier requests.

2412 Server Response:

```
2413 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2414   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2415   <wsa:EndpointReference
2416     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2417     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2418     notOnOrAfter="2006-04-06T17:39:26Z" wsu:Id="EPRIDwrdfxWpBRhCXsEMW1cjo">
2419     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2420     <wsa:Metadata>
2421       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2422       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2423       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2424       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2425       <disco:SecurityContext>
2426         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2427         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2428           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2429           <sa:Assertion
2430             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2431             ID="CREDZbnmfBjqRINmQeWvyxCG"
2432             IssueInstant="2006-04-06T15:39:26Z" Version="2.0">
2433             ... assertion data was here ...
2434           </sa:Assertion>
2435         </sec:Token>
2436       </disco:SecurityContext>
2437     </wsa:Metadata>
2438   </wsa:EndpointReference>
2439 </disco:QueryResponse>
```

2440 Things to note about this response:

- 2441 • the query was successful (status code is OK).
- 2442 • The ID-WSF EPR is **almost** exactly the same as the ID-WSF EPR returned in the first 2 examples above.
2443 The differences are only in the timestamps and the element IDs. This is because both EPRs are for the same principal
2444 accessing the same service (we just discovered the EPR through a different query).

2445 3.13.5. SvcMDQuery (empty)

2446 A query for all SvcMD stored at the DS on behalf of the invoking provider.

2447 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08"/>

2448 Things to note about this query:

- 2449 • it's empty -- meaning that all registered SvcMDs are requested.
- 2450 • the query is executed in the context of the provider invoking the query (identified in the invocation context specified in the ID-WSF framework headers)

2452 Server Response:

```
2453 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2454 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2455 <disco:SvcMD svcMDID="SVCMDIDy_7yFABwuFPkgYIjKCjv">
2456 <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2457 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2458 <disco:ServiceContext>
2459 <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2460 <disco:EndpointContext>
2461 <disco:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</disco:Address>
2462 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2463 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2464 </disco:EndpointContext>
2465 </disco:ServiceContext>
2466 </disco:SvcMD>
2467 </disco:SvcMDQueryResponse>
```

2468 Things to note about this response:

- 2469 • the query was successful (status code is OK).
- 2470 • one SvcMD was returned for the ID-WSF People Service (which is the only SvcMD registered by the invoking provider).

2472 3.13.6. SvcMDQuery w/Bad SvcMDID

2473 A service metadata query using an invalid SvcMDID.

```
2474 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2475 <disco:SvcMDID>123</disco:SvcMDID>
2476 </disco:SvcMDQuery>
```

2477 Things to note about this query:

- 2478 • 123 is an invalid SvcMDID

2479 Server Response:

```
2480 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2481 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2482 <lu:Status code="NoResults"/>
2483 </lu:Status>
2484 </disco:SvcMDQueryResponse>
```

2485 Things to note about this response:

- 2486 • the query failed (status code is Failed).
- 2487 • The optional sub-status is included which indicates that there were *NoResults* for the query.

2488 3.13.7. SvcMDRegister w/single SvcMD

2489 Registration of a single service metadata instance in the DS.

```
2490 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2491   <disco:SvcMD>
2492     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2493     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2494     <disco:ServiceContext>
2495       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2496       <disco:EndpointContext>
2497         <disco:Address>https://payment.testing.com</disco:Address>
2498         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2499         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2500         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2501         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2502       </disco:EndpointContext>
2503     </disco:ServiceContext>
2504   </disco:SvcMD>
2505 </disco:SvcMDRegister>
```

2506 Things to note about this request:

- 2507 • a single SvcMD is registered
- 2508 • no svcMDID attribute is specified on the SvcMD during registration (it will be assigned by the DS if the request is
2509 successful).
- 2510 • the service being registered is a test payment service provided by the same provider we've seen earlier, supports a
2511 single service and framework version and exposes three different security mechanisms.

2512 Server Response:

```
2513 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2514   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2515   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2516 </disco:SvcMDRegisterResponse>
```

2517 Things to note about this response:

- 2518 • the registration was successful (status code is OK).
- 2519 • The SvcMD was assigned the svcMDID *SVCMDIDg9WP0thd_HvPd427KY9M*.

2520 3.13.8. SvcMDQuery w/Good SvcMDID

2521 Query the SvcMD that we just registered using the SvcMDID that was returned in the response.

```
2522 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2523   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2524 </disco:SvcMDQuery>
```

2525 Things to note about this query:

- 2526 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous
2527 step.

2528 Server Response:

```
2529 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2530   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2531   <disco:SvcMD svcMDID="SVCMDIDg9WP0thd_HvPd427KY9M">
```

```
2532 <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2533 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2534 <disco:ServiceContext>
2535   <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2536   <disco:EndpointContext>
2537     <disco:Address>https://payment.testing.com</disco:Address>
2538     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2539     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2540     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2541     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2542   </disco:EndpointContext>
2543 </disco:ServiceContext>
2544 </disco:SvcMD>
2545 </disco:SvcMDQueryResponse>
```

2546 Things to note about this response:

- 2547 • the query was successful (status code is OK).
- 2548 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2549 • The remaining data is identical to that registered by the provider in the previous request.

2550 3.13.9. SvcMDDelete w/Good SvcMDID

2551 Delete the Service Metadata that we just registered and queried.

```
2552 <disco:SvcMDDelete xmlns:disco="urn:liberty:disco:2006-08">
2553   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2554 </disco:SvcMDDelete>
```

2555 Things to note about this query:

- 2556 • the SvcMDID that we obtained in the registration above is used

2557 Server Response:

```
2558 <disco:SvcMDDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2559   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2560 </disco:SvcMDDeleteResponse>
```

2561 Things to note about this response:

- 2562 • the delete was successful (status code is OK).

2563 3.13.10. SvcMDDelete w/Already Deleted SvcMDID

2564 Delete the same service metadata that we just deleted.

```
2565 <disco:SvcMDDelete xmlns:disco="urn:liberty:disco:2006-08">
2566   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2567 </disco:SvcMDDelete>
```

2568 Things to note about this query:

- 2569 • the SvcMDID that we already deleted is specified

2570 Server Response:

```
2571 <disco:SvcMDDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2572   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2573 </disco:SvcMDDeleteResponse>
```

2574 Things to note about this response:

- 2575 • the delete was successful (status code is OK), even though the SvcMD was already deleted as the delete of a non-
- 2576 existant SvcMD is defined by this specification to be successful.

2577 3.13.11. SvcMDRegister w/Complex SvcMD

2578 A basic registration of a SvcMD with the Discovery Service. This is a little bit different in that the SvcMD is rather
 2579 complex (which is good for testing the Query interface).

```

2580 <disco: SvcMDRegister xmlns: disco="urn: liberty: disco: 2006-08">
2581   <disco: SvcMD>
2582     <disco: Abstract>TestDisco Test Calendar Service</disco: Abstract>
2583     <disco: ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco: ProviderID>
2584     <disco: ServiceContext>
2585       <disco: ServiceType>urn:x-test:cal:2006-01</disco: ServiceType>
2586       <disco: ServiceType>urn:x-test:cal:2006-09</disco: ServiceType>
2587       <disco: EndpointContext>
2588         <disco: Address>https://old-calendars.testing.com</disco: Address>
2589         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.1"/>
2590         <disco: SecurityMechID>urn: liberty: security: 2003-08: TLS: Bearer</disco: SecurityMechID>
2591         <disco: SecurityMechID>urn: liberty: security: 2003-08: TLS: null</disco: SecurityMechID>
2592       </disco: EndpointContext>
2593       <disco: EndpointContext>
2594         <disco: Address>https://old2-calendars.testing.com</disco: Address>
2595         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2596         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2597         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: null</disco: SecurityMechID>
2598       </disco: EndpointContext>
2599       <disco: EndpointContext>
2600         <disco: Address>http://old-calendars.testing.com</disco: Address>
2601         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2602         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: Bearer</disco: SecurityMechID>
2603         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: SAML2</disco: SecurityMechID>
2604       </disco: EndpointContext>
2605     </disco: ServiceContext>
2606     <disco: ServiceContext>
2607       <disco: ServiceType>urn:x-test:cal:2008-03</disco: ServiceType>
2608       <disco: EndpointContext>
2609         <disco: Address>https://calendars.testing.com</disco: Address>
2610         <disco: Address>https://calendars.testing.backup.com</disco: Address>
2611         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2612         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: SAML2</disco: SecurityMechID>
2613         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2614       </disco: EndpointContext>
2615       <disco: EndpointContext>
2616         <disco: Address>http://calendars.testing.com</disco: Address>
2617         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2618         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: SAML2</disco: SecurityMechID>
2619       </disco: EndpointContext>
2620     </disco: ServiceContext>
2621   </disco: SvcMD>
2622 </disco: SvcMDRegister>
    
```

2623 Things to note about this query:

- 2624 • it's a rather complex SvcMD, but registration is still the same.

2625 Server Response:

```

2626 <disco: SvcMDRegisterResponse xmlns: disco="urn: liberty: disco: 2006-08">
2627   <lu: Status xmlns: lu="urn: liberty: util: 2006-08" code="OK"/>
2628   <disco: SvcMDID>SVCMDIDmifyjzIKO6tNd8evymL</disco: SvcMDID>
2629 </disco: SvcMDRegisterResponse>
    
```

2630 Things to note about this response:

- 2631 • the registration was successful (status code is OK).
- 2632 • The SvcMD was assigned the svcMDID *SVCMDIDmifyjzIKO6tNd8evymnL*.
- 2633 • Now the fun begins as this SvcMD let's us exercise many of the interesting portions of the Discovery Query interface.

2634 **3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD**

2635 Query the SvcMD that we just registered using the SvcMDID that was returned in the response to make sure it was
2636 registered correctly.

```
2637 <disco: SvcMDQuery xmlns: disco="urn: liberty: disco: 2006-08">
2638   <disco: SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco: SvcMDID>
2639 </disco: SvcMDQuery>
```

2640 Things to note about this query:

- 2641 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous
2642 step.

2643 Server Response:

```
2644 <disco: SvcMDQueryResponse xmlns: disco="urn: liberty: disco: 2006-08">
2645   <lu: Status xmlns: lu="urn: liberty: util: 2006-08" code="OK"/>
2646   <disco: SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2647     <disco: Abstract>TestDisco Test Calendar Service</disco: Abstract>
2648     <disco: ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco: ProviderID>
2649     <disco: ServiceContext>
2650       <disco: ServiceType>urn:x-test:cal:2006-01</disco: ServiceType>
2651       <disco: ServiceType>urn:x-test:cal:2006-09</disco: ServiceType>
2652       <disco: EndpointContext>
2653         <disco: Address>https://old-calendars.testing.com</disco: Address>
2654         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.1"/>
2655         <disco: SecurityMechID>urn: liberty: security: 2003-08: TLS: Bearer</disco: SecurityMechID>
2656         <disco: SecurityMechID>urn: liberty: security: 2003-08: TLS: null</disco: SecurityMechID>
2657       </disco: EndpointContext>
2658       <disco: EndpointContext>
2659         <disco: Address>https://old2-calendars.testing.com</disco: Address>
2660         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2661         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2662         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: null</disco: SecurityMechID>
2663       </disco: EndpointContext>
2664       <disco: EndpointContext>
2665         <disco: Address>http://old-calendars.testing.com</disco: Address>
2666         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2667         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: Bearer</disco: SecurityMechID>
2668         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: SAML2</disco: SecurityMechID>
2669       </disco: EndpointContext>
2670     </disco: ServiceContext>
2671     <disco: ServiceContext>
2672       <disco: ServiceType>urn:x-test:cal:2008-03</disco: ServiceType>
2673       <disco: EndpointContext>
2674         <disco: Address>https://calendars.testing.com</disco: Address>
2675         <disco: Address>https://calendars.testing.backup.com</disco: Address>
2676         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2677         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: SAML2</disco: SecurityMechID>
2678         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2679       </disco: EndpointContext>
2680       <disco: EndpointContext>
2681         <disco: Address>http://calendars.testing.com</disco: Address>
2682         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2683         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: SAML2</disco: SecurityMechID>
```

```
2684     </disco:EndpointContext>
2685     </disco:ServiceContext>
2686     </disco:SvcMD>
2687 </disco:SvcMDQueryResponse>
```

2688 Things to note about this response:

- 2689 • the query was successful (status code is OK).
- 2690 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2691 • The remaining data is identical to that registered by the provider in the previous request. Key here is that the
- 2692 ordering and grouping of elements has been maintained.

