



Liberty ID-WSF Discovery Service Specification

Version: 2.0

Editors:

Jeff Hodges, NeuStar, Inc.

Conor Cahill, Intel Corporation

Contributors:

John Beatty, Sun Microsystems, Inc.

Jonathan Sergent, Sun Microsystems, Inc.

Robert Aarts, Nokia Corporation

Carolina Canales-Valenzuela, Ericsson

Darryl Champagne, IEEE-ISTO

Gary Ellison, Sun Microsystems, Inc.

Jukka Kainulainen, Nokia Corporation

John Kemp, Nokia Corporation

Paul Madsen, NTT, formerly Entrust, 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-v2.0.pdf

1 **Notice**

2 This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the
3 document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works
4 of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact
5 the Liberty 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
7 rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are
8 not and shall not be held responsible in any manner for identifying or failing to identify any or all such third party
9 intellectual property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance**
10 **makes any warranty of any kind, express or implied, including any implied warranties of merchantability,**
11 **non-infringement of third party intellectual property rights, and fitness for a particular purpose.** Implementers
12 of this Specification are advised to review the Liberty Alliance Project's website (<http://www.projectliberty.org/>) for
13 information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance
14 Management Board.

15 Copyright © 2006 Adobe Systems; America Online, Inc.; American Express Company; Amsoft Systems Pvt Ltd.;
16 Avatier Corporation; Axalto; Bank of America Corporation; BIPAC; BMC Software, Inc.; Computer Associates
17 International, Inc.; DataPower Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.;
18 Ericsson; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le
19 développement de l'administration électronique (ADAE); Gamefederation; Gemplus; General Motors; Giesecke &
20 Devrient GmbH; GSA Office of Governmentwide Policy; Hewlett-Packard Company; IBM Corporation; Intel
21 Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard International; Mobile Telephone Networks (Pty)
22 Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon Telegraph and Telephone Corporation; Nokia
23 Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation;
24 Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
25 Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
26 Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Trusted Network Technologies; UTI; VeriSign, Inc.; Vodafone
27 Group Plc.; Wave Systems Corp. All rights reserved.

28 Liberty Alliance Project
29 Licensing Administrator
30 c/o IEEE-ISTO
31 445 Hoes Lane
32 Piscataway, NJ 08855-1331, USA
33 info@projectliberty.org

34 Contents

35	1. Introduction	6
36	1.1. Conceptual Model and Terminology	6
37	1.2. Scope	7
38	1.3. Notation and Conventions	7
39	1.3.1. XML Namespaces	7
40	2. Discovery Service Information Model	9
41	2.1. Overview of Discovery Service Information Model	9
42	2.2. Versioning in ID-WSF	10
43	2.3. ID-WSF Endpoint References (EPRs)	10
44	2.3.1. EPR Profiling Attributes	11
45	2.3.2. EPR Profiling Elements	12
46	2.3.3. ID-WSF Web Services Addressing EPR Profile	16
47	2.3.4. Effective Web Services Addressing EPR	20
48	2.3.5. Example Liberty ID-WSF EPRs	21
49	2.4. Service Metadata	23
50	2.4.1. Service Metadata element	23
51	2.4.2. Minting ID-WSF EPRs based upon Service Metadata	26
52	2.4.3. Service Metadata Example	26
53	3. Discovery Service	30
54	3.1. Service URIs	30
55	3.2. Status Codes	31
56	3.3. Operation: <i>DiscoveryQuery</i>	32
57	3.3.1. wsa:Action values for <i>DiscoveryQuery</i> Messages	32
58	3.3.2. <Query> Message	32
59	3.3.3. QueryResponse	36
60	3.3.4. DiscoveryQuery Processing Rules	38
61	3.4. Operation: <i>MDAssociationAdd</i>	38
62	3.4.1. wsa:Action values for <i>MDAssociationAdd</i> Messages	38
63	3.4.2. SvcMDAssociationAdd Message	39
64	3.4.3. SvcMDAssociationAddResponse Message	39
65	3.4.4. MDAssociation Add Processing Rules	40
66	3.5. Operation: <i>MDAssociationQuery</i>	40
67	3.5.1. wsa:Action values for <i>MDAssociationQuery</i> Messages	41
68	3.5.2. SvcMDAssociationQuery Message	41
69	3.5.3. SvcMDAssociationQueryResponse Message	41
70	3.5.4. MDAssociation Query Processing Rules	42
71	3.6. Operation: <i>MDAssociationDelete</i>	42
72	3.6.1. wsa:Action values for <i>MDAssociationDelete</i> Messages	42
73	3.6.2. SvcMDAssociationDelete Message	42
74	3.6.3. SvcMDAssociationDeleteResponse Message	43
75	3.6.4. MDAssociation Delete Processing Rules	43
76	3.7. Operation: <i>MetadataRegister</i>	44
77	3.7.1. wsa:Action values for <i>MetadataRegister</i> Messages	44
78	3.7.2. SvcMDRegister Message	44
79	3.7.3. SvcMDRegisterResponse Message	45
80	3.7.4. Metadata Register Processing Rules	46
81	3.8. Operation: <i>MetadataQuery</i>	46
82	3.8.1. wsa:Action values for <i>MetadataQuery</i> Messages	47
83	3.8.2. SvcMDQuery Message	47
84	3.8.3. SvcMDQueryResponse Message	47
85	3.8.4. Metadata Query Processing Rules	48
86	3.9. Operation: <i>MetadataReplace</i>	48

87	3.9.1. wsa:Action values for MetadataReplace Messages	49
88	3.9.2. SvcMDReplace Message	49
89	3.9.3. SvcMDReplaceResponse Message	49
90	3.9.4. Metadata Replace Processing Rules	50
91	3.10. Operation: <i>MetadataDelete</i>	51
92	3.10.1. wsa:Action values for MetadataDelete Messages	51
93	3.10.2. SvcMDDelete Message	51
94	3.10.3. SvcMDDeleteResponse Message	51
95	3.10.4. Metadata Delete Processing Rules	52
96	3.11. Option Value for Response Authentication	53
97	3.12. Including Keys in the ModifyResponse Message	53
98	3.13. Discovery Service Example Messages (NON-Normative)	54
99	3.13.1. Query People Service	54
100	3.13.2. Query provider	56
101	3.13.3. Query (empty)	57
102	3.13.4. Query People Service and Provider	59
103	3.13.5. SvcMDQuery (empty)	60
104	3.13.6. SvcMDQuery w/Bad SvcMDID	60
105	3.13.7. SvcMDRegister w/single SvcMD	61
106	3.13.8. SvcMDQuery w/Good SvcMDID	61
107	3.13.9. SvcMDDelete w/Good SvcMDID	62
108	3.13.10. SvcMDDelete w/Already Deleted SvcMDID	62
109	3.13.11. SvcMDRegister w/Complex SvcMD	63
110	3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD	64
111	3.13.13. SvcMDReplace of existing SvcMD	65
112	3.13.14. SvcMDQuery of Replaced SvcMDID	66
113	3.13.15. SvcMDDelete of replaced SvcMD	67
114	3.13.16. SvcMDRegister w/multiple SvcMDs	67
115	3.13.17. Query Calendar Service	69
116	3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD	69
117	3.13.19. SvcMDAssociationQuery w/SvcMDID	70
118	3.13.20. SvcMDAssociationQuery w/o SvcMDID	70
119	3.13.21. Query Calendar Service (again)	70
120	3.13.22. Query Calendar Service w/Action	71
121	3.13.23. Query Calendar Service w/resultsType=all	72
122	3.13.24. Query for all Calendar Service EPRs	74
123	3.13.25. Query for one Calendar Service EPRs	75
124	3.13.26. Query specific version of Calendar Service	76
125	3.13.27. SvcMDReplace Calendar Service	78
126	3.13.28. Query old Calendar Service	78
127	3.13.29. Query for all Calendar Service EPRs	79
128	3.13.30. SvcMDAssociationAdd the ATM Service SvcMD	80
129	3.13.31. Query ATM Service w/resultsType=best	80
130	3.13.32. Query ATM Service w/Withdraw Action	82
131	3.13.33. Query ATM Service w/Option	83
132	3.13.34. Query ATM Service w/unknown Option	84
133	3.13.35. Query ATM Service w/good and bad Option	85
134	4. Discovery Service ID-WSF EPR conveyed via a Security Token	87
135	4.1. EPR Generation Rules	87
136	4.2. SAML 2.0 Security Tokens	87
137	4.3. SAML 1.x (Liberty ID-FF) Security Tokens	88
138	5. ID-WSF 1.x Resource Offering conveyed in an EPR	90
139	6. Acknowledgments	92
140	References	93

141	A. Discovery Service Version 2.0 XSD	95
142	B. Discovery Service WSDL	102

143 1. Introduction

144 This specification defines a framework for describing and discovering web services in general and *identity web services*
145 in particular. The conceptual model and terminology is first provided to set the context for the rest of the specification.
146 Next, the data types for the information maintained by a Discovery Service are specified. Then the Discovery Service
147 itself is specified.

148 1.1. Conceptual Model and Terminology

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

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

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

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

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

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

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

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

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

182 A resource commonly has access control policies associated with it. These access control policies are typically under
183 the purview of the entity or entities associated with the resource (in common language, the entity or entities could be
184 said to "own", or "manage", the resource). The access control policies associated with a resource must be enforced by
185 the service instance.

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

189 1.2. Scope

190 This specification:

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

197 1.3. Notation and Conventions

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

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

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

205 1.3.1. XML Namespaces

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

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

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

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

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

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

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

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

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

- 217 • The prefix *sbf*: stands for the Liberty Soap Bindings Framework namespace [[LibertySOAPBinding](#)]:
218 *urn:liberty:sb*
- 219 • The prefix *sec*: stands for the Liberty Security Mechanisms namespace [[LibertySecMech](#)]:
220 *urn:liberty:security:2006-08*
- 221 • The prefix *wsa*: stands for the W3C Web Services Addressing (WSA) namespace [[WSAv1.0](#)]:
222 *http://www.w3.org/@@@/@/@/addressing*
- 223 • The prefix *wSDL*: stands for the primary WSDL v1.1 namespace [[WSDLv1.1](#)]:
224 *http://schemas.xmlsoap.org/wSDL/*
- 225 • The prefix *wSDLsoap*: stands for the namespace of the WSDL-SOAP binding [[WSDLv1.1](#)]:
226 *http://schemas.xmlsoap.org/wSDL/soap/*
- 227 • The prefix *xs*: stands for the W3C XML schema namespace [[Schema1-2](#)]:
228 *http://www.w3.org/2001/XMLSchema*
- 229 • The prefix *xsi*: stands for the W3C XML schema instance namespace:
230 *http://www.w3.org/2001/XMLSchema-instance*

231 2. Discovery Service Information Model

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

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

240 2.1. Overview of Discovery Service Information Model

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

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

- 247 • A Principal wielding some *user agent* interacts with some *service provider* and is authenticated in some Liberty-
248 compliant fashion, such that the service provider obtains possession of a *discovery bootstrap* assertion for the
249 Principal. This assertion contains a pointer to the Principal's Discovery Service instance in the form of an ID-WSF
250 EPR.
 - 251 • Now, the service provider, acting as a *web service consumer* (WSC) and using the ID-WSF EPR obtained above,
252 queries the Principal's Discovery Service for a pointer to some other desired service of the Principal—e.g. the
253 Principal's Profile Service or Calendar Service.
 - 254 • The Discovery Service returns one or more ID-WSF EPRs to the querying WSC, pointing to the Principal's service
255 instance(s), of the requested type, if any.
 - 256 • The WSC now employs the returned ID-WSF EPR(s) to interact with the identified service instance(s), which
257 themselves will be acting in the role of a *web service provider*. The WSC returns results as appropriate to the
258 Principal's user agent.
- 259 There are various permutations of this general interaction model. For example, the Principal's user agent may
260 itself act in the role of a WSC. Or, a Principal may not be actively involved in a given interaction—a WSC is
261 simply interacting with a WSP on a Principal's behalf. For example, it may be renewing some contract, such as a
262 magazine subscription.