2693 3.13.13. SvcMDReplace of existing SvcMD

2694 Replace the complex SvcMD that we just registered with a simpler version

```
2695 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
2696   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2697     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2698     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2699     <disco:ServiceContext>
2700       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2701       <disco:EndpointContext>
2702         <disco:Address>https://payment.testing.com</disco:Address>
2703         <sbfc:Framework xmlns:sbfc="urn:liberty:sbfc" version="2.0"/>
2704         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2705         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2706         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2707       </disco:EndpointContext>
2708     </disco:ServiceContext>
2709   </disco:SvcMD>
2710 </disco:SvcMDReplace>
```

2711 Things to note about this request:

- 2712 • the svcMDID attribute has the value obtained during the previous registration so that registration should be replaced.

2713 Server Response:

```
2714 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
2715   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2716 </disco:SvcMDReplaceResponse>
```

2717 Things to note about this response:

- 2718 • the replacement was successful (status code is OK).
- 2719 • no other data is returned. The SvcMDID does not change when the SvcMD is replaced.

2720 3.13.14. SvcMDQuery of Replaced SvcMDID

2721 Query the SvcMD that we just replaced.

```
2722 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2723   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2724 </disco:SvcMDQuery>
```

2725 Things to note about this query:

- 2726 • We use the same svcMDID that we had for the previous SvcMD as the value doesn't change when we do a re-
2727 placement.

2728 Server Response:

```
2729 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2730   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2731   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2732     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2733     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2734     <disco:ServiceContext>
2735       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2736       <disco:EndpointContext>
2737         <disco:Address>https://payment.testing.com</disco:Address>
2738         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2739         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2740         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2741         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2742       </disco:EndpointContext>
2743     </disco:ServiceContext>
2744   </disco:SvcMD>
2745 </disco:SvcMDQueryResponse>
```

2746 Things to note about this response:

- 2747 • the query was successful (status code is OK).
- 2748 • the new SvcMD is returned.

2749 3.13.15. SvcMDDelete of replaced SvcMD

2750 Delete the Service Metadata that we just replaced (to clean up after ourselves).

```
2751 <disco:SvcMDDelete xmlns:disco="urn:liberty:disco:2006-08">
2752   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2753 </disco:SvcMDDelete>
```

2754 Things to note about this query:

- 2755 • the SvcMDID that we used in the replacement above is used

2756 Server Response:

```
2757 <disco:SvcMDDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2758   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2759 </disco:SvcMDDeleteResponse>
```

2760 Things to note about this response:

- 2761 • the delete was successful (status code is OK).

2762 3.13.16. SvcMDRegister w/multiple SvcMDs

2763 A registration of 3 different SvcMD elements of varying levels of complexity

```
2764 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2765   <disco:SvcMD>
2766     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2767     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2768     <disco:ServiceContext>
2769       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
2770       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
2771     </disco:ServiceContext>
```

```

2772     <disco:Address>https://1-calendars.testing.com</disco:Address>
2773     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
2774     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2775     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2776 </disco:EndpointContext>
2777 <disco:EndpointContext>
2778     <disco:Address>https://2-calendars.testing.com</disco:Address>
2779     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2780     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2781     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2782 </disco:EndpointContext>
2783 <disco:EndpointContext>
2784     <disco:Address>http://3-calendars.testing.com</disco:Address>
2785     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2786     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
2787     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2788 </disco:EndpointContext>
2789 </disco:ServiceContext>
2790 <disco:ServiceContext>
2791     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2792     <disco:EndpointContext>
2793         <disco:Address>https://4-calendars.testing.com</disco:Address>
2794         <disco:Address>https://5-calendars.testing.backup.com</disco:Address>
2795         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2796         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2797         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2798     </disco:EndpointContext>
2799     <disco:EndpointContext>
2800         <disco:Address>http://6-calendars.testing.com</disco:Address>
2801         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2802         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2803     </disco:EndpointContext>
2804 </disco:ServiceContext>
2805 </disco:SvcMD>
2806 <disco:SvcMD>
2807     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2808     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2809     <disco:ServiceContext>
2810         <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2811         <disco:EndpointContext>
2812             <disco:Address>https://payment.testing.com</disco:Address>
2813             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2814             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2815             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2816             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2817         </disco:EndpointContext>
2818     </disco:ServiceContext>
2819 </disco:SvcMD>
2820 <disco:SvcMD>
2821     <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
2822     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2823     <disco:ServiceContext>
2824         <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
2825         <disco:Options>
2826             <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
2827             <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
2828             <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
2829         </disco:Options>
2830         <disco:EndpointContext>
2831             <disco:Address>https://test2.atm.CA.US.testing.com</disco:Address>
2832             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2833             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2834             <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2835         </disco:EndpointContext>
2836     </disco:EndpointContext>
2837         <disco:Address>https://readers.atm.CA.US.testing.com</disco:Address>
2838         <disco:Address>https://readers.atm.NY.US.testing.com</disco:Address>

```

```

2839     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2840     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2841     <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2842     <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
2843   </disco:EndpointContext>
2844   <disco:EndpointContext>
2845     <disco:Address>https://writers.atm.testing.com</disco:Address>
2846     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2847     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2848     <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
2849     <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
2850   </disco:EndpointContext>
2851 </disco:ServiceContext>
2852 </disco:SvcMD>
2853 </disco:SvcMDRegister>

```

2854 Things to note about this registration:

- 2855 • there SvcMDs use many of the features of the data structures to define various contexts in which the service can
- 2856 be reached and which actions and/or options are available at said services.
- 2857 • We do **NOT** represent that the SvcMDs registered here are typical or normal. They were explicitly created to
- 2858 examine/test some of the intricacies of handling queries against the SvcMD data structure.

2859 Server Response:

```

2860 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2861   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2862   <disco:SvcMDID>SVCMDIDmfPMblwCRSiwnu8D3BGO</disco:SvcMDID>
2863   <disco:SvcMDID>SVCMDIDeZQGkXuw750_uh3Q90LO</disco:SvcMDID>
2864   <disco:SvcMDID>SVCMDIDrpg8SjpeUdSmla_ZSFUN</disco:SvcMDID>
2865 </disco:SvcMDRegisterResponse>

```

2866 Things to note about this response:

- 2867 • the registration was successful (status code is OK).
- 2868 • the svcMDID for each of the added elements is returned in sequence. The first SvcMDID to the first <SvcMD> in the
- 2869 request, the second to the second, and so forth.

2870 3.13.17. Query Calendar Service

2871 Query for the test calendar service for this user.

```

2872 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2873   <disco:RequestedService>
2874     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2875     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2876   </disco:RequestedService>
2877 </disco:Query>

```

2878 Things to note about this query:

- 2879 • the service type on this query is one of those specified in one of the SvcMDs that were just registered in the previous
- 2880 call.

2881 Server Response:

```

2882 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2883   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2884     <lu:Status code="NoResults"/>
2885   </lu:Status>
2886 </disco:QueryResponse>

```

2887 Things to note about this response:

- 2888 • the query failed (status code was Failed). This is because while the SvcMD has been *registered*, it has **not** been
2889 *associated* with the user.

2890 **3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD**

2891 Associate a single SvcMD with the current principal.

```
2892 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">  
2893   <disco: SvcMDID>SVCMDIDMfPMb1wcrSiwnu8D3BGO</disco: SvcMDID>  
2894 </disco: SvcMDAssociationAdd>
```

2895 Things to note about this query:

- 2896 • the SvcMDID specified is the SvcMDID assigned to the test calendar service registered above (the first SvcMD in
2897 the multi-SvcMD registration).

2898 Server Response:

```
2899 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">  
2900   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>  
2901 </disco: SvcMDAssociationAddResponse>
```

2902 Things to note about this response:

- 2903 • the query was successful (status code is OK).
- 2904 • this service is now associated with the principal and available to subsequent Discovery Service queries.

2905 **3.13.19. SvcMDAssociationQuery w/SvcMDID**

2906 Query the SvcMD Associations to see if that SvcMD is now associated with the principal

```
2907 <disco: SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08">  
2908   <disco: SvcMDID>SVCMDIDMfPMb1wcrSiwnu8D3BGO</disco: SvcMDID>  
2909 </disco: SvcMDAssociationQuery>
```

2910 Things to note about this query:

- 2911 • the SvcMDID provided is the SvcMDID that was just associated with the user in the previous request

2912 Server Response:

```
2913 <disco: SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">  
2914   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>  
2915   <disco: SvcMDID>SVCMDIDMfPMb1wcrSiwnu8D3BGO</disco: SvcMDID>  
2916 </disco: SvcMDAssociationQueryResponse>
```

2917 Things to note about this response:

- 2918 • the query was successful (status code is OK).
- 2919 • the matching SvcMDIDs were returned (since you can query more than one and not all may have matched on a
2920 successful query)

2921 **3.13.20. SvcMDAssociationQuery w/o SvcMDID**

2922 Query all SvcMD Associations for the current principal.

2923 <disco:SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08"/>

2924 Things to note about this query:

- 2925 • No SvcMDIDs are specified which causes all SvcMD associations that were created by the invoking provider to
2926 be listed.

2927 Server Response:

```
2928 <disco:SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2929   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2930   <disco:SvcMDID>SVCMDIDy_7yFABwuFPkgYIjKCjv</disco:SvcMDID>
2931   <disco:SvcMDID>SVCMDIDMfPMb1wCRSiwnu8D3BGO</disco:SvcMDID>
2932 </disco:SvcMDAssociationQueryResponse>
```

2933 Things to note about this response:

- 2934 • the query was successful (status code is OK).
- 2935 • two SvcMDIDs were returned. One for the service recently associated and one for the ID-WSF People Service that
2936 had been previously associated. (This call, as well as the previous registration and association calls, was made in
2937 the context of that provider so both should be visible).

2938 3.13.21. Query Calendar Service (again)

2939 Query for the test calendar service for this user (which previously failed, but now the SvcMD has been associated with
2940 this principal).

```
2941 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2942   <disco:RequestedService>
2943     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2944     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2945   </disco:RequestedService>
2946 </disco:Query>
```

2947 Things to note about this query:

- 2948 • again we're looking for the test calendar service (which has now been associated with the principal).
- 2949 • This query was constructed to be resolvable only by the data within the 2nd <ServiceContext> element (it's the
2950 only one with the ServiceType "urn:x-test:cal:2008-03") and within there, the 2nd <EndpointContext> element
2951 (it's the only one with the SecurithMechID ...:null:SAMLV2).

2952 Server Response:

```
2953 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2954   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2955   <wsa:EndpointReference
2956     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2957     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2958     notOnOrAfter="2006-04-06T17:40:28Z"
2959     wsu:Id="EPRID3jLhZ6fsjx3xFNF22Hyx">
2960     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
2961     <wsa:Metadata>
2962       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2963       <sbfl:Framework xmlns:sbfl="urn:liberty:sb" version="2.0"/>
2964       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2965       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2966       <disco:SecurityContext>
2967         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2968         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2969           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
```

```

2970         <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2971             ID="CRED-HgB2SRBPptovTK7ckof"
2972             IssueInstant="2006-04-06T15:40:28Z"
2973             Version="2.0">
2974             ... assertion data was here ...
2975         </sa:Assertion>
2976     </sec:Token>
2977 </disco:SecurityContext>
2978 </wsa:Metadata>
2979 </wsa:EndpointReference>
2980 </disco:QueryResponse>

```

2981 Things to note about this response:

- 2982 • the query was successful (status code is OK).
- 2983 • the ID-WSF EPR was generated from the expected SvcMD elements (the unique address *http://6-calendars.testing.com* shows this).

2985 3.13.22. Query Calendar Service w/Action

2986 Query for a Calendar Service including an <Action> element in the request.

```

2987 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2988   <disco:RequestedService>
2989     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2990     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2991     <disco:Action>urn:x-test:cal:2008-03:GetMeeting</disco:Action>
2992   </disco:RequestedService>
2993 </disco:Query>

```

2994 Things to note about this query:

- 2995 • the action specified on the request is **not** explicitly listed in the registered SvcMD.

2996 Server Response:

```

2997 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2998   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2999   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3000       xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3001       notOnOrAfter="2006-04-06T17:41:01Z"
3002       wsu:Id="EPRIDq9XReiwZpP_tftRcutoP">
3003     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3004     <wsa:Metadata>
3005       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3006       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3007       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3008       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3009       <disco:SecurityContext>
3010         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3011         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3012             usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3013           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3014               ID="CREDst8YWUPHbqUdMKjnV_I7"
3015               IssueInstant="2006-04-06T15:41:01Z"
3016               Version="2.0">
3017             ... assertion data was here ...
3018           </sa:Assertion>
3019         </sec:Token>
3020       </disco:SecurityContext>
3021     </wsa:Metadata>
3022   </wsa:EndpointReference>
3023 </disco:QueryResponse>

```

3024 Things to note about this response:

- 3025 • the query was successful (status code is OK).
- 3026 • the returned ID-WSF EPR is essentially identical to the ID-WSF EPR returned in the previous query (only differing
- 3027 in timestamps and element IDs).
- 3028 • the <Action> specified on the request was considered to be matched because **no** <Action> elements were speci-
- 3029 fied in the registered SvcMD -- which by definition means that the SvcMD matches all possible <Action> values.

3030 **3.13.23. Query Calendar Service w/resultsType=all**

3031 Query for the Calendar Service specifying the "...:TLS:SAMLV2" SecurityMechID and resultsType=all

```

3032 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3033   <disco:RequestedService resultsType="all">
3034     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3035     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3036   </disco:RequestedService>
3037 </disco:Query>

```

3038 Things to note about this query:

- 3039 • resultsType setting of "all" indicates that the requestor wants all possible results, not just a limited match. This is
- 3040 typically done when the client wants to choose which of the results to use.
- 3041 • This query was constructed to be resolvable only by the data within the 1st <EndpointContext> element (the
- 3042 SecurithMechID ...:TLS:SAMLV2) in the 2nd <ServiceContext> element (the ServiceType "urn:x-test:cal:
- 3043 2008-03").

3044 Server Response:

```

3045 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3046   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3047   <wsa:EndpointReference
3048     xmlns:wsa="http://www.w3.org/2005/08/addressing"
3049     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3050     notOnOrAfter="2006-04-06T17:41:05Z" wsu:Id="EPRIDgNpGPKjwRsZVIC63VOMt">
3051     <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3052     <wsa:Metadata>
3053       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3054       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3055       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3056       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3057       <disco:SecurityContext>
3058         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3059         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3060           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3061           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3062             ID="CREDYA0WkksXLutGxWwSF2m"
3063             IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3064             ... assertion data was here ...
3065           </sa:Assertion>
3066         </sec:Token>
3067       </disco:SecurityContext>
3068     </wsa:Metadata>
3069   </wsa:EndpointReference>
3070   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3071     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3072     notOnOrAfter="2006-04-06T17:41:06Z" wsu:Id="EPRIDeBlMvrPuq-TphWEqQ7G0">
3073     <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3074     <wsa:Metadata>
3075       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>

```

```

3076 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3077 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3078 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3079 <disco:SecurityContext>
3080 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3081 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3082 usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3083 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3084 ID="CRED816B4EvhFszmG01Ab5F7"
3085 IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3086 ... assertion data was here ...
3087 </sa:Assertion>
3088 </sec:Token>
3089 </disco:SecurityContext>
3090 </wsa:Metadata>
3091 </wsa:EndpointReference>
3092 </disco:QueryResponse>

```

3093 Things to note about this response:

- 3094 • the query was successful (status code is OK).
- 3095 • as expected 2 ID-WSF EPRs were returned because the two endpoints which matched the search criteria cannot
- 3096 be placed into the same EPR.

3097 3.13.24. Query for all Calendar Service EPRs

3098 A query of all of the data available for the calendar service

```

3099 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3100 <disco:RequestedService resultsType="all">
3101 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3102 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3103 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3104 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3105 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3106 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3107 <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3108 <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3109 </disco:RequestedService>
3110 </disco:Query>

```

3111 Things to note about this query:

- 3112 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3113 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3114 Server Response:

```

3115 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3116 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3117 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3118 xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3119 notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDC08vEmUOfarryqg3zo8j">
3120 <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3121 <wsa:Metadata>
3122 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3123 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
3124 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3125 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3126 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3127 <disco:SecurityContext>
3128 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>

```

```

3129     <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3130         usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3131         <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3132             ID="CREd1QVGIZghandZqdE90QRa"
3133             IssueInstant="2006-04-06T15:41:09Z" Version="2.0">
3134             ... assertion data was here ...
3135         </sa:Assertion>
3136     </sec:Token>
3137 </disco:SecurityContext>
3138 <disco:SecurityContext>
3139     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3140 </disco:SecurityContext>
3141 </wsa:Metadata>
3142 </wsa:EndpointReference>
3143 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3144     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3145     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDqEcv9d0T430TQLdhB116">
3146     <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3147     <wsa:Metadata>
3148     ... Metatdata was here ....
3149     </wsa:Metadata>
3150 </wsa:EndpointReference>
3151 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3152     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3153     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDRPIps8-wMwgf4A_7DsHS">
3154     <wsa:Address>http://3-calendars.testing.com</wsa:Address>
3155     <wsa:Metadata>
3156     ... Metatdata was here ....
3157     </wsa:Metadata>
3158 </wsa:EndpointReference>
3159 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3160     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3161     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDkLl5ahJdrutDV6gMPLyr">
3162     <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3163     <wsa:Metadata>
3164     ... Metatdata was here ....
3165     </wsa:Metadata>
3166 </wsa:EndpointReference>
3167 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3168     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3169     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID0TmCLBCVoYZV7_R8jgky">
3170     <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3171     <wsa:Metadata>
3172     ... Metatdata was here ....
3173     </wsa:Metadata>
3174 </wsa:EndpointReference>
3175 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3176     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3177     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID9nsZ7UfGOG5UMMymJLRp">
3178     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3179     <wsa:Metadata>
3180     ... Metatdata was here ....
3181     </wsa:Metadata>
3182 </wsa:EndpointReference>
3183 </disco:QueryResponse>

```

3184 Things to note about this response:

- 3185 • the query was successful (status code is OK).
- 3186 • Much of the data in this response was elided in order to not waste alot of paper. The first EPR in the response is shown fairly completely.
- 3187
- 3188 • There are 6 EPRs (the minimum way to represent all of the data in the SvcMD that matched the requested parameters).

- 3189 • The first 3 EPRs have two service types (for the 2006-01 and the 2006-09 versions of the calendar service) as a
 3190 single ID-WSF EPR may have multiple service types if they are all related to the same logical service.

3191 **3.13.25. Query for one Calendar Service EPRs**

- 3192 A query for the first EPR out of all those within the Calendar service (to show the impact of the `resultsType` setting
 3193 of *only-one*.

```
3194 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3195   <disco:RequestedService resultsType="only-one">
3196     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3197     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3198     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3199     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3200     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3201     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3202     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3203     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3204   </disco:RequestedService>
3205 </disco:Query>
```

3206 Things to note about this query:

- 3207 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3208 • The `resultsType` attribute is set to "*only-one*" indicating that the Discovery Service should only return the first
 3209 matching ID-WSF EPR.

3210 Server Response:

```
3211 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3212   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3213   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3214     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3215     notOnOrAfter="2006-04-06T17:41:14Z" wsu:Id="EPRIDVh71zxFCQu4yWBOKLPzH">
3216     <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3217     <wsa:Metadata>
3218       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3219       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
3220       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3221       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3222       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3223       <disco:SecurityContext>
3224         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3225         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3226           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3227           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3228             ID="CREDYjkZsJiWfQwKf5gaJoze"
3229             IssueInstant="2006-04-06T15:41:14Z" Version="2.0">
3230             ... assertion data was here ...
3231           </sa:Assertion>
3232         </sec:Token>
3233       </disco:SecurityContext>
3234       <disco:SecurityContext>
3235         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3236       </disco:SecurityContext>
3237     </wsa:Metadata>
3238   </wsa:EndpointReference>
3239 </disco:QueryResponse>
```

3240 Things to note about this response:

- 3241 • the query was successful (status code is OK).

3242 • Only one ID-WSF EPR was returned (because of the query parameters) (as compared to the 6 returned in the
3243 previous example with the same query other than the `resultsType` attribute).

3244 • The one ID-WSF EPR that is returned is the **first** EPR that would have otherwise been returned.

3245 3.13.26. Query specific version of Calendar Service

3246 A query for the 2006-09 version of the Calendar Service.

```
3247 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3248   <disco:RequestedService>
3249     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3250     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3251   </disco:RequestedService>
3252 </disco:Query>
```

3253 Things to note about this query:

3254 • This query was constructed to be resolvable only by the data within the 1st `<ServiceContext>` element (it's the
3255 only one with the `ServiceType "urn:x-test:cal:2006-09"`) and within there, both the 1st and the 2nd
3256 `<EndpointContext>` element (they are the only ones with the `SecurithMechID:TLS:Bearer`).