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

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

- 273 • The above Principal's-perspective model is now enabled.
- 274 There are various permutations of this general WSP-perspective service instance registration model. For example,
275 the same administrative entity may be deploying the both the Discovery Service and the other services and so
276 may employ alternative means, e.g. bulk configuration, to effect service metadata registration with their Discovery
277 Service.

278 2.2. Versioning in ID-WSF

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

282 The specific areas of versioning include:

- 283 • **Service Versioning** — the version of the Service APIs that are available from a service instance.
- 284 The service version is implicitly part of the `<ServiceType>` URI which identifies a specific version of a
285 specific logical service (e.g. a Profile service or a Discovery Service). For example, the service type URI:
286 `urn:liberty:disco:2006-08` represents a specific version of the logical service "Discovery Service" (in this
287 case, a version identified by the date 2006-08).
- 288 There are times when the Discovery Service may need to know the logical type of service. In the case of Liberty
289 defined services, we have adopted the convention of using a colon separated URN in the "liberty" namespace
290 where the last element is the version identifier. In this case, the Discovery service could extract the logical type
291 of service from the service type. However, that is just the Liberty convention and does not necessarily apply to
292 service type definitions outside of the Liberty URN namespace.
- 293 In non-Liberty-defined `<ServiceType>` URIs, the Discovery Service may understand the convention used in that
294 particular namespace, or it may have some configuration defined knowledge of which URIs match to which logical
295 type of service (perhaps via a user configurable parameter). If the DS cannot separate the logical type of service
296 from the service version, the DS SHOULD treat the entire `<ServiceType>` URI as the logical type of service.
- 297 The `<ServiceType>` URI is also typically used as the Namespace identifier for the XML schema for the service,
298 so the version identifier typically shows up there as well – although this is NOT a normative requirement.
- 299 • **Framework Versioning** — the version of the communications framework used for ID-WSF messaging. Each ID-
300 WSF message has a potential collection of SOAP headers defined by the various ID-WSF specifications which are
301 tied together by the [LibertyIDWSF20SCR]. The [LibertySOAPBinding] specification defines the `<Framework>`
302 element which carries a description of the framework. As of this release that consists primarily of a `version`
303 attribute. [LibertyIDWSF20SCR] defines a particular version string to represent each concrete version of the
304 specifications.
- 305 The `Framework description` is included in ID-WSF messages, ID-WSF minted EPRs and in Discovery
306 Service `<Query>` operations (in other words, the framework description is actively specified at each stage of
307 the ID-WSF interaction model).

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

311 2.3. ID-WSF Endpoint References (EPRs)

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

```

313
314 <wsa:EndpointReference ...>
315   <wsa:Address>...some URI here...</wsa:Address>
316
317   <wsa:ReferenceParameters>.....</wsa:ReferenceParameters>
318
319   <wsa:Metadata>...some metadata here... </wsa:Metadata>
320 </wsa:EndpointReference>
321

```

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

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

328 **Note**

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

```

331
332 <wsa:EndpointReference
333   notOnOrAfter="2005-08-15T23:18:56Z"
334   ...>
335   <wsa:Address>
336     https://profile-provider.com/profiles/someFoobarProfile
337   </wsa:Address>
338
339   <wsa:Metadata>
340     <ds:Abstract>
341       This is a personal profile containing common name information.
342     </ds:Abstract>
343
344     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
345
346     <ds:ServiceType>&PS1Namespace;</ds:ServiceType>
347
348     <ds:Framework version="2.0" />
349
350     <ds:SecurityContext>
351       <ds:SecurityMechID>
352         urn:liberty:security:2005-02:TLS:SAML
353       </ds:SecurityMechID>
354
355       <sec:Token>
356         <!-- some security token goes here -->
357       </sec:Token>
358     </ds:SecurityContext>
359
360     <ds:Options>
361       <ds:Option>urn:liberty:id-sis-pp</ds:Option>
362       <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
363       <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
364       <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
365     </ds:Options>
366   </wsa:Metadata>
367 </wsa:EndpointReference>
368

```

369 **Example 2. An Instantiated ID-WSF EPR**

370 **2.3.1. EPR Profiling Attributes**

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

374 **2.3.1.1. wsu:Id — unique identifier for xml references to an EPR.**

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

```
377  
378  
379 <!-- wsu:Id attribute for EPR - for xml document references to EPR -->  
380  
381 <xs:attribute ref="wsu:Id" use="optional" />  
382
```

383 **Figure 1. wsu:Id — Schema Fragment**

384 **2.3.1.2. reqRef — request reference**

385 The `reqRef` attribute ([Figure 2](#)) identifies which `<RequestedServiceType>` element in the Discovery Service
386 `<Query>` request that this EPR was minted in response to. In other words this is used to associate the EPR in
387 the `<QueryResponse>` with the `<RequestedServiceType>` in the `<Query>` request.

```
388  
389 <!-- Request Reference - to id which Request generated EPR -->  
390  
391 <xs:attribute name="reqRef" type="xs:string" use="optional"/>  
392
```

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

394 **2.3.1.3. notOnOrAfter**

395 The `notOnOrAfter` attribute states the expiration timestamp for the EPR with which it is associated ([Figure 3](#)). See
396 [Example 2](#), above, for an instantiated EPR example.

397 Values of the `notOnOrAfter` attribute MUST be expressed in accordance with Liberty ID-WSF time value restric-
398 tions.

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

```
401  
402 <!-- EPR Expiration Timestamp -->  
403  
404 <xs:attribute name="NotOnOrAfter" type="xs:dateTime" />  
405  
406
```

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

408 **2.3.2. EPR Profiling Elements**

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

412 **2.3.2.1. Abstract**

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

```
415  
416 <!-- Abstract: natural-language description of service -->  
417  
418 <xs:element name="Abstract" type="xs:string"/>  
419  
420
```

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  
429
```

430 **Figure 5. <ProviderID> — Schema Fragment**

431 **2.3.2.3. Service Type**

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

```
435  
436 <!-- Service Type -->  
437  
438 <xs:element name="ServiceType" type="xs:anyURI"/>  
439  
440
```

441 **Figure 6. <ServiceType> — Schema Fragment**

442 Some examples of possible ServiceType URIs:

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

446 **2.3.2.4. securityContext**

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

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

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

455 • urn:liberty:disco:tokenref:ObtainFromIDP

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

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

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

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

```
466  
467 <!-- Security Context Container -->  
468  
469 <xs:element name="SecurityContext">  
470   <xs:complexType>  
471     <xs:sequence>  
472       <xs:element ref="SecurityMechID"  
473         minOccurs="1"  
474         maxOccurs="unbounded" />  
475  
476       <xs:element ref="sec:Token"  
477         minOccurs="0"  
478         maxOccurs="unbounded" />  
479     </xs:sequence>  
480   </xs:complexType>  
481 </xs:element>  
482  
483
```

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

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

487 2.3.2.5. SecurityMechID

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

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

```
493
494 <!-- Security Mechanism ID -->
495
496 <xs:element name="SecurityMechID" type="xs:anyURI" />
497
498
```

Figure 8. <SecurityMechID> — Schema Fragment

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

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

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

506 2.3.2.6. Framework

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

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

511 The structure and content of this element is defined in [LibertySOAPBinding].

```
512
513 <!-- Framework Description -->
514
515 <xs:element ref="sb:Framework" maxOccurs="unbounded" />
516
517
```

Figure 9. <Framework> — Schema Fragment

519 2.3.2.7. Action

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

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

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

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

```
529
530 <!-- Action(s) - the interfaces available at this service -->
531
532 <xs:element name="Action" type="xs:anyURI" />
533
```

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

535 2.3.2.8. Options

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

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

```
543
544 <!-- Options -->
545
546 <xs:element name="Options" type="OptionsType"/>
547
548 <xs:element name="Option" type="xs:anyURI" />
549
550 <xs:complexType name="OptionsType">
551   <xs:sequence>
552     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/>
553   </xs:sequence>
554 </xs:complexType>
555
556
```

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

558 The <Options> element contains zero or more <Option> elements, each of which contains a URI identifying a
559 particular option. The set of possible URIs for an <Option> element should be defined by the service type. For
560 example, a person profile service specification would specify a set of options particular to its own domain. However,
561 one common <Option> flag related to security, and thus common to ID-WSF services, is defined in [Section 3.11](#):
562 Option Value for Response Authentication .

563 2.3.3. ID-WSF Web Services Addressing EPR Profile

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

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

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

572 Note

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