3257 Server Response:

```
3258 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3259   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3260   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3261     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3262     notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDVsvR1CWheodUvbDFelh">
3263     <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3264     <wsa:Metadata>
3265       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3266       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
3267       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3268       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3269       <disco:SecurityContext>
3270         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3271         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3272           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3273           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3274             ID="CREDFy-6w5S8iOVearNxJur_"
3275             IssueInstant="2006-04-06T15:41:19Z" Version="2.0">
3276             ... assertion data was here ...
3277           </sa:Assertion>
3278         </sec:Token>
3279       </disco:SecurityContext>
3280     </wsa:Metadata>
3281   </wsa:EndpointReference>
3282   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3283     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3284     notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDDKWLXiXVxpSJJi3oY3ge">
3285     <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3286     <wsa:Metadata>
3287       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3288       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3289       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3290       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3291       <disco:SecurityContext>
3292         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3293         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3294           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3295           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3296             ID="CREDW_y92CoMn7x57YOWzbnB"
3297             IssueInstant="2006-04-06T15:41:20Z" Version="2.0">
```

```
3298         ... assertion data was here ...
3299         </sa:Assertion>
3300     </sec:Token>
3301 </disco:SecurityContext>
3302 </wsa:Metadata>
3303 </wsa:EndpointReference>
3304 </disco:QueryResponse>
```

3305 Things to note about this response:

- 3306 • the query was successful (status code is OK).
- 3307 • Two ID-WSF EPRs were returned by the Discovery Service because the matching data had two different endpoints
3308 which must be represented in separate EPRs.
- 3309 • Because there was no `resultsType` specified on the request, the DS could have returned just the first ID-WSF
3310 EPR if they chose to as that ID-WSF EPR meets the basic requirements of the request. In this particular instance
3311 the Discovery Service acted as if "all" had been specified, but that should not be depended upon.
- 3312 If you need a particular `resultsType` behavior, you need to specify it on the request.
- 3313 • The Discovery Service could have used the same assertion for the two ID-WSF EPRs if appropriate. In such a case,
3314 the one of the ID-WSF EPRs would have had a `<sec:Token>` element with a `ref` attribute containing a pointer
3315 to the `<sec:Token>` in the other ID-WSF EPR.

3316 3.13.27. SvcMDReplace Calendar Service

3317 Replace the SvcMD for the Calendar service with a much simpler SvcMD.

```
3318 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
3319     <disco:SvcMD svcMDID="SVCMDIDMfPmb1wcRSiwnu8D3BGO">
3320         <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3321         <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3322         <disco:ServiceContext>
3323             <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3324             <disco:EndpointContext>
3325                 <disco:Address>https://calendar.testing.com</disco:Address>
3326                 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3327                 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3328             </disco:EndpointContext>
3329         </disco:ServiceContext>
3330     </disco:SvcMD>
3331 </disco:SvcMDReplace>
```

3332 Things to note about this query:

- 3333 • the SvcMDID is the SvcMDID for the complex Calendar Service SvcMD that we have been querying over the past
3334 few requests.
- 3335 • the new SvcMD is much simpler.

3336 Server Response:

```
3337 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
3338     <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3339 </disco:SvcMDReplaceResponse>
```

3340 Things to note about this response:

- 3341 • the replacement was successful (status code is OK).

3342 **3.13.28. Query old Calendar Service**

3343 Query the Calendar Service data that was replaced.

```
3344 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3345   <disco:RequestedService>
3346     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3347     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3348   </disco:RequestedService>
3349 </disco:Query>
```

3350 Things to note about this query:

- 3351 • this is the same query we ran earlier that obtained results.

3352 Server Response:

```
3353 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3354   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3355     <lu:Status code="NoResults"/>
3356   </lu:Status>
3357 </disco:QueryResponse>
```

3358 Things to note about this response:

- 3359 • the query failed (status code is Failed)
- 3360 • the sub-status is NoResults - indicating that no matching data was found
- 3361 • The new (replacement) SvcMD for the Calendar Service has taken effect for the principal without the need for it
- 3362 to be associated with the principal (as it is already associated).

3363 **3.13.29. Query for all Calendar Service EPRs**

3364 A query of all of the data available for the calendar service (same query we ran a few requests ago).

```
3365 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3366   <disco:RequestedService resultsType="all">
3367     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3368     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3369     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3370     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3371     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3372     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3373     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3374     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3375   </disco:RequestedService>
3376 </disco:Query>
```

3377 Things to note about this query:

- 3378 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3379 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3380 Server Response:

```
3381 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3382   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3383   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3384     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3385     notOnOrAfter="2006-04-06T17:41:32Z" wsu:Id="EPRID23j-JGWAXusqiV80ARzC">
```

```

3386 <wsa:Address>https://calendar.testing.com</wsa:Address>
3387 <wsa:Metadata>
3388 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3389 <sbfl:Framework xmlns:sbfl="urn:liberty:sbfl" version="2.0"/>
3390 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3391 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3392 <disco:SecurityContext>
3393 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3394 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3395     usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3396 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3397     ID="CREDxHDqbb4F1TV7jSYxlmxq"
3398     IssueInstant="2006-04-06T15:41:32Z" Version="2.0">
3399     ... assertion data was here ...
3400 </sa:Assertion>
3401 </sec:Token>
3402 </disco:SecurityContext>
3403 </wsa:Metadata>
3404 </wsa:EndpointReference>
3405 </disco:QueryResponse>

```

3406 Things to note about this response:

- 3407 • the query was successful (status code is OK).
- 3408 • Only the data from the replacement SvcMD is represented in the one ID-WSF EPR in the results.

3409 3.13.30. SvcMDAssociationAdd the ATM Service SvcMD

3410 Associate the ATM Service SvcMD with the current principal.

```

3411 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">
3412 <disco:SvcMDID>SVCMDIDrpg8SJpeUdSmla_ZSFUN</disco:SvcMDID>
3413 </disco:SvcMDAssociationAdd>

```

3414 Things to note about this query:

- 3415 • the SvcMDID specified is the SvcMDID assigned to the test ATM service registered above (the third SvcMD in
- 3416 the multi-SvcMD registration that was done earlier).

3417 Server Response:

```

3418 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">
3419 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3420 </disco:SvcMDAssociationAddResponse>

```

3421 Things to note about this response:

- 3422 • the association was successful (status code is OK).
- 3423 • the ATM Service SvcMD is now associated with the principal and available to subsequent Discovery Service
- 3424 queries.

3425 3.13.31. Query ATM Service w/resultsType=best

3426 Query for the ATM Service with the resultsType setting of "best."

```

3427 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3428 <disco:RequestedService resultsType="best">
3429 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3430 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3431 </disco:RequestedService>
3432 </disco:Query>

```

3433 Things to note about this query:

3434 • No <disco:Action>s are specified which implies the caller wants to have access to all operations at the provider.

3435 This is important for the ATM Service because in the SvcMD, each endpoint was registered with a subset of actions
 3436 (no one endpoint has them all, so the results will take several EPRs).

3437 Server Response:

```

3438 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3439   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3440   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3441     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3442     notOnOrAfter="2006-04-06T17:41:43Z" wsu:Id="EPRIDfmT9HOSbmMs3jZ1_qSuY">
3443     <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
3444     <wsa:Metadata>
3445       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3446       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3447       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3448       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3449       <disco:SecurityContext>
3450         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3451         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3452           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3453           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3454             ID="CREDS8x8pCKTOzaQMI14fkel"
3455             IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3456             ... assertion data was here ...
3457           </sa:Assertion>
3458         </sec:Token>
3459       </disco:SecurityContext>
3460       <disco:Options>
3461         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3462         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3463         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3464       </disco:Options>
3465       <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3466     </wsa:Metadata>
3467   </wsa:EndpointReference>
3468   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3469     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3470     notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDdHYSEKB8_w5bPicTPe8o">
3471     <wsa:Address>https://readers.atm.CA.US.testing.com</wsa:Address>
3472     <wsa:Metadata>
3473       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3474       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3475       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3476       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3477       <disco:SecurityContext>
3478         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3479         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3480           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3481           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3482             ID="CRED4nQ1FHP3vzuoqRhTMZrf"
3483             IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3484             ... assertion data was here ...
3485           </sa:Assertion>
3486         </sec:Token>
3487       </disco:SecurityContext>
3488       <disco:Options>
3489         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3490         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3491         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3492       </disco:Options>
3493       <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3494       <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
    
```

```

3495     </wsa:Metadata>
3496 </wsa:EndpointReference>
3497 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3498     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3499     notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDggifVjR-zSAxkokPyfCo">
3500 <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3501 <wsa:Metadata>
3502 <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3503 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3504 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3505 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3506 <disco:SecurityContext>
3507 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3508 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3509     usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3510 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3511     ID="CREDZNT1b1H6VxPNrpwawtF6"
3512     IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3513     ... assertion data was here ...
3514 </sa:Assertion>
3515 </sec:Token>
3516 </disco:SecurityContext>
3517 <disco:Options>
3518 <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3519 <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3520 <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3521 </disco:Options>
3522 <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3523 <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3524 </wsa:Metadata>
3525 </wsa:EndpointReference>
3526 </disco:QueryResponse>

```

3527 Things to note about this response:

- 3528 • the query was successful (status code is OK).
- 3529 • It took three ID-WSF EPRs to represent the set of actions at the ATM Service.
- 3530 • One might think that because the "...GetBalance" action is on both the 1st and 2nd ID-WSF EPR and it is the only
3531 action on the 1st ID-WSF EPR, the results could have excluded that ID-WSF EPR and the client would still get to
3532 all of the resources at the ATM Service.
- 3533 However, the WSP placed the 1st "...GetBalance" action in the first <EndpointContext> and therefore gives it
3534 a higher priority in the results.
- 3535 • The Discovery service could have left off the "...GetBalance" action on the 2nd ID-WSF EPR but that wouldn't
3536 have been much of a savings and so it was included. Clients should NOT depend upon this type of behavior.
- 3537 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the
3538 ID-WSF EPRs generated from that SvcMD.

3539 3.13.32. Query ATM Service w/Withdraw Action

3540 Query for the ATM Service where "...Withdraw" action is available.

```

3541 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3542 <disco:RequestedService resultsType="all">
3543 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3544 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3545 <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3546 </disco:RequestedService>
3547 </disco:Query>

```

3548 Things to note about this query:

- 3549 • The <disco:Action> element is specified with the "urn:x-test:atm:2007-11:Withdraw") action value. So the
- 3550 client only intends to use this operation at the ATM Service.

3551 Server Response:

```

3552 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3553   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3554   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3555     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3556     notOnOrAfter="2006-04-06T17:41:52Z" wsu:Id="EPRIIDkueZCk5N4IpSmX-0BD-9">
3557     <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3558     <wsa:Metadata>
3559       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3560       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3561       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3562       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3563       <disco:SecurityContext>
3564         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3565         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3566           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3567           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3568             ID="CREDYfLT1S7tcZ8LkuU-zlrs"
3569             IssueInstant="2006-04-06T15:41:52Z" Version="2.0">
3570             ... assertion data was here ...
3571           </sa:Assertion>
3572         </sec:Token>
3573       </disco:SecurityContext>
3574       <disco:Options>
3575         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3576         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3577         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3578       </disco:Options>
3579       <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3580       <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3581     </wsa:Metadata>
3582   </wsa:EndpointReference>
3583 </disco:QueryResponse>

```

3584 Things to note about this response:

- 3585 • the query was successful (status code is OK).
- 3586 • Only one ID-WSF EPR was returned which contained the endpoint where the "...Withdraw" action is available.
- 3587 • The Discovery service could have left off the "...Transfer" action but that wouldn't have been much of a savings
- 3588 and so it was included. Clients should NOT depend upon this type of behavior.
- 3589 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the
- 3590 ID-WSF EPRs generated from that SvcMD.

3591 3.13.33. Query ATM Service w/Option

3592 Query for the ATM service specifying an option

```

3593 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3594   <disco:RequestedService resultsType="only-one">
3595     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3596     <disco:Options>
3597       <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3598     </disco:Options>
3599     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>

```

3600 </disco:RequestedService>
 3601 </disco:Query>

3602 Things to note about this query:

3603 • The resultsType attribute is set to "only-one" indicating that the Discovery Service should only return the first
 3604 matching ID-WSF EPR.

3605 • the <Option> element is included with one of the values present in the SvcMD for the ATM Service.

3606 Server Response:

```

3607 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3608 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3609 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3610     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3611     notOnOrAfter="2006-04-06T17:42:19Z" wsu:Id="EPRIDzKyaJ_h5Qb2jLjLjq59E">
3612 <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
3613 <wsa:Metadata>
3614 <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3615 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3616 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3617 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3618 <disco:SecurityContext>
3619 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3620 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3621     usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3622 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3623     ID="CRED0FAqvGgoeySpLTZHbJa7"
3624     IssueInstant="2006-04-06T15:42:19Z" Version="2.0">
3625     ... assertion data was here ...
3626 </sa:Assertion>
3627 </sec:Token>
3628 </disco:SecurityContext>
3629 <disco:Options>
3630 <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3631 <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3632 <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3633 </disco:Options>
3634 <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3635 </wsa:Metadata>
3636 </wsa:EndpointReference>
3637 </disco:QueryResponse>
    
```

3638 Things to note about this response:

3639 • the query was successful (status code is OK).

3640 • The one option specified in the request matched one of the options specified in the SvcMD, so that is considered a
 3641 match. The caller does not have to specify all of the options in the SvcMD (but the SvcMD does have to have all
 3642 of the options listed on the request).

3643 • Even though only one option was specified on the request, all of the options listed in the SvcMD are included in
 3644 the response.

3645 3.13.34. Query ATM Service w/unknown Option

3646 Query for the ATM service with an option that doesn't exist in the SvcMD.

```

3647 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3648 <disco:RequestedService resultsType="all">
3649 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3650 <disco:Options>
    
```

```
3651     <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>
3652   </disco:Options>
3653   <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3654 </disco:RequestedService>
3655 </disco:Query>
```

3656 Things to note about this query:

- 3657 • The option specified is **not** in the ATM Service SvcMD.

3658 Server Response:

```
3659 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3660   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3661     <lu:Status code="NoResults"/>
3662   </lu:Status>
3663 </disco:QueryResponse>
```

3664 Things to note about this response:

- 3665 • the query failed (status code is Failed).
- 3666 • The sub-status was "NoResults" indicating no data matched the requested parameters

3667 3.13.35. Query ATM Service w/good and bad Option

3668 Query for the ATM service with an option that does exist and an option that doesn't exist in the SvcMD.

```
3669 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3670   <disco:RequestedService resultsType="all">
3671     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3672     <disco:Options>
3673       <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3674       <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>
3675     </disco:Options>
3676     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3677   </disco:RequestedService>
3678 </disco:Query>
```

3679 Things to note about this query:

- 3680 • both specified options have to be available for this request to be considered matched (if, on the other hand, the
3681 request had the two <disco:Option> elements in separate <disco:Options> containers, a SvcMD could match
3682 either one).

3683 Server Response:

```
3684 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3685   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3686     <lu:Status code="NoResults"/>
3687   </lu:Status>
3688 </disco:QueryResponse>
```

3689 Things to note about this response:

- 3690 • the query failed (status code is Failed).
- 3691 • The sub-status was "NoResults" indicating no data matched the requested parameters

3692 4. Discovery Service ID-WSF EPR conveyed via a Security Token

3693 In both single sign-on and web services environments, many recipients of a security token find the need to subsequently
3694 invoke the identified principal's Discovery Service in order to discover and invoke identity services on behalf of said
3695 principal. For example, a SAML SP upon receiving an SSO assertion may want to discover and invoke the principals
3696 Profile Service and would need the Discovery Service ID-WSF EPR in order to do so.

3697 In the SSO environment, this concept is often referred to as the "Discovery Service Bootstrap" in that the SP is using
3698 the data in the SSO assertion to bootstrap into the ID-WSF environment.

3699 The need for this Discovery Service ID-WSF EPR is not restricted to SSO environments as any WSP that is invoked
3700 by a WSC may in turn need to act as a WSC and invoke other WSPs in order to fulfill the requested operation. For
3701 example, a Profile Service WSP may need to invoke the Interaction Service in order to request consent from the user
3702 before releasing data to a WSC.

3703 This section describes the recommended interoperable method for an Identity Provider and/or Discovery Service can
3704 embed an ID-WSF EPR for the Discovery Service within security and/or Identity tokens that they issue. Unfortunately,
3705 because of the variance in structure and formats of various tokens, the model used tends to be specific to the format of
3706 the security token. The remainder of this section documents how this is accomplished within some specific token
3707 formats.

3708 4.1. EPR Generation Rules

3709 The Discovery Service Bootstrap ID-WSF EPR which is placed into any security token must be generated according
3710 to the following rules:

- 3711 • The `<wsa:EndpointReference>` that MAY contain `<SecurityContext>` element(s) in turn containing
3712 `<sec:Token>` elements containing embedded security tokens, which are necessary to access the Discovery Service
3713 instance(s).
- 3714 • The `<sec:Token>` element MAY instead include a reference to an external security token using a `<wsse:
3715 SecurityTokenReference>` containing a non-relative URI reference to a security token.
- 3716 • The `<sec:Token>` element's `ref` attribute MAY instead refer to local security token available elsewhere in the
3717 same security token (such as another ID-WSF EPR within the security token). These references SHOULD only
3718 refer to elements within the security token carrying the ID-WSF EPR so that the reference will remain valid if the
3719 security token is separated from any message carrying the token.

3720 It is even possible (and in some cases typical) for the reference to be to the enveloping security token itself (the
3721 security token that contains this ID-WSF EPR) In such cases, the enveloping security token SHOULD carry the
3722 necessary information to support its consumption at the Discovery Service (as well as the information necessary
3723 for consumption at its primary relying party (the SP/WSP)).

3724 For example, with a SAML Assertion, this includes:

- 3725 • A second `<Audience>` element with the Discovery Service's ProviderID.
- 3726 • A subject confirmation method that the relying party can meet. This will frequently be `urn:oasis:names:
3727 tc:SAML:2.0:cm:bearer` in which case the same confirmation can be used by both parties. However, the
3728 assertion could contain multiple confirmation methods one for the initial party to use when invoking the relying
3729 party and one for the relying party to use when invoking the DS.

3730 This will allow the Discovery Service to validate the assertion using the normal assertion processing rules without
3731 having to manage some form of exception for self issued assertions.

3732 4.2. SAML 2.0 Security Tokens

3733 In a SAML 2.0 Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the
3734 <saml2:AttributeStatement> element in a <saml2:Assertion>.

3735 The <saml2:AttributeStatement> SHOULD be constructed according to the following rules:

- 3736 • The Name attribute of the <saml2:Attribute> element MUST be:

3737 *urn:liberty:disco:2006-08:DiscoveryEPR*

- 3738 • The NameFormat attribute of the <saml2:Attribute> element MUST be:

3739 *urn:oasis:names:tc:SAML:2.0:attrname-format:uri*

- 3740 • One or more <saml2:AttributeValue> elements MUST be included which each containing a single <wsa:
3741 EndpointReference> element identifying a Discovery Service instance(s). These Discovery Service instances
3742 SHOULD offer identity services for the Principal identified in the Subject element inside the <saml2:
3743 Assertion>.

3744 An example <saml2:AttributeStatement> that might be found in a SAMLv2 <saml2:Assertion> follows.
3745 The example includes a <sec:Token> element which has a reference to the surrounding assertion.

```

3746
3747 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3748   <Attribute Name="urn:liberty:disco:2006-08:DiscoveryEPR"
3749     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
3750     <AttributeValue>
3751       <wsa:EndpointReference>
3752         <wsa:Address>https://example.com/disco/</wsa:Address>
3753
3754         <wsa:Metadata>
3755           <Abstract>
3756             The Principal's Discovery Service Resource
3757           </Abstract>
3758
3759           <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3760
3761           <ProviderID>http://example.com/</ProviderID>
3762
3763           <SecurityContext>
3764             <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer</SecurityMechID>
3765             <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3766             </SecurityContext>
3767           </wsa:Metadata>
3768         </wsa:EndpointReference>
3769       </AttributeValue>
3770     </Attribute>
3771   </AttributeStatement>
3772

```

3773 **Example 21. <AttributeStatement> that might be found in a SAMLv2 AuthnResponse**

3774 In all cases, this <AttributeStatement> MUST carry an ID-WSF EPR for the Liberty Discovery Service. Any
3775 other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

3776 4.3. SAML 1.x (Liberty ID-FF) Security Tokens

3777 In a SAML 1.x Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the
3778 <saml:AttributeStatement> element in a <saml:Assertion>.

3779 The <saml:AttributeStatement> SHOULD be constructed according to the following rules:

3780 • For the <saml:Attribute> element:

3781 • The AttributeName attribute MUST be "DiscoveryEPR."

3782 • The AttributeNamespace attribute MUST be "urn:liberty:disco:2006-08."

3783 • The <Subject> element of the <saml:AttributeStatement> element MUST carry the identity of the principal
3784 whose Discovery Service is referenced by this EPR and SHOULD be the same identity in the subject of the other
3785 statements in the <saml:Assertion>.

3786 • One or more <saml:AttributeValue> elements MUST be included which each containing a single <wsa:
3787 EndpointReference> element identifying a Discovery Service instance(s). These Discovery Service instances
3788 SHOULD offer identity services for the Principal identified in the Subject element inside this <saml:
3789 AttributeStatement>.

3790 An example <saml:AttributeStatement> that might be found in a SAML 1.1 <saml:Assertion> follows. The
3791 example includes a <sec:Token> element which has a reference to the surrounding assertion.

```
3792
3793 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
3794   <Subject>
3795     <NameIdentifier Format="urn:liberty:iff:nameid:federated">
3796       d0CQF8elJTDLmzE0
3797     </NameIdentifier>
3798   </Subject>
3799   <Attribute AttributeName="DiscoveryEPR"
3800     AttributeNamespace="urn:liberty:disco:2006-08">
3801     <AttributeValue>
3802       <wsa:EndpointReference>
3803         <wsa:Address>https://example.com/disco/</wsa:Address>
3804
3805         <wsa:Metadata>
3806           <Abstract>
3807             The Principal's Discovery Service Resource
3808           </Abstract>
3809
3810           <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3811
3812           <ProviderID>http://example.com/</ProviderID>
3813
3814           <SecurityContext>
3815             <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer</SecurityMechID>
3816             <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3817
3818             </SecurityContext>
3819           </wsa:Metadata>
3820         </wsa:EndpointReference>
3821       </AttributeValue>
3822     </Attribute>
3823 </AttributeStatement>
```

3824 **Example 22.** <AttributeStatement> that might be found in a SAML 1.1 AuthnResponse

3825 In all cases, this <AttributeStatement> MUST only carry an ID-WSF EPR for the Liberty Discovery Service. Any
3826 other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

3827 5. ID-WSF 1.x Resource Offering conveyed in an EPR

3828 In order to support the discovery and subsequent invocation of ID-WSF 1.0 and 1.1 services it may be necessary for
3829 the Discovery Service to carry the ID-WSF 1.x Resource Offering information within the ID-WSF EPR.