```
576  
577 <xs:element name="EndpointReference" type="tns:EndpointReferenceType" />  
578  
579 <xs:complexType name="EndpointReferenceType">  
580   <xs:sequence>  
581     <xs:element name="Address"  
582       type="tns:AttributedURIType" />  
583  
584     <xs:element name="ReferenceParameters"  
585       type="tns:ReferenceParametersType"  
586       minOccurs="0" />  
587  
588     <xs:element ref="tns:Metadata"  
589       minOccurs="0" />  
590  
591     <xs:any namespace="##other"  
592       processContents="lax"  
593       minOccurs="0"  
594       maxOccurs="unbounded" />  
595   </xs:sequence>  
596  
597   <xs:anyAttribute namespace="##other" processContents="lax" />  
598 </xs:complexType>  
599  
600  
601
```

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

603 2.3.3.1. ID-WSF EPR Minting Rules

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

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

- 609 1. A `notOnOrAfter` attribute **MAY** be present in each ID-WSF EPR. If absent, or if it has a value of `1970-01-`
610 `01T00:00:00Z`, it means the issuer is not stipulating an expiration time for this ID-WSF EPR, and that its wielder
611 is obliged to follow its own local policy for refreshing any cached copies. If present, the value should be set by
612 the issuer according to local policy.
- 613 2. The value of the <wsa:Address> element **MUST** contain the endpoint address of the service instance being
614 described by this EPR. This literally-addressed form of ID-WSF EPR is useful in order to ease the burden
615 of WSCs from having to retrieve and parse WSDL in common cases. Additionally, the rules specified in
616 [Section 2.3.3.2: ID-WSF EPR Specifics](#) **MUST** be adhered to.
- 617 3. A `wsu:Id` attribute **MAY** be present on the EPR root element.
- 618 4. A `reqRef` attribute **MAY** be present on the EPR root element.
- 619 5. Exactly one <Abstract> element **MAY** be present in the EPR <Metadata> element.

- 620 6. Exactly one <ProviderID> element MUST be present in the EPR <Metadata> element.
- 621 7. One or more <ServiceType> elements MUST be present in the EPR <Metadata> element.
- 622 8. One or more <Framework> elements MUST be present in the EPR <Metadata> element.
- 623 9. Optionally, one or more <Options> element(s). These are discussed in detail above, in [Section 2.3.2.8](#).
- 624 10. Optionally, one or more <Action> element(s). These are discussed in detail above, in [Section 2.3.2.7](#).
- 625 11. One or more <SecurityContext> elements SHOULD be present in each ID-WSF EPR. If so they, and their
626 content, MUST adhere to the rules below, as well as the additional specific rules in [Section 2.3.3.3: Security](#)
627 Mechanism Specifics :
- 628 a. If no security or identity tokens are to be embedded, then place all the supported security mechanisms,
629 denoted by <SecurityMechID> elements, in a single <SecurityContext> element.
- 630 b. Else, if security and/or identity tokens are to be embedded or referenced (via <sec:Token> ele-
631 ments), then one MUST group corresponding <SecurityMechID> and <sec:Token> elements into
632 the same <SecurityContext> element. In other words, all security and identity tokens within a
633 <SecurityContext> element MUST apply to ALL of the security mechanisms in the same context.
- 634 c. A security and/or identity token embedded in a <sec:Token> in a given ID-WSF EPR's
635 <SecurityContext> element MAY be referenced from other <SecurityContext> elements, whether
636 the other <SecurityContext> elements are contained within the given ID-WSF EPR or whether they are
637 in another ID-WSF EPR in the list of ID-WSF EPRs being constructed.
- 638 Such referencing is accomplished by using the `ref` attribute of a <sec:Token> element. When constructing
639 such a reference, the referencing <sec:Token> MUST reference the <sec:Token> element containing the
640 target embedded security token, as specified in [\[LibertySecMech\]](#).
- 641 d. All <sec:Token> elements included in the <SecurityContext> element MUST have the `usage` attribute
642 set to the appropriate value (as documented in [\[LibertySecMech\]](#)) indicating their intended purpose.
- 643 e. If the issuer is unable to generate a necessary token, it MUST include an empty <sec:Token> element with
644 the `ref` attribute set to the value `urn:liberty:disco:tokenref:ObtainFromIDP`

645 2.3.3.2. ID-WSF EPR Specifics

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

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

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

- 653 • The <wsdl:binding> contains a <wsdlsoap:binding> element. This specifies that the SOAP binding for
654 WSDL is being used.
- 655 • The `style` attribute of the <wsdlsoap:binding> element is "document".
- 656 • The `transport` attribute of the <wsdlsoap:binding> element is `http://schemas.xmlsoap.org/soap/http`.

657 • The abstract WSDL corresponding to the `<ServiceType>` MUST contain a single `<portType>` element. The
658 `<wsdl:binding>` element provides bindings for the operations specified in this `<wsdl:portType>`. Each
659 operation binding includes an input element and an output element, each containing a single `<wsdlsoap:body>`
660 element. The `use` attribute of the `<wsdlsoap:body>` elements is "literal".

661 • The `location` attribute of `<wsdlsoap:address>` is equal to `<wsa:Address>`.

662 • All other optional elements and attributes are not specified and thus default to the SOAP binding of WSDL.

663 2.3.3.3. Security Mechanism Specifics

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

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

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

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

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

681 2.3.3.4. Action Specifics

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

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

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

688 2.3.3.5. Identity Invocation Context specifics

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

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

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

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

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

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

706 **Note**

707 Since the security tokens usually carry the identity of the `Sender` and that of the `InvocationIdentity` it is
708 possible that a single `<SecurityContext>` may include multiple security 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
710 the `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
718 invocation identity **MUST** be expressed in the minted ID-WSF EPR as detailed in the rules above (first bullet
719 item).

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

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

725 • A token with the `usage` attribute set to `urn:liberty:security:tokenusage:2006-02:SecurityToken`
726 contains a security token that **MUST** be placed into the `<wsse:Security>` header block (according to [\[Liberty-](#)
727 [SecMech\]](#) and its related profiles) when a message is generated for the target of the ID-WSF EPR.

728 If multiple `<sec:Token>`s are included in a single `<ds:SecurityContext>`, they **MUST ALL** be placed into
729 the same `<wsse:Security>` header block.

730 • A token with the `usage` attribute set to `urn:liberty:security:tokenusage:2006-08:TargetIdentity`
731 contains an identity token that **MUST** be placed into the `<sb:TargetIdentity>` header block (according to
732 [\[LibertySOAPBinding\]](#)) when a message is generated for the target of the ID-WSF EPR.

733 **2.3.4. Effective Web Services Addressing EPR**

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

736 • There is no normative XML schema defined as such, this is just an approximation of what the schema could look
737 like.

- 738 • While the elements within the <Metadata> element appear to be ordered, they can all appear in any order and can
- 739 have other elements appear between the listed elements. This is why they are contained within a multi-occurrence
- 740 <xs:choice>.

- 741 • Four attributes have been added to the EPR element itself: the notOnOrAfter timestamp and three different IDs
- 742 (for use in different circumstances).

- 743 • Seven sub-elements were added to the <Metadata> element.

- 744 • The <Metadata> sub-element <disco:ProviderID> MUST appear exactly once in an ID-WSF EPR, even
- 745 though the schema below does not enforce that requirement (because of a limitation in XML Schema – or perhaps
- 746 in the author’s understanding of XML schema).

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

```

750
751 ...
752 <xs:element name="EndpointReference" type="tns:EndpointReferenceType"/>
753 <xs:complexType name="EndpointReferenceType" mixed="false">
754   <xs:sequence>
755     <xs:element name="Address" type="tns:AttributedURIType"/>
756     <xs:element name="ReferenceParameters"
757       type="tns:ReferenceParametersType" minOccurs="0"/>
758     <xs:element ref="tns:Metadata" minOccurs="0"/>
759     <xs:any namespace="##other" processContents="lax"
760       minOccurs="0" maxOccurs="unbounded"/>
761   </xs:sequence>
762   <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional" />
763   <xs:attribute ref="wsu:Id" use="optional" />
764   <xs:attribute name="reqRef" type="xs:string" use="optional" />
765   <xs:anyAttribute namespace="##other" processContents="lax"/>
766 </xs:complexType>
767
768 <xs:complexType name="ReferenceParametersType" mixed="false">
769   <xs:sequence>
770     <xs:any namespace="##any" processContents="lax"
771       minOccurs="0" maxOccurs="unbounded" />
772   </xs:sequence>
773   <xs:anyAttribute namespace="##other" processContents="lax"/>
774 </xs:complexType>
775
776 <xs:element name="Metadata" type="tns:MetadataType"/>
777 <xs:complexType name="MetadataType" mixed="false">
778   <xs:sequence>
779     <xs:choice minOccurs="0" maxOccurs="unbounded">
780       <xs:element ref="disco:Abstract" minOccurs="0" />
781       <xs:element ref="sbf:Framework" maxOccurs="unbounded"/>
782       <xs:element ref="disco:ProviderID" />
783       <xs:element ref="disco:ServiceType" maxOccurs="unbounded"/>
784       <xs:element ref="disco:SecurityContext" minOccurs="0" maxOccurs="unbounded" />
785       <xs:element ref="disco:Options" minOccurs="0" maxOccurs="unbounded" />
786       <xs:element ref="disco:Action" minOccurs="0" maxOccurs="unbounded" />
787       <xs:any namespace="##other" processContents="lax"
788         minOccurs="0" maxOccurs="unbounded" />
789     </xs:choice>
790   </xs:sequence>
791   <xs:anyAttribute namespace="##other" processContents="lax"/>
792 </xs:complexType>
793

```

794 **Example 3. Effective ID-WSF EPR Schema**

795 2.3.5. Example Liberty ID-WSF EPRs

```
796
797 <wsa:EndpointReference
798     notOnOrAfter="2005-08-15T23:18:56Z"
799     ...>
800 <wsa:Address>
801     http://profile-provider.com/profiles/someFoobarProfileAddr
802 </wsa:Address>
803
804 <wsa:Metadata>
805     <ds:Abstract>
806         This is a personal profile containing common name information.
807     </ds:Abstract>
808
809     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
810
811     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
812
813     <sbf:Framework version="2.0" />
814
815     <ds:SecurityContext>
816         <ds:SecurityMechID>
817             urn:liberty:security:2006-08:ClientTLS:SAMLV2
818         </ds:SecurityMechID>
819
820         <ds:SecurityMechID>
821             urn:liberty:security:2005-02:ClientTLS:SAML
822         </ds:SecurityMechID>
823
824         <sec:Token wsu:id="_10"
825             usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
826             <!-- some security token goes here -->
827         </sec:Token>
828     </ds:SecurityContext>
829
830     <ds:SecurityContext >
831         <ds:SecurityMechID>
832             urn:liberty:security:2005-02:ClientTLS:X509
833         </ds:SecurityMechID>
834
835         <sec:Token wsu:id="_20"
836             usage="urn:liberty:security:tokenusage:2006-08:InvocationIdentity">
837             <!-- Identity Token goes here -->
838         </sec:Token>
839     </ds:SecurityContext>
840
841     <ds:Options>
842         <ds:Option>urn:liberty:id-sis-pp</ds:Option>
843         <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
844         <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
845         <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
846     </ds:Options>
847 </wsa:Metadata>
848 </wsa:EndpointReference>
849
850 <wsa:EndpointReference
851     notOnOrAfter="2005-08-15T23:18:56Z"
852     ...>
853 <wsa:Address>
854     http://profile-provider.com/profiles/anotherFoobarProfileEndpointAddr
855 </wsa:Address>
856
857 <wsa:Metadata>
858     <ds:Abstract>
859         This is a personal profile containing common name information.
```

```

860     </ds:Abstract>
861
862     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
863
864     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
865
866     <sb:Framework version="2.0" />
867
868     <ds:SecurityContext>
869         <ds:SecurityMechID>
870             urn:liberty:security:2006-08:TLS:SAMLV2
871         </ds:SecurityMechID>
872
873         <sec:Token ref="_10" usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" />
874     </ds:SecurityContext>
875
876     <ds:Options>
877         <ds:Option>urn:liberty:id-sis-pp</ds:Option>
878         <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
879         <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
880         <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
881     </ds:Options>
882 </wsa:Metadata>
883 </wsa:EndpointReference>
884

```

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

887 2.4. Service Metadata

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

890 2.4.1. Service Metadata element

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

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

897 Most of the fields present in the Service Metadata have the same purpose and meaning as the elements of the same
898 name in the ID-WSF EPR (as this is where the Discovery service gets those elements for the ID-WSF EPR).

899 When fields permit multiple values, the order of entries in the SvcMD is significant with higher preference items
900 coming first. This comes into play should the WSC request a subset of the possible results when querying the
901 Discovery Service (in which case the entries with the higher preference – those listed first – would be used to mint the
902 ID-WSF EPRs in the response).