3830 The process involves taking the fields that would normally be present in the Resource Offering and placing them into
3831 the appropriate fields within the EPR according to the following rules:

3832 The `<SoapAction>` element in `ServiceInstance/Description` is placed into the `<Metadata>` element.

3833 • The `<ResourceID>` element and/or the `<EncryptedResourceID>` element are placed into the `<Metadata>`
3834 element as-is.

3835 • The `<ServiceType>` element in the `<ServiceInstance>` element is placed into the `<Metadata>` element.

3836 • The `<ProviderID>` element in the `<ServiceInstance>` element is placed into the `<Metadata>` element.

3837 • The `<SecurityMechID>` element in `ServiceInstance/Description` is placed into the
3838 `<SecurityContext>` element (and will be combined with other `SecurityMechIDs` based upon whether or not they
3839 share the same endpoint *and* credential (or do not use a credential)).

3840 • The data from the `<Endpoint>` element in `ServiceInstance/Description` is placed into the `<Address>`
3841 element. Note that if there are multiple distinct `<Endpoint>`s they must be placed into different ID-WSF EPRs
3842 rather than being able to be placed into a single EPR like they were in an RO.

3843 • Options are placed into the `<Metadata>` element.

3844 • Abstract is placed into the `<Metadata>` element.

3845 • Credentials, which in the days of the Resource Offering were carried elsewhere in the message and referenced from
3846 the `ServiceInstance/Description` element are now carried directly within the EPR in a `<sec:Token>` ele-
3847 ment in the `<SecurityContext>` element.

3848 In addition, the ID-WSF EPR MUST also include at least one `<sbfl:Framework>` element with the appropriate value
3849 (1.0 or 1.1) in the `version` attribute for the ID-WSF version being used.

3850 As an example, let's start with an example ID-WSF 1.x Resource Offering:

```

3851
3852 <ResourceOffering>
3853   <ResourceID> 123 </ResourceID>
3854   <ServiceInstance>
3855     <ServiceType>urn:liberty:idsis-pp:2003-08</ServiceType>
3856     <ProviderID>http://pp.services.aol.com</ProviderID>
3857     <Description CredentialRef="1">
3858       <SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</SecurityMechID>
3859       <Endpoint>https://ep1.pp.service.aol.com</Endpoint>
3860     </Description>
3861     <Description>
3862       <SecurityMechID>urn:liberty:security:2006-08:Client-TLS:Null</SecurityMechID>
3863       <Endpoint>https://ep1.pp.service.aol.com</Endpoint>
3864     </Description>
3865   </ServiceInstance>
3866 </ResourceOffering>
3867 <Credentials>
3868   <saml1:Assertion AssertionID="1" ...>
3869     Assertion data goes here
3870   </saml1:Assertion>
3871 </Credentials>
3872

```

3873 Translating this using the above rules would result in the following ID-WSF EPR:

```

3874
3875 <wsa:EndpointReference>
3876   <wsa:Address>https://ep1.pp.services.aol.com</wsa:Address>
3877   <wsa:Metadata>
3878     <ds1:ResourceID>123</ds1:ResourceID>
3879     <ds2:ProviderID>http://pp.services.aol.com</ds2:ProviderID>
3880     <ds2:ServiceType>urn:liberty:idsis-pp:2003-08</ds2:ServiceType>
3881     <ds2:Framework Version="1.1" />
3882     <ds2:SecurityContext>
3883       <ds2:SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</ds2:SecurityMechID>
3884       <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3885         <saml1:Assertion AssertionID="1" ... >
3886           ... assertion data goes here ...
3887         </saml1:Assertion>
3888       </sec:Token>
3889     </ds2:SecurityContext>
3890     <ds2:SecurityContext>
3891       <ds2:SecurityMechID>
3892         urn:liberty:security:2006-08:Client-TLS:Null
3893       </ds2:SecurityMechID>
3894     </ds2:SecurityContext>
3895   </wsa:Metadata>
3896 </wsa:EndpointReference>
3897

```

3898 And subsequently, the invocation of the ID-WSF 1.x service would look to be something along the lines of (assuming
3899 that the WSC chose to use the "...:TLS: bearer" Security Mechanism):

```

3900
3901 <?xml version="1.0" encoding="utf-8" ?>
3902 <S:Envelope...
3903   <S:Header>
3904     <sb:Correlation S:mustUnderstand="1"
3905       messageID=uuid:958312848-29348938-232342121
3906       timestamp="2003-06-06T18:29:18Z" />
3907     <wsse:Security>
3908       <saml1:Assertion AssertionID="1" ... >
3909         ... assertion data goes here ...
3910       </saml1:Assertion>
3911     </wsse:Security>
3912   </S:Header>
3913   <S:Body>
3914     <pp:Query>
3915       <pp:ResourceID>123</pp:ResourceID>
3916       <pp:QueryItem>
3917         ... query data goes here ...
3918       </pp:QueryItem>
3919     </pp:Query>
3920   </S:Body>
3921 </S:Envelope>
3922

```

3923 **6. Acknowledgments**

3924 Many people have made contributions to this specification as it has evolved over time. The original specification was
3925 written by John Beatty with subsequent versions "inked" by Jonathan Sergent and later, Jeff Hodges and now myself.

3926 The changes made in this latest release of the specification are due in a large part to the work of Jeff Hodges, Robert
3927 Aarts, John Kemp, Gary Ellison and Greg Whitehead. Many others, including those that are listed as contributors on
3928 the cover page, have also played a part in this and earlier releases of the specification. Many thanks to all who partici-
3929 pated (and apologies if I have forgotten to mention your name).

3930 References

3931 Normative

- 3932 [LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version v2.0, Liberty Alliance Project (30 July,
3933 2006). <http://www.projectliberty.org/specs>
- 3934 [LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version 2.0-errata-v1.0,
3935 Liberty Alliance Project (21 April, 2007). <http://www.projectliberty.org/specs>
- 3936 [LibertySecMech11] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance
3937 Project (18 April 2004). <http://www.projectliberty.org/specs/>
- 3938 [LibertySecMech20SAML] Hirsch, Frederick, eds. "ID-WSF 2.0 SecMech SAML Profile," Version 2.0-errata-v1.0,
3939 Liberty Alliance Project (08 November, 2006). <http://www.projectliberty.org/specs>
- 3940 [LibertyAuthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. "Liberty ID-WSF Authentication, Single
3941 Sign-On, and Identity Mapping Services Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (28
3942 November, 2006). <http://www.projectliberty.org/specs>
- 3943 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty ID-
3944 WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007). [http://](http://www.projectliberty.org/specs)
3945 www.projectliberty.org/specs
- 3946 [LibertyIDWSF20SCR] Whitehead, Greg, eds. Version 1.0 errata v1.0, Liberty Alliance Project (21 April, 2007).
3947 <http://www.projectliberty.org/specs>
- 3948 [LibertyIDWSFv20Errata] Champagne, Darryl, Lockhart, Rob, Tiffany, Eric, eds. "Liberty ID-WSF 2.0 Errata," Ver-
3949 sion 1.0, Liberty Alliance Project (13 April, 2007). <http://www.projectliberty.org/specs>
- 3950 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engi-
3951 neering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 3952 [RFC3986] Berners-Lee, T., Fielding, R., Masinter, L., eds. (January 2005). "Uniform Resource Identifier (URI):
3953 Generic Syntax," RFC 3986 (Obsoletes RFC2732, RFC2396, RFC1808) (Updates RFC1738) (Also STD0066)
3954 (Status: STANDARD), The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc3986.txt>
- 3955 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions and Protocol
3956 for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organ-
3957 ization for the Advancement of Structured Information Standards [http://docs.oasis-open.org/security/saml/](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
3958 [v2.0/saml-core-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 3959 [SAMLMeta2] Cantor, Scott, Moreh, Jahan, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Metadata for the OASIS
3960 Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for the
3961 Advancement of Structured Information Standards [http://docs.oasis-open.org/security/saml/v2.0/saml-met-](http://docs.oasis-open.org/security/saml/v2.0/saml-met-adata-2.0-os.pdf)
3962 [adata-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-met-adata-2.0-os.pdf)
- 3963 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October 2004).
3964 "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium [http://](http://www.w3.org/TR/xmlschema-1/)
3965 www.w3.org/TR/xmlschema-1/
- 3966 [WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds.
3967 World Wide Web Consortium W3C Recommendation (9 May 2006). [http://www.w3.org/TR/2006/REC-ws-](http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/)
3968 [addr-core-20060509/](http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/)

- 3969 [WSAv1.0-SOAP] "WS-Addressing 1.0 SOAP Binding," Gudgin, Martin, Hadley, Marc, eds. World Wide Web Consortium W3C Recommendation (9 May 2006). <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 3970
- 3971 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith, Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001). <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 3972
- 3973
- 3974 [wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the Advancement of Structured Information Standards <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
- 3975
- 3976
- 3977
- 3978 [wss-sms-draft] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 30, 2003). "Web Services Security: SOAP Message Security," Draft WSS-SOAPMessageSecurity-14-06-2003, Organization for the Advancement of Structured Information Standards <http://www.oasis-open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf>
- 3979
- 3980
- 3981
- 3982 [xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (10 December 2002). "XML Encryption Syntax and Processing," W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlenc-core/>
- 3983

3984 Informative

- 3985 [LibertyPAOS] Aarts, Robert, Kemp, John, eds. "Liberty Reverse HTTP Binding for SOAP Specification," Version 2.0, Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 3986
- 3987 [LibertyClientProfiles] Aarts, Robert, Kainulainen, Jukka, Kemp, John, eds. "Liberty ID-WSF Profiles for Liberty-Enabled User Agents and Devices," Version 2.0-errata-v1.0, Liberty Alliance Project (22 January, 2007). <http://www.projectliberty.org/specs>
- 3988
- 3989

3990 **A. Discovery Service Version 2.0 XSD**

```

3991
3992 <?xml version="1.0" encoding="UTF-8"?>
3993 <xs:schema targetNamespace="urn:liberty:disco:2006-08"
3994     xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
3995     xmlns:sb="urn:liberty:sb:2006-08"
3996     xmlns:sbf="urn:liberty:sb"
3997     xmlns:sec="urn:liberty:security:2006-08"
3998     xmlns:lu="urn:liberty:util:2006-08"
3999     xmlns:wsa="http://www.w3.org/2005/08/addressing"
4000     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
4001     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
4002     xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
4003     xmlns:xs="http://www.w3.org/2001/XMLSchema"
4004
4005     xmlns="urn:liberty:disco:2006-08"
4006     elementFormDefault="qualified"
4007     attributeFormDefault="unqualified"
4008 >
4009
4010 <xs:import namespace="urn:liberty:util:2006-08"
4011     schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
4012
4013 <xs:import namespace="urn:liberty:sb:2006-08"
4014     schemaLocation="liberty-idwsf-soap-binding-v2.0.xsd"/>
4015
4016 <xs:import namespace="urn:liberty:sb"
4017     schemaLocation="liberty-idwsf-soap-binding.xsd"/>
4018
4019 <xs:import namespace="http://www.w3.org/2005/08/addressing"
4020     schemaLocation="ws-addr-1.0.xsd"/>
4021
4022 <xs:import namespace="urn:oasis:names:tc:SAML:2.0:metadata"
4023     schemaLocation="saml-schema-metadata-2.0.xsd"/>
4024
4025 <xs:import namespace="urn:liberty:security:2006-08"
4026     schemaLocation="liberty-idwsf-security-mechanisms-v2.0.xsd"/>
4027
4028 <xs:import
4029     namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
4030     schemaLocation="wss-secext-1.0.xsd"/>
4031
4032 <xs:import
4033     namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
4034     schemaLocation="wss-util-1.0.xsd"/>
4035
4036 <!-- **** Discovery Service Data Elements & Types **** -->
4037
4038 <!-- The data elements and types in this section are used to
4039     embellish WS-Addressing Endpoint References (EPRs).
4040     They are placed in the /wsa:EndpointReference/Metadata
4041     element. Specific usage and cardinalities are stipulated
4042     in the Discovery Service v2.0 Specification. -->
4043
4044 <!-- Abstract: natural-language description of service -->
4045
4046 <xs:element name="Abstract" type="xs:string"/>
4047
4048 <!-- Provider ID -->
4049
4050 <xs:element name="ProviderID" type="xs:anyURI"/>
4051
4052 <!-- Service Type -->
4053
4054 <xs:element name="ServiceType" type="xs:anyURI"/>
4055

```

```

4056 <!-- Framework Description -->
4057
4058 <xs:element name="Framework" type="sbf:FrameworkType" />
4059
4060 <!-- EPR Expiration Timestamp -->
4061
4062 <xs:attribute name="notOnOrAfter" type="xs:dateTime"/>
4063
4064 <!-- Security Context Container -->
4065
4066 <xs:element name="SecurityContext">
4067   <xs:complexType>
4068     <xs:sequence>
4069       <xs:element ref="SecurityMechID"
4070         minOccurs="1"
4071         maxOccurs="unbounded"/>
4072
4073       <xs:element ref="sec:Token"
4074         minOccurs="0"
4075         maxOccurs="unbounded"/>
4076     </xs:sequence>
4077   </xs:complexType>
4078 </xs:element>
4079
4080 <!-- Security Mechanism ID -->
4081
4082 <xs:element name="SecurityMechID" type="xs:anyURI"/>
4083
4084 <!-- Options -->
4085
4086 <xs:element name="Options" type="OptionsType"/>
4087
4088 <xs:element name="Option" type="xs:anyURI" />
4089
4090 <xs:complexType name="OptionsType">
4091   <xs:sequence>
4092     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/>
4093   </xs:sequence>
4094 </xs:complexType>
4095
4096 <!-- Address -->
4097
4098 <xs:element name="Address" type="xs:anyURI"/>
4099
4100 <!-- Action(s) - the interfaces available at this service -->
4101
4102 <xs:element name="Action" type="xs:anyURI" />
4103 <!-- Keys Element - For use in ModifyResponse -->
4104
4105 <xs:element name="Keys" type="KeysType"/>
4106
4107 <xs:complexType name="KeysType">
4108   <xs:sequence>
4109     <xs:element ref="md:KeyDescriptor"
4110       minOccurs="1"
4111       maxOccurs="unbounded"/>
4112   </xs:sequence>
4113 </xs:complexType>
4114
4115 <!-- Service Metadata (SvcMD) - metadata about service instance -->
4116
4117 <xs:element name="SvcMD" type="SvcMetadataType"/>
4118 <xs:complexType name="SvcMetadataType">
4119   <xs:sequence>
4120     <xs:element ref="Abstract" />
4121     <xs:element ref="ProviderID" />
4122     <xs:element ref="ServiceContext" maxOccurs="unbounded" />

```

```

4123     </xs:sequence>
4124     <xs:attribute name="svcMDID" type="xs:string" use="optional" />
4125 </xs:complexType>
4126
4127 <!-- ServiceContext - describes service type/option/endpoint context -->
4128 <xs:element name="ServiceContext" type="ServiceContextType"/>
4129 <xs:complexType name="ServiceContextType">
4130   <xs:sequence>
4131     <xs:element ref="ServiceType"      maxOccurs="unbounded" />
4132     <xs:element ref="Options"         minOccurs="0"
4133                                     maxOccurs="unbounded" />
4134     <xs:element ref="EndpointContext" maxOccurs="unbounded" />
4135   </xs:sequence>
4136 </xs:complexType>
4137
4138 <!-- EndpointContext - describes endpoints used to access service -->
4139 <xs:element name="EndpointContext" type="EndpointContextType" />
4140 <xs:complexType name="EndpointContextType">
4141   <xs:sequence>
4142     <xs:element ref="Address"         maxOccurs="unbounded" />
4143     <xs:element ref="sbf:Framework"  maxOccurs="unbounded" />
4144     <xs:element ref="SecurityMechID" maxOccurs="unbounded" />
4145     <xs:element ref="Action"         minOccurs="0"
4146                                     maxOccurs="unbounded" />
4147   </xs:sequence>
4148 </xs:complexType>
4149
4150 <!-- SvcMD ID element used to refer to Service Metadata elements -->
4151 <xs:element name="SvcMDID" type="xs:string" />
4152
4153 <!-- **** Discovery Service Protocol Messages Elements & Types **** -->
4154
4155 <!-- Query Message Element & Type -->
4156
4157 <xs:element name="Query" type="QueryType"/>
4158
4159 <xs:complexType name="QueryType">
4160   <xs:sequence>
4161     <xs:element name="RequestedService"
4162                 type="RequestedServiceType"
4163                 minOccurs="0"
4164                 maxOccurs="unbounded" />
4165   </xs:sequence>
4166
4167   <xs:anyAttribute namespace="##other" processContents="lax" />
4168 </xs:complexType>
4169
4170 <xs:complexType name="RequestedServiceType">
4171   <xs:sequence>
4172     <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
4173
4174     <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
4175
4176     <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
4177
4178     <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
4179
4180     <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
4181
4182     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
4183
4184     <xs:any namespace="##other"
4185             processContents="lax"
4186             minOccurs="0"
4187             maxOccurs="unbounded" />
4188   </xs:sequence>
4189 </xs:complexType>

```

```

4190
4191     <xs:attribute name="reqID" type="xs:string" use="optional" />
4192     <xs:attribute name="resultsType" type="xs:string" use="optional" />
4193
4194 </xs:complexType>
4195
4196 <!-- QueryResponse Message Element & Type -->
4197
4198 <xs:element name="QueryResponse" type="QueryResponseType"/>
4199
4200 <xs:complexType name="QueryResponseType">
4201     <xs:sequence>
4202         <xs:element ref="lu:Status"/>
4203
4204         <xs:element ref="wsa:EndpointReference"
4205             minOccurs="0"
4206             maxOccurs="unbounded"/>
4207     </xs:sequence>
4208     <xs:anyAttribute namespace="##other" processContents="lax"/>
4209 </xs:complexType>
4210
4211
4212 <!-- -->
4213 <!-- DS Interfaces for SvcMD Associations -->
4214 <!-- -->
4215 <!-- These interfaces support the adding, deleting, -->
4216 <!-- querying SvcMD Associations for a principal. -->
4217 <!-- -->
4218
4219 <!-- SvcMDAssociationAdd operation -->
4220
4221 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType"/>
4222
4223 <xs:complexType name="SvcMDAssociationAddType">
4224     <xs:sequence>
4225         <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4226     </xs:sequence>
4227     <xs:anyAttribute namespace="##other" processContents="lax"/>
4228 </xs:complexType>
4229
4230 <!-- Response for SvcMDAssociationAdd operation -->
4231
4232 <xs:element name="SvcMDAssociationAddResponse"
4233     type="SvcMDAssociationAddResponseType"/>
4234
4235 <xs:complexType name="SvcMDAssociationAddResponseType">
4236     <xs:sequence>
4237         <xs:element ref="lu:Status" />
4238     </xs:sequence>
4239     <xs:anyAttribute namespace="##other" processContents="lax"/>
4240 </xs:complexType>
4241 <!-- SvcMDAssociationDelete operation -->
4242
4243 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType"/>
4244
4245 <xs:complexType name="SvcMDAssociationDeleteType">
4246     <xs:sequence>
4247         <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4248     </xs:sequence>
4249     <xs:anyAttribute namespace="##other" processContents="lax"/>
4250 </xs:complexType>
4251 <!-- Response for SvcMDAssociationDelete operation -->
4252
4253 <xs:element name="SvcMDAssociationDeleteResponse"
4254     type="SvcMDAssociationDeleteResponseType"/>
4255
4256 <xs:complexType name="SvcMDAssociationDeleteResponseType">

```

```

4257     <xs:sequence>
4258         <xs:element ref="lu:Status" />
4259     </xs:sequence>
4260     <xs:anyAttribute namespace="##other" processContents="lax"/>
4261 </xs:complexType>
4262 <!-- SvcMDAssociationQuery operation -->
4263
4264 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType"/>
4265
4266 <xs:complexType name="SvcMDAssociationQueryType">
4267     <xs:sequence>
4268         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4269     </xs:sequence>
4270     <xs:anyAttribute namespace="##other" processContents="lax"/>
4271 </xs:complexType>
4272 <!-- Response for SvcMDAssociationQuery operation -->
4273
4274 <xs:element name="SvcMDAssociationQueryResponse"
4275     type="SvcMDAssociationQueryResponseType"/>
4276
4277 <xs:complexType name="SvcMDAssociationQueryResponseType">
4278     <xs:sequence>
4279         <xs:element ref="lu:Status" />
4280         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4281     </xs:sequence>
4282     <xs:anyAttribute namespace="##other" processContents="lax"/>
4283 </xs:complexType>
4284
4285 <!-- -->
4286 <!-- DS Interfaces for Service Metadata Management -->
4287 <!-- -->
4288 <!-- These interfaces document a create, replace, -->
4289 <!-- delete, and query interface for the service -->
4290 <!-- metadata which is later associated with a -->
4291 <!-- principal. -->
4292 <!-- -->
4293
4294 <!-- Register operation for Service Metadata -->
4295
4296 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
4297
4298 <xs:complexType name="SvcMDRegisterType">
4299     <xs:sequence>
4300         <xs:element ref="SvcMD" maxOccurs="unbounded" />
4301     </xs:sequence>
4302     <xs:anyAttribute namespace="##other" processContents="lax"/>
4303 </xs:complexType>
4304
4305 <!-- Response for SvcMDRegister operation -->
4306
4307 <xs:element name="SvcMDRegisterResponse"
4308     type="SvcMDRegisterResponseType"/>
4309
4310 <xs:complexType name="SvcMDRegisterResponseType">
4311     <xs:sequence>
4312
4313         <xs:element ref="lu:Status" />
4314         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4315         <xs:element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
4316
4317     </xs:sequence>
4318     <xs:anyAttribute namespace="##other" processContents="lax"/>
4319 </xs:complexType>
4320
4321 <!-- Delete operation on Service Metadata -->
4322
4323 <xs:element name="SvcMDDelete" type="SvcMDDeleteType"/>