```
903
904 <!-- Service Metadata (SvcMD) - metadata about service instance -->
905
906 <xs:element name="SvcMD" type="SvcMetadataType"/>
907 <xs:complexType name="SvcMetadataType">
908   <xs:sequence>
909     <xs:element ref="Abstract"           />
910     <xs:element ref="ProviderID"        />
911     <xs:element ref="ServiceContext"    maxOccurs="unbounded" />
912   </xs:sequence>
913   <xs:attribute name="svcMDID" type="xs:string" use="optional" />
914 </xs:complexType>
915
916 <!-- ServiceContext - describes service type/option/endpoint context -->
917 <xs:element name="ServiceContext" type="ServiceContextType"/>
918 <xs:complexType name="ServiceContextType">
919   <xs:sequence>
920     <xs:element ref="ServiceType"      maxOccurs="unbounded" />
921     <xs:element ref="Options"          minOccurs="0"
922                                     maxOccurs="unbounded" />
923     <xs:element ref="EndpointContext" maxOccurs="unbounded" />
924   </xs:sequence>
925 </xs:complexType>
926
927 <!-- EndpointContext - describes endpoints used to access service -->
928 <xs:element name="EndpointContext" type="EndpointContextType" />
929 <xs:complexType name="EndpointContextType">
930   <xs:sequence>
931     <xs:element ref="Address"          maxOccurs="unbounded" />
932     <xs:element ref="sbf:Framework"   maxOccurs="unbounded" />
933     <xs:element ref="SecurityMechID"   maxOccurs="unbounded" />
934     <xs:element ref="Action"          minOccurs="0"
935                                     maxOccurs="unbounded" />
936   </xs:sequence>
937 </xs:complexType>
938
939 <!-- SvcMD ID element used to refer to Service Metadata elements -->
940 <xs:element name="SvcMDID" type="xs:string" />
941
```

942 **Figure 13. Service Metadata — Schema Fragment**943 **2.4.1.1. svcMDID**

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

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

948 **2.4.1.2. Abstract**

949 A text description of the service.

950 **2.4.1.3. ProviderID**

951 The URI of the provider of this service instance.

952 **2.4.1.4. ServiceContext**

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

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

957 **2.4.1.4.1. ServiceType**

958 The URI of which defines the type of service.

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

962 **2.4.1.4.2. Option**

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

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

965 **2.4.1.4.3. EndpointContext**

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

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

970 The sub-elements include:

971 **2.4.1.4.3.1. Address**

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

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

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

978 **2.4.1.4.3.2. Framework**

979 The SOAP Bindings ([\[LibertySOAPBinding\]](#)) `<sbef:Framework>` element describing the version of the ID-WSF
980 framework supported at this endpoint..

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

983 **2.4.1.4.3.3. SecurityMechID**

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

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

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

990 **2.4.1.4.3.4. Action**

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

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

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

995 **2.4.2. Minting ID-WSF EPRs based upon Service Metadata**

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

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

1000 The Discovery Service MUST mint ID-WSF EPRs as if the following process took place (there is NOT a normative
1001 requirement to implement this exact process, just a requirement that the results generated by whatever process is used
1002 by the DS MUST result in the set of data that would result from this process).

1003 1. Eliminate portions of the Service Metadata that do not conform to the search requirements (such as unsupported
1004 (by the WSC) security mechanisms or framework versions, or undesired service types).

1005 2. If an <EndpointContext> element had all occurrences of a given sub-element (such as <Framework>)
1006 eliminated, eliminate the context.

1007 3. For each remaining <Address>) element within a remaining <EndpointContext> element, an EPR SHOULD
1008 be minted.

1009 4. For each EPR, assign one <Address>) element to the <wsa:Address>) element in the EPR and use the rest of
1010 the <EndpointContext> that contained this address to build the necessary <Metadata> sub-elements for the
1011 ID-WSF EPR (. <SecurityContext>(s), <Action>(s), <Framework>(s), etc.).

1012 5. Fill out the rest of the ID-WSF EPR using the service wide elements (<Abstract>, <ProviderID>,
1013 <ServiceType>(s), etc.).

1014 6. If necessary, generate any security and/or identity tokens and place them into the appropriate
1015 <SecurityContext> element(s).

1016 The set of EPRs generated by this process may be further restricted by the request parameters on the DiscoveryQuery
1017 operation, see [Section 3.3](#)).

1018 **2.4.3. Service Metadata Example**

1019 Some examples to help show how service metadata works.

1020 **2.4.3.1. A simple service**

1021 This is an example of a simple service that has a single endpoint, supports a single framework version (2.0), and only
1022 supports a single security mechanism.

```
1023
1024 <ds:svcMD svcMDID="1234">
1025   <ds:Abstract>This is a simple service metadata definition</ds:abstract>
1026   <ds:ProviderID>http://simpler.providers.com</ds:ProviderID>
1027   <ds:ServiceContext>
1028     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1029     <ds:EndpointContext>
1030       <ds:Address>https://simple.providers.com/PP</ds:Address>
1031       <sb:Framework version="2.0" />
1032       <ds:SecurityMechID>
1033         urn:liberty:security:2003-08:TLS:Bearer
1034       </ds:SecurityMechID>
1035     </ds:EndpointContext>
1036   </ds:ServiceContext>
1037 </ds:SvcMD>
1038
```

1039 **Figure 14. Service Metadata example: A simple service**

1040 **2.4.3.2. A complex service**

1041 This is an example of a service metadata definition with a number of complex attributes including:

- 1042 • Multiple service versions **and** multiple framework versions on the same endpoint.
- 1043 • There are two service contexts, one for one version of the service and one for a different version of the service.
1044 So, for example, the 2003-08 version of the service is only available at the URL *https://old.providers.com/PP* and
1045 only for framework version *1.1*
- 1046 • Multiple interfaces on different endpoints with different security mechanisms
- 1047 • There are multiple, redundant, addresses for the TLS endpoint for the *2007-11* version of the service.

```

1048
1049 <!-- Service Metadata Example: A complex service -->
1050 <ds:SvcMD svcMDID="4567">
1051   <ds:Abstract>This is a complex service metadata definition</ds:abstract>
1052   <ds:ProviderID>http://complex.providers.com</ds:Provider ID>
1053   <ds:ServiceContext>>
1054     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1055     <ds:EndpointContext>
1056       <ds:Address>https://old.providers.com/PP</ds:Address>
1057       <sb:Framework version="1.1" />
1058       <ds:SecurityMechID>
1059         urn:liberty:security:2003-08:TLS:Bearer
1060       </ds:SecurityMechID>
1061     </ds:EndpointContext>
1062   </ds:ServiceContext>>
1063   <ds:ServiceContext>>
1064     <ds:ServiceType>urn:liberty:pp:2007-11</ds:ServiceType>
1065     <ds:EndpointContext>
1066       <ds:Address>https://complex.providers.com/PP</ds:Address>
1067       <ds:Address>https://backup.complex.providers.com/PP</ds:Address>
1068       <sb:Framework version="2.0" />
1069       <ds:SecurityMechID>
1070         urn:liberty:security:2003-08:TLS:Bearer
1071       </ds:SecurityMechID>
1072     </ds:EndpointContext>
1073     <ds:EndpointContext>
1074       <ds:Address>http://complex.providers.com/PP</ds:Address>
1075       <sb:Framework version="2.0" />
1076       <ds:SecurityMechID>
1077         urn:liberty:security:2003-08:null:SAMLV2
1078       </ds:SecurityMechID>
1079     </ds:EndpointContext>
1080   </ds:ServiceContext>>
1081 </ds:SvcMD>
1082

```

1083 **Figure 15. Service Metadata example: A complex service**

1084 2.4.3.3. Another complex service

1085 This is an example of a service metadata definition where the service has some of its operations at one endpoint and
 1086 others at a different endpoint (thus splitting the service operations across different instances).

1087 This service is still defined with a single service context since the endpoints all expose the same service type.

```
1088
1089 <!-- Service Metadata Example: A simple service -->
1090 <ds:SvcMD svcMDID="8901">
1091   <ds:Abstract>Another example complex service</ds:abstract>
1092   <ds:ProviderID>http://split.providers.com</ds:ProviderID>
1093   <ds:ServiceContext>>
1094     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1095     <ds:EndpointContext>
1096       <ds:Address>https://cluster1.split.providers.com/PP</ds:Address>
1097       <sb:Framework version="2.0" />
1098       <ds:SecurityMechID>
1099         urn:liberty:security:2003-08:TLS:Bearer
1100       </ds:SecurityMechID>
1101       <ds:Action>urn:liberty:pp:2003-08:Query</ds:Action>
1102     </ds:EndpointContext>
1103     <ds:EndpointContext>
1104       <ds:Address>https://writer.split.providers.com/PP</ds:Address>
1105       <sb:Framework version="2.0" />
1106       <ds:SecurityMechID>
1107         urn:liberty:security:2003-08:TLS:Bearer
1108       </ds:SecurityMechID>
1109       <ds:Action>urn:liberty:pp:2003-08:Modify</ds:Action>
1110     </ds:EndpointContext>
1111   </ds:ServiceContext>>
1112 </ds:SvcMD>
1113
```

1114

Figure 16. Service Metadata example: Another complex service

1115 3. Discovery Service

1116 A Discovery Service is a web service providing both identity based and non-identity based operations.

1117 The identity based Discovery Service interfaces facilitate requesters' discovery of identity service instances on a
1118 per-identity basis, and acquisition of ID-WSF Endpoint References (ID-WSF EPRs) "pointing" to the discovered
1119 service instances. These ID-WSF EPRs provide requesters with the information necessary to invoke discovered service
1120 instances.

1121 The non-identity based Discovery Service interfaces provide a WSP with principal-independent management of their
1122 metadata stored at the Discovery Service (which is, through an identity-based interface, associated with a principal).

1123 Thus in an abstract sense, the Discovery Service is essentially a web service interface to per-identity "discovery
1124 resources", each of which can be viewed as a registry of ID-WSF EPRs. The notion of "discovery resources" is an
1125 abstract way of referring to what are concretely "identity-indexed Discovery Service instances".

1126 The Discovery Service can also be used as a non-identity service to discover and obtain ID-WSF EPRs for non-identity
1127 services. For example, the Discovery Service could be used to locate the available Authentication Services before a
1128 principal identity has been established.

1129 Entities can register ID-WSF EPRs, pointing to their identity services, with a discovery resource, and this will allow
1130 other entities to discover them. A common use case is that a Principal places references (aka ID-WSF EPRs) to his
1131 or her personal profile, calendar, and so on, in a discovery resource so that they may be discovered by other entities,
1132 e.g. web service providers who wish to provide the Principal with value-added services.

1133 When invoked as an identity service, the act of discovering service instances is implicitly on a per-identity basis. This
1134 occurs in a number of fashions in ID-WSF including:

1135 • When a Principal authenticates to a service provider using a SAMLv2 profile (or similarly via
1136 ID-FF), the identity provider conveys, within the authentication assertion, an ID-WSF EPR pointing
1137 explicitly to the *Principal's* discovery service resource, which the SP may then use to discover the
1138 Principal's various services.

1139 • A Principal's (LUAD-)WSC authenticates via the Authentication Service (see [\[LibertyAuthn\]](#)),
1140 which will likely return an ID-WSF EPR for the Principal's Discovery Service resource.

1141 • Any Identity Token (see [\[LibertySecMech\]](#)), or security token may contain a Discovery Service
1142 bootstrap ID-WSF EPR (see [Section 4: Discovery Service ID-WSF EPR conveyed via a Security Token](#)
1143) which contains the necessary information to access the Principal's Discovery Service resource.

1144 The Discovery service is identified by ID-WSF EPRs, which themselves have been crafted (typically by an identity
1145 provider) such that they identify the discovery service resource (aka Discovery Service instance) mapped to the
1146 Principal in question.

1147 The Discovery Service is intended to be used in conjunction with other ID-WSF specifications. For example, security
1148 mechanisms are not specified here, because they are defined in [\[LibertySecMech\]](#). At the same time, the Discovery
1149 Service is specified such that it could be used with other security mechanisms, not yet defined.

1150 The Discovery Service is designed to be describable by WSDL [[WSDLv1.1](#)], and an abstract WSDL definition
1151 is included in this document, see [Appendix B: Discovery Service WSDL](#). This WSDL document defines two
1152 "WSDL operations" for the Discovery Service. The first is the *DiscoveryQuery* operation. This operation returns
1153 an enumeration of ID-WSF EPRs for a given search criteria.

1154 To enforce access control policies, security tokens may need to be presented by the client when interacting with a
1155 Discovery Service instance. While the definition of these security tokens is outside the scope of this specification, it
1156 is common for the same provider that is hosting the Discovery Service to also be the entity that generates the security

1157 tokens necessary to access the service. To avoid extra network round-trips, arrangements are made here so that
 1158 security tokens may be provided as part of the Discovery Service lookup response.

1159 3.1. Service URIs

1160 **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>

1161 3.2. Status Codes

1162 The following status code strings are defined:

- 1163 • *OK*: message processing succeeded
- 1164 • *Failed*: general failure code
- 1165 • *Forbidden*: the request was denied based on policy
- 1166 • *Duplicate*: the request was denied because it would result in duplicate data in the service
- 1167 • *LogicalDuplicate*: the request was denied because it would result in logically duplicate data in the service
- 1168 • *NoResults*: the query had no matching results
- 1169 • *NotFound*: the specified item(s) were not found

1170 These strings are expected to appear in the "code" attribute of <Status> elements used in SOAP-bound Discovery
1171 Service protocol messages [[LibertySOAPBinding](#)]. Specific uses for the status codes are defined in the processing
1172 rules for individual messages. The "ref" attribute on the <Status> element is not used in this specification, so it MUST
1173 NOT appear on Status elements in Discovery Service protocol messages. The contents of the comment attribute are
1174 not defined by this specification, but it may be used for additional descriptive text intended for human consumption
1175 (for example, to carry information that will aid debugging).

1176 3.3. Operation: *DiscoveryQuery*

1177 The *DiscoveryQuery* WSDL operation enables a requester to obtain an enumeration of ID-WSF EPRs (see [Section 2:](#)
1178 [Discovery Service Information Model](#)) — the requester sends a <Query> message and receives a <QueryResponse>
1179 message in return. Also, because a provider hosting a Discovery Service may also be playing other roles on behalf
1180 of Principals (such as a *Policy Decision Point* or an *Authentication Authority*), the *DiscoveryQuery* operation can also
1181 function as a security token service, providing the requester with an efficient means of obtaining security tokens that
1182 may be necessary to invoke service instances described in the <QueryResponse>.

1183 3.3.1. wsa:Action values for DiscoveryQuery Messages

1184 <Query> messages MUST include a <wsa:Action> SOAP header with the value of urn:liberty:disco:2006-08:Query.

1185 <QueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1186 urn:liberty:disco:2006-08:QueryResponse.

1187 3.3.2. <Query> Message

1188 A <Query> request is an attempt to retrieve ID-WSF EPRs suitable for use in the same identity context that was used
1189 to make the request. In particular, the Target Identity (See [[LibertySOAPBinding](#)]), if applicable, is used to restrict the
1190 results to just those for the specified principal. The Invocation Identity will be verified against an access control list to
1191 ensure that they have access to the requested results.

1192 A <Query> request message is empty in the minimal case. Such a request indicates the requester is requesting all
1193 available ID-WSF EPRs, regardless of security mechanisms or service types. The result set is dependant upon the
1194 local access control policies of the discovery service instance.

1195 Alternatively, a request can be qualified with a set of <RequestedService> elements, which enables the requester to
1196 specify that all ID-WSF EPRs returned must be offered via one or more service instances complying with the specified
1197 search criteria. For each <RequestedService> specified, the requester specifies the search criteria for the DS to use
1198 in determining if there is a matching instance. The search criteria includes zero or more of any of the following:

1199 • <ServiceType> the requested type of service. Service Type URIs are defined by the individual service
1200 specifications and contain both service class and service versioning information.

1201 Multiple <ServiceType>s MAY be specified in a single <RequestedService>s element in order to allow the
1202 WSC to specify what the WSC considers to be different versions of the same service.

1203 When multiple entries are listed, the order of such entries is an indication of the preference as to which
1204 <ServiceType> the WSC would prefer to see in the results, with the first being the most preferred. This
1205 typically only impacts a request where the WSC indicates that they only want a subset of the results returned (see
1206 [resultsType](#) below).

1207 When a request results in multiple ID-WSF EPRs in a response, the preference order specified by the WSC on the
1208 request MAY have no impact on the order of results returned by the Discovery Services. The Discovery Service
1209 is free to return the results of the request in whatever order it chooses.

1210 If not specified, then any service instance would be considered a match for this criteria.

1211 A service instance must support at least one of the specified service types in order to be considered a match for
1212 this criteria.

- 1213 • `<ProviderID>` the requested provider ID(s). This is used when the WSC wants to communicate with a particular
1214 WSP. Frequently such requests are made without specifying a `<ServiceType>` element in the request, but doing
1215 so is not prohibited.
- 1216 If not specified, then any service instance would be considered a match for this criteria.
- 1217 A service instance must contain at least one of the specified providerIDs in order to be considered a match for this
1218 criteria.
- 1219 The order of the `<ProviderID>` elements is an indication as to the preference of the requester with the first such
1220 element being the most desired (declining preference order). The Discovery Services is free to return the results
1221 of the request in whatever order it chooses.
- 1222 • `<Options>` — an optional multi-occurrence element defining options SETs desired for the service.
- 1223 If not specified, any service instance will be considered a match for this criteria.
- 1224 An option SET is defined within each `<Options>` element and contains a list of the desired options. The service
1225 instance MUST support ALL of the options within the option SET in order to be considered a match for this
1226 request.
- 1227 If more than one `<Options>` element is specified (thus defining multiple option SETs), service instances that
1228 match ANY of the SETS are considered a match for this request. As noted above, to match a SET, you have to
1229 match ALL of the entries within the SET.
- 1230 Service instance EPRs registered without an `<Options>` element are always considered a match from the point of
1231 view of any possible `<Options>` search criteria.
- 1232 The order of the `<options>` elements is an indication as to the preference of the requester with the first such
1233 element being the most desired (declining preference order). The Discovery Services is free to return the results
1234 of the request in whatever order it chooses.
- 1235 • `<SecurityMechID>` - an optional multi-occurrence element specifying the security mechanism identifier(s) (see
1236 [[LibertySecMech](#)]) that the WSC is willing to use to invoke the WSP. If not specified, any security mechanism
1237 registered for a service will be considered a match for this criteria.
- 1238 A service instance MUST support at least one of the requested security mechanisms in order to be considered a
1239 match for this request.
- 1240 The order of the `<SecurityMechID>` elements is an indication as to the preference of the requester with the first
1241 such element being the most desired (declining preference order). The Discovery Services is free to return the
1242 results of the request in whatever order it chooses.
- 1243 • `<Framework>` — an optional multi-occurrence element specifying the framework description(s) supported by the
1244 WSC.
- 1245 If not specified, the Discovery Service SHOULD use the value of the framework description used in the ID-WSF
1246 framework layer for the current request (e.g. if the call to the Discovery Service was made using an ID-WSF
1247 version 2.0 message the request SHOULD be treated as if a `<disco:Framework>` element was present and
1248 contained the value specified in the `<sbf:Framework>` SOAP header.
- 1249 Multiple `<disco:Framework>` elements MAY be specified, indicating that the WSC has the capability to support
1250 ANY of the specified versions. The order of elements in such a case indicates the WSC's preference with the most
1251 preferred coming first.
- 1252 Note that while both the `<disco:Framework>` and the `<sbf:Framework>` elements are of the same type
1253 (`<sbf:FrameworkType>`), the elements themselves are in different namespaces. The element within the
1254 `<RequestedService>` is in the Discover Service Namespace, while the element within any ID-WSF EPRs and
1255 the SOAP header block on an ID-WSF message are in the SOAP Bindings namespace.

- 1256 • `<Action>` — an optional multi-occurrence element specifying the `wsa:Action` value(s) for the interfaces of the
1257 service that the WSC intends to make use of.
- 1258 If not specified, the Discovery Service SHOULD treat this request as a request for all of the interfaces at the
1259 requested specified instance.
- 1260 Unlike the other sub-elements of the `<RequestedService>` element, if multiple `<Action>` elements are
1261 specified it indicates that the WSC intends to invoke **all** of the specified interfaces and the Discovery Service
1262 SHOULD return the set of EPRs that are necessary to reach the complete set of specified interfaces.
- 1263 Services registered without an `<Action>` element (which is the norm) are treated as exposing **all** of the interfaces
1264 defined for that type of service.
- 1265 The Discovery Service will return the set of EPRs for service instances that intersect with the search criteria specified
1266 in the `<RequestedService>` element. This may result in a single EPR in the response or it may result in a multitude
1267 of EPRs, depending upon the search criteria and the service instance definitions available (see [Section 2.3.3: ID-WSF](#)
1268 [Web Services Addressing EPR Profile](#)).
- 1269 The result set of EPRs generated in response to a particular `<RequestedService>` element can be further controlled
1270 using the following attributes:
- 1271 • `reqID` — an optional attribute identifying this `<RequestedService>` request. Typically only used when
1272 multiple `<RequestedService>` elements are included in a single Discovery Service `<Query>`.
- 1273 If present the value of this attribute will be placed into the `reqRef` attribute in any EPRs that result from this
1274 `<RequestedService>` element (see [Section 2.3.1.2](#) above).
- 1275 The value of `reqID` SHOULD be different for all `<RequestedService>` elements in a given `<Query>`.
- 1276 • `resultsType` — an optional attribute describing the results desired by the requestor. This value may be set to:
- 1277 • **best** — the Discovery Service SHOULD return the smallest set of EPRs while still meeting the minimum
1278 requirements of the request. This will typically be a single EPR.
- 1279 • **all** — the Discovery Service SHOULD return all of the matching entries for the given search criteria.
1280 This would typically be used when the client wants to choose which EPRs within the DS database it should
1281 use.
1282 This option should be used with caution as it can cause the DS to perform substantial work in order to mint all
1283 of the matching EPRs and the necessary security tokens for those EPRs.
- 1284 • **only-one** — a restricted version of **best** which further restricts the resulting output to **exactly one** EPR, even if
1285 the minimum requirements of the request would require multiple EPRs. A client would typically specify this
1286 option if it was going to ignore anything other than the first EPR returned.
- 1287 If `resultsType` is not specified the Discovery Service may make its own determination (under local policy) as to
1288 which set of results to return.

1289 Requestors SHOULD include at least one <ServiceType> or <ProviderID> element, and MAY include any number
1290 of both of them.

1291 Requestors SHOULD construct a Query to be as qualified as possible, as the Discovery Service instance may have to
1292 perform significant work for each item in the result set, especially if security tokens will be generated.