```

```
4324
4325 <xs:complexType name="SvcMDDeleteType">
4326   <xs:sequence>
4327     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4328   </xs:sequence>
4329   <xs:anyAttribute namespace="##other" processContents="lax" />
4330 </xs:complexType>
4331
4332 <!-- Response for delete operation on Service Metadata -->
4333
4334 <xs:element name="SvcMDDeleteResponse" type="SvcMDDeleteResponseType"/>
4335
4336 <xs:complexType name="SvcMDDeleteResponseType">
4337   <xs:sequence>
4338     <xs:element ref="lu:Status" />
4339   </xs:sequence>
4340   <xs:anyAttribute namespace="##other" processContents="lax" />
4341 </xs:complexType>
4342
4343 <!-- Query operation on Service Metadata -->
4344
4345 <xs:element name="SvcMDQuery" type="SvcMDQueryType"/>
4346
4347 <xs:complexType name="SvcMDQueryType">
4348   <xs:sequence>
4349     <xs:element ref="SvcMDID"
4350       minOccurs="0"
4351       maxOccurs="unbounded" />
4352   </xs:sequence>
4353   <xs:anyAttribute namespace="##other" processContents="lax" />
4354 </xs:complexType>
4355
4356 <!-- Response for Query operation on Service Metadata -->
4357
4358 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType"/>
4359
4360 <xs:complexType name="SvcMDQueryResponseType">
4361   <xs:sequence>
4362     <xs:element ref="lu:Status" />
4363     <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
4364   </xs:sequence>
4365   <xs:anyAttribute namespace="##other" processContents="lax" />
4366 </xs:complexType>
4367
4368 <!-- Replace operation on Service Metadata -->
4369
4370 <xs:element name="SvcMDReplace" type="SvcMDReplaceType"/>
4371
4372 <xs:complexType name="SvcMDReplaceType">
4373   <xs:sequence>
4374     <xs:element ref="SvcMD" maxOccurs="unbounded" />
4375   </xs:sequence>
4376   <xs:anyAttribute namespace="##other" processContents="lax" />
4377 </xs:complexType>
4378
4379 <!-- Response for SvcMDReplace operation -->
4380
4381 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType"/>
4382
4383 <xs:complexType name="SvcMDReplaceResponseType">
4384   <xs:sequence>
4385     <xs:element ref="lu:Status" />
4386   </xs:sequence>
4387   <xs:anyAttribute namespace="##other" processContents="lax" />
4388 </xs:complexType>
4389
4390 </xs:schema>
```

4391
4392

4393 **B. Discovery Service WSDL**

```

4394
4395 <?xml version="1.0"?>
4396 <definitions name="disco-svc"
4397   targetNamespace="urn:liberty:disco:2006-08"
4398   xmlns:tns="urn:liberty:disco:2006-08"
4399   xmlns="http://schemas.xmlsoap.org/wsdl/"
4400   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4401   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4402   xmlns:sb="urn:liberty:sb:2006-08"
4403   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
4404   xmlns:disco="urn:liberty:disco:2006-08"
4405   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4406   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
4407     http://schemas.xmlsoap.org/wsdl/
4408     http://www.w3.org/2006/02/addressing/wsdl
4409     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
4410
4411   <!-- Abstract WSDL for Liberty Discovery Service v2.0 Specification -->
4412
4413   <xsd:documentation>
4414
4415     XML Schema from Liberty Discovery Service Specification.
4416
4417     ### NOTICE ###
4418
4419     Copyright (c) 2004-2006 Liberty Alliance participants, see
4420     http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
4421
4422   </xsd:documentation>
4423
4424   <types>
4425     <xsd:schema>
4426       <xsd:import namespace="urn:liberty:disco:2006-08"
4427         schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
4428     </xsd:schema>
4429   </types>
4430
4431   <message name="Query">
4432     <part name="body" element="disco:Query"/>
4433   </message>
4434   <message name="QueryResponse">
4435     <part name="body" element="disco:QueryResponse"/>
4436   </message>
4437
4438   <message name="SvcMDAssociationAdd">
4439     <part name="body" element="disco:SvcMDAssociationAdd"/>
4440   </message>
4441   <message name="SvcMDAssociationAddResponse">
4442     <part name="body" element="disco:SvcMDAssociationAddResponse"/>
4443   </message>
4444
4445   <message name="SvcMDAssociationQuery">
4446     <part name="body" element="disco:SvcMDAssociationQuery"/>
4447   </message>
4448   <message name="SvcMDAssociationQueryResponse">
4449     <part name="body" element="disco:SvcMDAssociationQueryResponse"/>
4450   </message>
4451
4452   <message name="SvcMDAssociationDelete">
4453     <part name="body" element="disco:SvcMDAssociationDelete"/>
4454   </message>
4455   <message name="SvcMDAssociationDeleteResponse">
4456     <part name="body" element="disco:SvcMDAssociationDeleteResponse"/>
4457   </message>
4458

```

```
4459 <message name="SvcMDRegister">
4460   <part name="body" element="disco: SvcMDRegister" />
4461 </message>
4462 <message name="SvcMDRegisterResponse">
4463   <part name="body" element="disco: SvcMDRegisterResponse" />
4464 </message>
4465
4466 <message name="SvcMDQuery">
4467   <part name="body" element="disco: SvcMDQuery" />
4468 </message>
4469 <message name="SvcMDQueryResponse">
4470   <part name="body" element="disco: SvcMDQueryResponse" />
4471 </message>
4472
4473 <message name="SvcMDReplace">
4474   <part name="body" element="disco: SvcMDReplace" />
4475 </message>
4476 <message name="SvcMDReplaceResponse">
4477   <part name="body" element="disco: SvcMDReplaceResponse" />
4478 </message>
4479
4480 <message name="SvcMDDelete">
4481   <part name="body" element="disco: SvcMDDelete" />
4482 </message>
4483 <message name="SvcMDDeleteResponse">
4484   <part name="body" element="disco: SvcMDDeleteResponse" />
4485 </message>
4486
4487
4488 <portType name="DiscoveryPort">
4489
4490   <operation name="DiscoveryQuery">
4491     <input message="tns: Query"
4492       wsaw: Action="urn: liberty: disco: 2006-08: Query" />
4493     <output message="tns: QueryResponse"
4494       wsaw: Action="urn: liberty: disco: 2006-08: QueryResponse" />
4495   </operation>
4496
4497   <operation name="MDAssociationAdd">
4498     <input message="tns: SvcMDAssociationAdd"
4499       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationAdd" />
4500     <output message="tns: SvcMDAssociationAddResponse"
4501       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationAddResponse" />
4502   </operation>
4503
4504   <operation name="MDAssociationQuery">
4505     <input message="tns: SvcMDAssociationQuery"
4506       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationQuery" />
4507     <output message="tns: SvcMDAssociationQueryResponse"
4508       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationQueryResponse" />
4509   </operation>
4510
4511   <operation name="MDAssociationDelete">
4512     <input message="tns: SvcMDAssociationDelete"
4513       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationDelete" />
4514     <output message="tns: SvcMDAssociationDeleteResponse"
4515       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDAssociationDeleteResponse" />
4516   </operation>
4517
4518   <operation name="MetadataRegister">
4519     <input message="tns: SvcMDRegister"
4520       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDRegister" />
4521     <output message="tns: SvcMDRegisterResponse"
4522       wsaw: Action="urn: liberty: disco: 2006-08: SvcMDRegisterResponse" />
4523   </operation>
4524
4525   <operation name="MetadataQuery">
```

Liberty ID-WSF Discovery Service Specification

```

4526     <input message="tns:SvcMDQuery"
4527         wsaw:Action="urn:liberty:disco:2006-08:SvcMDQuery" />
4528     <output message="tns:SvcMDQueryResponse"
4529         wsaw:Action="urn:liberty:disco:2006-08:SvcMDQueryResponse" />
4530 </operation>
4531
4532 <operation name="MetadataReplace">
4533     <input message="tns:SvcMDReplace"
4534         wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplace" />
4535     <output message="tns:SvcMDReplaceResponse"
4536         wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplaceResponse" />
4537 </operation>
4538
4539 <operation name="MetadataDelete">
4540     <input message="tns:SvcMDDelete"
4541         wsaw:Action="urn:liberty:disco:2006-08:SvcMDDelete" />
4542     <output message="tns:SvcMDDeleteResponse"
4543         wsaw:Action="urn:liberty:disco:2006-08:SvcMDDeleteResponse" />
4544 </operation>
4545
4546
4547 </portType>
4548
4549 <!--
4550 An example of a binding and service that can be used with this
4551 abstract service description is provided below.
4552 -->
4553
4554 <binding name="DiscoveryBinding" type="tns:DiscoveryPort">
4555
4556     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
4557
4558     <operation name="DiscoveryQuery">
4559         <soap:operation soapAction="urn:liberty:disco:2006-08:Query" />
4560         <input> <soap:body use="literal"/> </input>
4561         <output> <soap:body use="literal"/> </output>
4562     </operation>
4563
4564     <operation name="MDAssociationAdd">
4565         <soap:operation
4566         soapAction="urn:liberty:disco:2006-08:SvcMDAssociationAdd" />
4567         <input> <soap:body use="literal"/> </input>
4568         <output> <soap:body use="literal"/> </output>
4569     </operation>
4570
4571
4572     <operation name="MDAssociationQuery">
4573         <soap:operation
4574         soapAction="urn:liberty:disco:2006-08:SvcMDAssociationQuery" />
4575         <input> <soap:body use="literal"/> </input>
4576         <output> <soap:body use="literal"/> </output>
4577     </operation>
4578
4579     <operation name="MDAssociationDelete">
4580         <soap:operation
4581         soapAction="urn:liberty:disco:2006-08:SvcMDAssociationDelete" />
4582         <input> <soap:body use="literal"/> </input>
4583         <output> <soap:body use="literal"/> </output>
4584     </operation>
4585
4586     <operation name="MetadataRegister">
4587         <soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDRegister" />
4588         <input> <soap:body use="literal"/> </input>
4589         <output> <soap:body use="literal"/> </output>
4590     </operation>
4591
4592     <operation name="MetadataQuery">

```

Liberty ID-WSF Discovery Service Specification

```
4593     <del>soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDQuery" />del
4594     <input> <soap:body use="literal"/> </input>
4595     <output> <soap:body use="literal"/> </output>
4596   </operation>
4597
4598   <operation name="MetadataReplace">
4599     <del>soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDReplace" />del
4600     <input> <soap:body use="literal"/> </input>
4601     <output> <soap:body use="literal"/> </output>
4602   </operation>
4603
4604   <operation name="MetadataDelete">
4605     <del>soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDDelete" />del
4606     <input> <soap:body use="literal"/> </input>
4607     <output> <soap:body use="literal"/> </output>
4608   </operation>
4609
4610 </binding>
4611
4612 <service name="DiscoveryService">
4613
4614   <port name="DiscoveryPort" binding="tns:DiscoveryBinding">
4615     <!-- Modify with the REAL SOAP endpoint -->
4616
4617     <soap:address location="http://example.com/discovery"/>
4618
4619   </port>
4620 </service>
4621
4622 </definitions>
4623
4624
4625
4626
4627
```