```
1293
1294 <!-- Query Message Element & Type -->
1295
1296 <xs:element name="Query" type="QueryType"/>
1297
1298 <xs:complexType name="QueryType">
1299   <xs:sequence>
1300     <xs:element name="RequestedService"
1301       type="RequestedServiceType"
1302       minOccurs="0"
1303       maxOccurs="unbounded" />
1304   </xs:sequence>
1305
1306   <xs:anyAttribute namespace="##other" processContents="lax" />
1307 </xs:complexType>
1308
1309 <xs:complexType name="RequestedServiceType">
1310   <xs:sequence>
1311     <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
1312
1313     <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
1314
1315     <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
1316
1317     <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
1318
1319     <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
1320
1321     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
1322
1323     <xs:any namespace="##other"
1324       processContents="lax"
1325       minOccurs="0"
1326       maxOccurs="unbounded" />
1327   </xs:sequence>
1328
1329   <xs:attribute name="reqID" type="xs:string" use="optional" />
1330   <xs:attribute name="resultsType" type="xs:string" use="optional" />
1331 </xs:complexType>
1332
1333
1334
1335
```

1336 **Figure 17. Query Message — Schema Fragment**

```

1337
1338 <soap:Envelope
1339   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
1340
1341   <soap:Header>
1342     ...
1343   </soap:Header>
1344
1345   <soap:Body>
1346     <Query xmlns="DS2Namespace;">
1347       <RequestedService>
1348         <ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1349
1350         <SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</SecurityMechID>
1351         <SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</SecurityMechID>
1352         <SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</SecurityMechID>
1353         <Framework version="2.0" />
1354
1355       </RequestedService>
1356     </Query>
1357   </soap:Body>
1358 </soap:Envelope>
1359

```

1360 **Example 5. SOAP message containing a Query**

1361 3.3.3. QueryResponse

1362 A <QueryResponse> message conveys the results of the query as a set of ID-WSF EPRs, i.e. profiled
 1363 <wsa:EndpointReference> elements (see [Section 2.3.3: ID-WSF Web Services Addressing EPR Profile](#)).

1364 As specified in [Section 2.3.3](#), security tokens, appropriate for subsequent invocation(s) of the service instances
 1365 represented by the returned ID-WSF EPRs, MAY be provided within the ID-WSF EPRs in the response.

1366 A status code is also included in the response.

```

1367
1368 <!-- QueryResponse Message Element & Type -->
1369
1370 <xs:element name="QueryResponse" type="QueryResponseType"/>
1371
1372 <xs:complexType name="QueryResponseType">
1373   <xs:sequence>
1374     <xs:element ref="lu:Status"/>
1375
1376     <xs:element ref="wsa:EndpointReference"
1377       minOccurs="0"
1378       maxOccurs="unbounded"/>
1379   </xs:sequence>
1380   <xs:anyAttribute namespace="##other" processContents="lax"/>
1381 </xs:complexType>
1382
1383

```

1384 **Figure 18. <QueryResponse> — Schema Fragment**

1385 An example SOAP message containing a <QueryResponse> message is illustrated in [Example 6](#). This example
 1386 includes a security token embedded in the returned ID-WSF EPR. Parts of the security token have been omitted due
 1387 to size.

```

1388
1389 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

```

```

1390
1391 <soap:Header>
1392   ...
1393 </soap:Header>
1394
1395 <soap:Body>
1396   <QueryResponse xmlns="&DS2Namespace;" >
1397     <Status code="OK" />
1398
1399     <wsa:EndpointReference
1400       notOnOrAfter="2005-08-15T23:18:56Z" >
1401       <wsa:Address>http://example.com/pip/bob</wsa:Address>
1402
1403       <wsa:Metadata>
1404         <ds:Abstract>
1405           Bob's personal profile
1406         </ds:Abstract>
1407
1408         <ds:ProviderID>http://example.com/</ds:Provider ID>
1409
1410         <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1411
1412         <ds:Framework Version="2.0" />
1413
1414         <ds:SecurityContext>
1415           <ds:SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</ds:SecurityMechID>
1416           <ds:SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</ds:SecurityMechID>
1417
1418           <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" >
1419             <saml2:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1420               ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
1421               Issuer="idp.example.com"
1422               IssueInstant="2003-09-09T16:58:33.173Z">
1423               <ds:Signature>...</ds:Signature>
1424               <saml2:Subject>
1425                 <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1426                   http://serviceprovider.com/
1427                 </saml2:NameID>
1428
1429                 <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1430                 <saml2:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
1431                   <ds:KeyInfo>
1432                     <ds:KeyName>
1433                       CN=serviceprovider.com,
1434                       OU=Services R US,O=Service Nation,...
1435                     </ds:KeyName>
1436                   </ds:KeyInfo>
1437                 </saml2:SubjectConfirmationData>
1438               </saml2:SubjectConfirmation>
1439             </saml2:Subject>
1440
1441             <saml2:AuthnStatement AuthnInstant="2003-09-09T16:57:30.000Z"
1442               SessionIndex="..."
1443               SessionNotOnOrAfter="..."
1444             >
1445               <saml2:AuthnContext
1446                 ...
1447               </saml2:AuthnContext>
1448             </saml2:AuthnStatement>
1449
1450           </saml2:Assertion>
1451         </sec:Token>
1452       </ds:SecurityContext>
1453     </wsa:Metadata>
1454   </wsa:EndpointReference>
1455 </QueryResponse>
1456 </soap:Body>

```

1457 </soap:Envelope>
1458

1459 **Example 6. SOAP-bound <QueryResponse> Message with Embedded Security Token**

1460 3.3.4. DiscoveryQuery Processing Rules

1461 The discovery Service returns entries based on the requester's search criteria (interpreted as described above in
1462 [Section 3.3.2: <Query> Message](#)), the policies of the discovery resource, and the contents of the discovery resource.

1463 For each <RequestedService> element in a <Query> message, the matching rules **MUST** applied independently
1464 (as if the other <RequestedService> elements were not present (potentially returning equivalent EPRs in response
1465 to different <RequestedService> elements).

1466 When building the results for a given <RequestedService> element, the Discovery Service **SHOULD** return the
1467 data in as few EPRs as possible (within the constraints of the EPR) especially with respect to data originating from the
1468 same SvcMD.

1469 The Discovery Service **SHOULD**, when possible, provide the security tokens necessary for the security mechanism(s)
1470 identified in the ID-WSF EPRs in the response. If the Discovery Service is not able to generate the necessary security
1471 token, it should indicate so by including an empty <sec:Token> element with the ref attribute set to the value:

1472 urn:liberty:disco:tokenref:ObtainFromIDP

1473 The Discovery Service **SHOULD** mint new EPRs such that they carry the same identity context that was used to invoke
1474 the Discovery Service in the invocation context for the targeted WSP.

1475 When minting EPRs in response to a request, the Discovery Service:

- 1476 • **MUST** include in the EPR the matching (or all, if none were specified on the request) <Option> values contained
1477 within the <ServiceContext> of the <SvcMD> that matched the request criteria.
- 1478 • **SHOULD NOT** include any data from the <SvcMD> that was not specified in the request **and** is not necessary to
1479 create a useful EPR. For example, if the request specified a single <SecurityMechID> value, the resulting EPRs
1480 should not include other <SecurityMechID> elements, even if they are present in the <SvcMD> and otherwise
1481 could be specified in the same EPR.

1482 The Discovery Service **MAY** order <wsa:EndpointReference> elements as it sees fit. If the Discovery Service is
1483 rank ordering the entries, it **MUST** use descending rank order. This enables the requester to assume that if the results
1484 were ordered, the first result is the most relevant.

1485 The following rules specify the status code in the response:

1486 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code
1487 **MUST** be *Failed*.

1488 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* or *NoResults* as a second-level
1489 status code.

1490 The service may not wish to reveal the reason for failure, in which case no second-level status code will appear.

1491 **3.4. Operation: *MDAssociationAdd***

1492 The *MDAssociationAdd* operation is used by the WSP to add an association of the principal to the specified metadata.

1493 **3.4.1. *wsa:Action* values for *MDAssociationAdd* Messages**

1494 <SvcMDAssociationAdd> messages MUST include a <wsa:Action> SOAP header with the value of
 1495 "urn:liberty:disco:2006-08:SvcMDAssociationAdd".

1496 <SvcMDAssociationAddResponse> messages MUST include a <wsa:Action> SOAP header with the value of
 1497 "urn:liberty:disco:2006-08:SvcMDAssociationAddResponse".

1498 **3.4.2. *SvcMDAssociationAdd* Message**

1499 The <SvcMDAssociationAdd> is called with one or more <SvcMDID> elements to add associations to these service
 1500 metadata descriptions for the principal.

1501 A WSP SHOULD NOT associate the same <SvcMD> (or different SvcMD element that carry metadata for the "same"
 1502 service) to a principal multiple times without first removing the previous entry.

1503 The values in the <SvcMDID> element(s) must have been obtained via one of the service metadata operations discussed
 1504 later in this specification.

```

1505
1506 <!-- SvcMDAssociationAdd operation -->
1507
1508 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType" />
1509
1510 <xs:complexType name="SvcMDAssociationAddType">
1511   <xs:sequence>
1512     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
1513   </xs:sequence>
1514   <xs:anyAttribute namespace="##other" processContents="lax" />
1515 </xs:complexType>
1516

```

1517 **Figure 19. <SvcMDAssociationAdd> — Schema Fragment**

1518 An example message body containing a <SvcMDAssociationAdd> message follows. This request adds a new
 1519 association for the current principal (note that the identity of the principal is carried in the invocation context and not
 1520 in the body of the message).

```

1521
1522 <ds:SvcMDAssociationAdd>
1523   <ds:SvcMDID>2323872</ds:SvcMDID>
1524 </ds:SvcMDAssociationAdd>
1525

```

1526 **Example 7. <SvcMDAssociationAdd> Message**

1527 **3.4.3. *SvcMDAssociationAddResponse* Message**

1528 This response to the <SvcMDAssociationAdd> request contains the following elements and attributes.

- 1529 • <lu:Status>: Contains status code; see processing rules.

```

1530
1531 <!-- Response for SvcMDAssociationAdd operation -->
1532
1533 <xs:element name="SvcMDAssociationAddResponse"
1534           type="SvcMDAssociationAddResponseType"/>
1535
1536 <xs:complexType name="SvcMDAssociationAddResponseType">
1537   <xs:sequence>
1538     <xs:element ref="lu:Status" />
1539   </xs:sequence>
1540   <xs:anyAttribute namespace="##other" processContents="lax"/>
1541 </xs:complexType>
1542

```

1543 **Figure 20. <SvcMDAssociationAddResponse> — Schema Fragment**

```

1544
1545 <ds:SvcMDAssociationAddResponse>
1546   <lu:Status code="OK" />
1547 </ds:SvcMDAssociationAddResponse>
1548

```

1549 **Example 8. <SvcMDAssociationAddResponse> Message**

1550 3.4.4. MDAssociation Add Processing Rules

- 1551 • Once the association is added by the WSP, the Discovery Service MUST consider this metadata (subject to local
1552 policy) when responding to subsequent DiscoveryQuery operations and should the associated metadata meet the
1553 requirements of the query, mint the necessary ID-WSF EPRs based upon the requirements of the WSC and the
1554 WSP.
- 1555 • The Discovery Service SHOULD reject attempts to associate a <SvcMDID> that has already been associated with
1556 the principal by this WSP. In such cases the Discovery service MAY set the second level status code in the response
1557 to *Duplicate*.
- 1558 • The Discovery Service MAY similarly reject attempts to associate a <SvcMDID> that references the same service
1559 type and WSP that is in one of the already associated service metadata descriptions. In such cases the Discovery
1560 service MAY set the second level status code in the response to *LogicalDuplicate*.
- 1561 • The Discovery Service MUST reject attempts to associate a <SvcMDID> that does not exist or is not owned by the
1562 WSP invoking the call. In such cases the Discovery service MAY set the second level status code in the response
1563 to *NotFound*.
- 1564 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code
1565 MUST be *Failed*.
- 1566 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Duplicate*, *LogicalDuplicate*, or
1567 *NotFound* as a second-level status code. The Discovery Service instance may not wish to reveal the reason for
1568 failure, in which case no second-level status code will appear.
- 1569 • A Discovery Service MAY provide some programmatic or browser based interface which allows the principal to
1570 manage the service associations that have been added to their resource at the Discovery Service. A principal may
1571 be able to use such interfaces to change or even remove service associations made by the WSP without the WSP's
1572 permission (it is the principal's resource) and perhaps, even without notification to the WSP.
- 1573 Such interfaces are out-of-scope for this specification, but are mentioned here to remind the WSP that they may
1574 exist.

1575 **3.5. Operation: *MDAssociationQuery***

1576 The *MDAssociationQuery* operation is used by the WSP to query the Discovery Service for any previously added
 1577 associations related to the principal.

1578 **3.5.1. wsa:Action values for *MDAssociationQuery* Messages**

1579 <SvcMDAssociationQuery> messages MUST include a <wsa:Action> SOAP header with the value of
 1580 "urn:liberty:disco:2006-08:SvcMDAssociationQuery".

1581 <SvcMDAssociationQueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of
 1582 "urn:liberty:disco:2006-08:SvcMDAssociationQueryResponse".

1583 **3.5.2. SvcMDAssociationQuery Message**

1584 The <SvcMDAssociationQuery> is called with zero or more <SvcMDID> elements to query associations to these
 1585 service metadata descriptions. If no <SvcMDID> elements are specified, ALL associations between the WSP's service
 1586 metadata and the principal are returned.

```
1587
1588 <!-- SvcMDAssociationQuery operation -->
1589
1590 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType" />
1591
1592 <xs:complexType name="SvcMDAssociationQueryType" >
1593   <xs:sequence>
1594     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1595   </xs:sequence>
1596   <xs:anyAttribute namespace="##other" processContents="lax" />
1597 </xs:complexType>
1598
```

1599 **Figure 21. <SvcMDAssociationQuery> — Schema Fragment**

1600 An example message body containing a <SvcMDAssociationQuery> message follows. This request asks for all
 1601 associations.

```
1602
1603 <ds:SvcMDAssociationQuery />
1604
```

1605 **Example 9. <SvcMDAssociationQuery> Message**

1606 **3.5.3. SvcMDAssociationQueryResponse Message**

1607 This response to the <SvcMDAssociationQuery> request contains the following elements and attributes.

- 1608 • <lu:Status>: Contains status code; see processing rules.
- 1609 • <SvcMDID>: the associated service metadata ID(s). If <SvcMDID>s were specified on the
 1610 <SvcMDAssociationQuery> the response will be limited to at most those IDs (if they have been associ-
 1611 ated with the principal).

```

1612
1613 <!-- Response for SvcMDAssociationQuery operation -->
1614
1615 <xs:element name="SvcMDAssociationQueryResponse"
1616         type="SvcMDAssociationQueryResponseType" />
1617
1618 <xs:complexType name="SvcMDAssociationQueryResponseType">
1619     <xs:sequence>
1620         <xs:element ref="lu:Status" />
1621         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1622     </xs:sequence>
1623     <xs:anyAttribute namespace="##other" processContents="lax" />
1624 </xs:complexType>
1625

```

1626 **Figure 22. <SvcMDAssociationQueryResponse> — Schema Fragment**

```

1627
1628 <ds: SvcMDAssociationQueryResponse>
1629     <lu:Status code="OK" />
1630     <ds: SvcMDID>2323872</ds: SvcMDID>
1631 </ds: SvcMDAssociationQueryResponse>
1632

```

1633 **Example 10. <SvcMDAssociationQueryResponse> Message**

1634 3.5.4. MDAssociation Query Processing Rules

- 1635 • The Discovery Service MUST limit the operation to only those associations added by the WSP to the current
1636 principal's resource (a WSP MUST NOT be able to query associations added at the same Discovery Service by
1637 other WSPs or associations added to a different principal). There MUST NOT be any indication on the response
1638 as to whether or not other such elements exist.
- 1639 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code
1640 MUST be *Failed*.
- 1641 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NotFound* as a second-level
1642 status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
1643 second-level status code will appear.

1644 3.6. Operation: *MDAssociationDelete*

1645 The *MDAssociationDelete* operation is used by the WSP to delete a previously added association of the principal to
1646 the specified metadata.

1647 3.6.1. wsa:Action values for MDAssociationDelete Messages

1648 <SvcMDAssociationDelete> messages MUST include a <wsa:Action> SOAP header with the value of
1649 "urn:liberty:disco:2006-08:SvcMDAssociationDelete".

1650 <SvcMDAssociationDeleteResponse> messages MUST include a <wsa:Action> SOAP header with the value
1651 of "urn:liberty:disco:2006-08:SvcMDAssociationDeleteResponse".

1652 3.6.2. SvcMDAssociationDelete Message

1653 The <SvcMDAssociationDelete> is called with one or more <SvcMDID> elements to delete associations to these
1654 service metadata descriptions.

1655 Note that the service metadata description is not impacted by this call. Only the principal's association with the
1656 metadata is impacted.

```
1657
1658 <!-- SvcMDAssociationDelete operation -->
1659
1660 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType"/>
1661
1662 <xs:complexType name="SvcMDAssociationDeleteType">
1663   <xs:sequence>
1664     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
1665   </xs:sequence>
1666   <xs:anyAttribute namespace="##other" processContents="lax"/>
1667 </xs:complexType>
1668
```

1669 **Figure 23. <SvcMDAssociationDelete> — Schema Fragment**

1670 An example message body containing a <SvcMDAssociationDelete> message follows. This request deletes a
1671 single association for the current principal (note that the identity of the principal is carried in the invocation context
1672 and not in the body of the message).

```
1673
1674 <ds:SvcMDAssociationDelete>
1675   <ds:SvcMDID>2323872</ds:SvcMDID>
1676 </ds:SvcMDAssociationDelete>
1677
```

1678 **Example 11. <SvcMDAssociationDelete> Message**

1679 3.6.3. SvcMDAssociationDeleteResponse Message

1680 This response to the <SvcMDAssociationDelete> request contains the following elements and attributes.

- 1681 • <lu:Status>: Contains status code; see processing rules.

```
1682
1683 <!-- Response for SvcMDAssociationDelete operation -->
1684
1685 <xs:element name="SvcMDAssociationDeleteResponse"
1686   type="SvcMDAssociationDeleteResponseType"/>
1687
1688 <xs:complexType name="SvcMDAssociationDeleteResponseType">
1689   <xs:sequence>
1690     <xs:element ref="lu:Status" />
1691   </xs:sequence>
1692   <xs:anyAttribute namespace="##other" processContents="lax"/>
1693 </xs:complexType>
1694
```

1695 **Figure 24. <SvcMDAssociationDeleteResponse> — Schema Fragment**

```
1696
1697 <ds:SvcMDAssociationDeleteResponse>
1698   <lu:Status code="OK" />
1699 </ds:SvcMDAssociationDeleteResponse>
1700
```

1701 **Example 12. <SvcMDAssociationDeleteResponse> Message**

1702 3.6.4. MDAssociation Delete Processing Rules

- 1703 • Once deleted, the association **MUST NOT** be subsequently used by the DS to mint ID-WSF EPRs in response to
1704 queries relative to this principal. However, WSPs should be prepared to receive requests from WSCs from clients
1705 who previously obtained ID-WSF EPRs minted from the associaton which haven't expired.

- 1706 • The Discovery Service **MUST** limit the operation to only those associations added by the WSP to the current
1707 principal's resource (a WSP **MUST NOT** be able to delete associations added at the same Discovery Service by
1708 other WSPs or associations added to a different principal). There **MUST NOT** be any indication on the response
1709 as to whether or not other such elements exist.

- 1710 • The Discovery Service **MUST** treat attempts to delete non-existent associations as a successful no-op. This applies
1711 whether or not there are other existing associations being deleted in the same request (so a request to delete a single
1712 association that doesn't exist will succeed, even though the Discovery Service does not have to actually delete the
1713 record).

- 1714 • This operation **MUST** be atomic and if successful, all portions of the request **MUST** have succeeded. If any
1715 portion of the request fails, the entire request must fail.

- 1716 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code
1717 **MUST** be *Failed*.

- 1718 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* as a second-level status code. The
1719 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code
1720 will appear.

1721 **3.7. Operation: *MetadataRegister***

1722 The *MetadataRegister* operation is used to register a new service metadata description with the Discovery Service.

1723 **3.7.1. wsa:Action values for *MetadataRegister* Messages**

1724 <SvcMDRegister> messages **MUST** include a <wsa:Action> SOAP header with the value of
1725 "urn:liberty:disco:2006-08:SvcMDRegister".

1726 <SvcMDRegisterResponse> messages **MUST** include a <wsa:Action> SOAP header with the value of
1727 "urn:liberty:disco:2006-08:SvcMDRegisterResponse".

1728 **3.7.2. SvcMDRegister Message**

1729 The <SvcMDRegister> is called with one or more service metadata descriptions to be registered at the Discovery
1730 Service on behalf of the WSP.

```

1731
1732 <!-- Register operation for Service Metadata -->
1733
1734 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
1735
1736 <xs:complexType name="SvcMDRegisterType">
1737   <xs:sequence>
1738     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1739   </xs:sequence>
1740   <xs:anyAttribute namespace="##other" processContents="lax"/>
1741 </xs:complexType>
1742
1743

```

1744 **Figure 25. <SvcMDRegister> — Schema Fragment**

1745 An example message body containing a <SvcMDRegister> message follows. This request registers a new service
1746 metadata description. Note that the WSP has not set the svcMDID attribute on the <SvcMD> element – this will be
1747 assigned by the DS and returned in the response to the WSP.

```

1748
1749     <ds:SvcMDRegister>
1750         <ds:SvcMD>
1751             <ds:Abstract>Profile Service</ds:abstract>
1752             <ds:ProviderID>http://profile.com</ds:ProviderID>
1753             <ds:ServiceContext>
1754                 <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1755                 <ds:EndpointContext>
1756                     <ds:Address>https://profile.com/</ds:Address>
1757                     <sb:Framework version="2.0" />
1758                     <ds:SecurityMechID>
1759                         urn:liberty:security:2003-08:TLS:Bearer
1760                     </ds:SecurityMechID>
1761                 </ds:EndpointContext>
1762             </ds:ServiceContext>
1763         </ds:SvcMD>
1764     </ds:SvcMDRegister>
1765

```

1766 **Example 13. <SvcMDRegister> Message**

1767 3.7.3. SvcMDRegisterResponse Message

1768 This response to the <SvcMDRegister> request contains the following elements and attributes.

- 1769 • <lu:Status>: Contains status code; see processing rules.
- 1770 • One or more <SvcMDID> if the call was successful (status code is OK). One SvcMDID is returned for each service
1771 metadata element registered.
- 1772 • <Keys>: Contains the key descriptors for the keys used by the Discovery Service to sign security tokens (see
1773 [Section 3.12](#) for a description of when and why this may be necessary).

```

1774
1775 <!-- Response for SvcMDRegister operation -->
1776
1777 <xs:element name="SvcMDRegisterResponse"
1778     type="SvcMDRegisterResponseType" />
1779
1780 <xs:complexType name="SvcMDRegisterResponseType">
1781     <xs:sequence>
1782
1783         <xs:element ref="lu:Status" />
1784         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1785         <xs:element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
1786
1787     </xs:sequence>
1788     <xs:anyAttribute namespace="##other" processContents="lax" />
1789 </xs:complexType>
1790
1791

```

1792 **Figure 26. <SvcMDRegisterResponse> — Schema Fragment**

```
1793
1794 <ds:SvcMDRegisterResp>
1795   <lu:Status code="OK" />
1796   <ds:SvcMDID>2323872</ds:SvcMDID>
1797 </ds:SvcMDRegister>
1798
```

1799 **Example 14. <SvcMDRegisterResponse> Message**

1800 3.7.4. Metadata Register Processing Rules

- 1801 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
1802 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.
- 1803 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use
1804 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the
1805 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.
- 1806 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The
1807 Discovery Service **MUST** enforce this atomicity.
- 1808 • For each <SvcMD> element, the Discovery Service instance **MAY** store the metadata provided such that it can be
1809 used (subject to policy) to mint ID-WSF EPRs in response to future *DiscoveryQuery* operations should that service
1810 metadata be associated with a principal's resource at the Discovery Service.
- 1811 If the Discovery Service instance does not store the metadata, it **MUST** return a *Failed* status code for the operation,
1812 and therefore not register any of the other entries provided.
- 1813 If the Discovery Service does store the metadata, it **MUST** assign a permanent identifier for the metadata usable
1814 by the WSP to subsequently reference the metadata. This identifier **MUST** be unique across all metadata objects
1815 stored by a WSP and **MAY** be unique across all metadata objects stored by all WSPs at that Discovery Service.
1816 This identifier is provided to the WSP in the response and can be subsequently used by the WSP to associate this
1817 metadata with a principal or to manage the metadata using one of the other metadata operations.
- 1818 • A WSP **MAY** register multiple service metadata descriptions that for all intents and purposes, appear to be fully
1819 equal. The Discovery Service **MUST NOT** generate an error solely because it thinks the descriptions are equal.
1820 The Discovery Service **MUST** treat these records as independent registrations and assign the associated unique
1821 SvcMDID values.
- 1822 • The Discovery Service **MAY** have some policy driven limit on the number of service metadata descriptions that it
1823 will allow a WSP to register. If a WSP attempts to register a new service metadata description that would exceed
1824 such a limit, the DS **SHOULD** include a secondary-level status code of *LimitExceeded*.
- 1825 A WSP should exercise care to only register new service metadata descriptions when an existing, registered,
1826 description that meets the WSP's needs is not available.
- 1827 • The Discovery Service **SHOULD** validate that a given SvcMD only contains entries for a single logical service (i.e.
1828 allow for different versions, but **not** allow differences in the basic service type). For example, a single SvcMD
1829 **SHOULD** not contain data for both a contact book service and a calendar service.
- 1830 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.
- 1831 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the registration request and **MAY**
1832 include a secondary-level status code of *Invalid*.
- 1833 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code
1834 **MUST** be *Failed*.

- 1835 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Invalid*, or *OverLimit* as a second-
1836 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
1837 second-level status code will appear.

1838 3.8. Operation: *MetadataQuery*

1839 The *MetadataQuery* operation is used to query the Discovery Service for existing, registered, service metadata
1840 descriptions.

1841 3.8.1. *wsa:Action* values for *MetadataQuery* Messages

1842 <SvcMDQuery> messages MUST set the value of the <wsa:Action> header to "urn:liberty:disco:2006-08:SvcMDQuery".

1843 <SvcMDQueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1844 "urn:liberty:disco:2006-08:SvcMDQueryResponse".

1845 3.8.2. *SvcMDQuery* Message

1846 The <SvcMDQuery> is called with zero or more <SvcMDID> elements to retrieve the specified list of service metadata
1847 descriptions. If no <SvcMDID>s are specified, ALL of the metadata stored at the Discovery service by the invoking
1848 WSP will be returned.

```
1849 <!-- Query operation on Service Metadata -->
1850
1851 <xs:element name="SvcMDQuery" type="SvcMDQueryType"/>
1852
1853 <xs:complexType name="SvcMDQueryType">
1854   <xs:sequence>
1855     <xs:element ref="SvcMDID"
1856       minOccurs="0"
1857       maxOccurs="unbounded"/>
1858   </xs:sequence>
1859   <xs:anyAttribute namespace="##other" processContents="lax"/>
1860 </xs:complexType>
1861
1862
1863
```

1864 **Figure 27. <SvcMDQuery> — Schema Fragment**

1865 An example message body containing a <SvcMDQuery> message follows. This request queries for a specific service
1866 metadata description by providing the ID of the desired metadata in the <SvcMDID> element.

```
1867 <ds: SvcMDQuery>
1868   <ds: SvcMDID>2323872</ds: SvcMDID>
1869 </ds: SvcMDQuery>
1870
1871
```

1872 **Example 15. <SvcMDQuery> Message**

1873 3.8.3. *SvcMDQueryResponse* Message

1874 This response to the <SvcMDQuery> request contains the following elements and attributes.

- 1875 • <lu:Status>: Contains status code; see processing rules.
- 1876 • One or more <SvcMD> elements if the call was successful (status code is OK).

```

1877
1878 <!-- Response for Query operation on Service Metadata -->
1879
1880 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType" />
1881
1882 <xs:complexType name="SvcMDQueryResponseType">
1883   <xs:sequence>
1884     <xs:element ref="lu:Status" />
1885     <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
1886   </xs:sequence>
1887   <xs:anyAttribute namespace="##other" processContents="lax" />
1888 </xs:complexType>
1889
1890

```

1891 **Figure 28. <SvcMDQueryResponse> — Schema Fragment**

```

1892
1893 <ds:SvcMDQueryResponse>
1894   <lu:Status code="OK" />
1895   <ds:SvcMD svcMDID="2323872" >
1896     <ds:Abstract>Profile Service</ds:Abstract>
1897     <ds:ProviderID>http://profile.com</ds:ProviderID>
1898     <ds:ServiceContext>
1899       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1900       <ds:EndpointContext>
1901         <ds:Address>https://profile.com/</ds:Address>
1902         <sb:Framework version="2.0" />
1903         <ds:SecurityMechID>
1904           urn:liberty:security:2003-08:TLS:Bearer
1905         </ds:SecurityMechID>
1906       </ds:EndpointContext>
1907     </ds:ServiceContext>
1908   </ds:SvcMD>
1909 </ds:SvcMDQueryResponse>
1910

```

1911 **Example 16. <SvcMDQueryResponse> Message**

1912 3.8.4. Metadata Query Processing Rules

- 1913 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
1914 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.
- 1915 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use
1916 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the
1917 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.
- 1918 • The Discovery Service **MUST** limit the results to only those metadata elements stored by the WSP (a WSP **MUST**
1919 **NOT** be able to retrieve metadata elements stored at the same Discovery Service by other WSPs). There **MUST**
1920 **NOT** be any indication on the response as to whether or not other such elements exist.
- 1921 • The Discovery Service **SHOULD** treat a request that matches a subset of the `svcMDID` values specified in the
1922 request as a successful request returning the entries that were found and nothing for the missing entries. The
1923 WSP will be able to distinguish which entries were found by examining the `svcMDID` attribute on the `<svcMD>`
1924 element(s) in the response.
- 1925 • If request processing succeeded **AND** results are returned, the top-level status code **MUST** be *OK*. Otherwise, the
1926 top-level status code **MUST** be *Failed*.

1927 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NoResults* as a second-level
 1928 status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
 1929 second-level status code will appear.

1930 3.9. Operation: *MetadataReplace*

1931 The *MetadataReplace* operation is used by a WSP to replace previously stored metadata in the Discovery Service.
 1932 This is how the WSP updates their metadata without having to reassociate with a principal.

1933 3.9.1. *wsa:Action* values for *MetadataReplace* Messages

1934 <SvcMDReplace> messages MUST include a <wsa:Action> SOAP header with the value of
 1935 "urn:liberty:disco:2006-08:SvcMDReplace".

1936 <SvcMDReplaceResponse> messages MUST include a <wsa:Action> SOAP header with the value of
 1937 "urn:liberty:disco:2006-08:SvcMDReplaceResponse".

1938 3.9.2. *SvcMDReplace* Message

1939 The <SvcMDReplace> is called with one or more replacement <SvcMD> elements each of which must include the
 1940 svcMDID attribute set to the ID of the respective metadata element they are to replace.

```

1941
1942 <!-- Replace operation on Service Metadata -->
1943
1944 <xs:element name="SvcMDReplace" type="SvcMDReplaceType" />
1945
1946 <xs:complexType name="SvcMDReplaceType">
1947   <xs:sequence>
1948     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1949   </xs:sequence>
1950   <xs:anyAttribute namespace="##other" processContents="lax" />
1951 </xs:complexType>
1952
1953
  
```

1954 **Figure 29. <SvcMDReplace> — Schema Fragment**

1955 An example message body containing a <SvcMDReplace> message follows. This request replaces an existing
 1956 metadata element to update the endpoint for the service.

```

1957
1958 <ds:SvcMDReplace>
1959   <ds:SvcMD svcMDID="2323872" >
1960     <ds:Abstract>Profile Service</ds:abstract>
1961     <ds:ProviderID>http://profile.com</ds:ProviderID>
1962     <ds:ServiceContext>
1963       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1964       <ds:EndpointContext>
1965         <ds:Address>https://newaddr.com/</ds:Address>
1966         <sb:Framework version="2.0" />
1967         <ds:SecurityMechID>
1968           urn:liberty:security:2003-08:TLS:Bearer
1969         </ds:SecurityMechID>
1970       </ds:EndpointContext>
1971     </ds:ServiceContext>
1972   </ds:SvcMD>
1973 </ds:SvcMDReplace>
1974
  
```

1975 **Example 17. <SvcMDReplace> Message**

1976 **3.9.3. SvcMDReplaceResponse Message**

1977 This response to the <SvcMDReplace> request contains the following elements and attributes.

- 1978 • <lu:Status>: Contains status code; see processing rules.

```

1979
1980 <!-- Response for SvcMDReplace operation -->
1981
1982 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType"/>
1983
1984 <xs:complexType name="SvcMDReplaceResponseType" >
1985   <xs:sequence>
1986     <xs:element ref="lu:Status" />
1987   </xs:sequence>
1988   <xs:anyAttribute namespace="##other" processContents="lax"/>
1989 </xs:complexType>
1990
1991

```

1992 **Figure 30. <SvcMDReplaceResponse> — Schema Fragment**

```

1993
1994 <ds:SvcMDReplaceResponse>
1995   <lu:Status code="OK" />
1996 </ds:SvcMDReplaceResponse>
1997

```

1998 **Example 18. <SvcMDReplaceResponse> Message**

1999 **3.9.4. Metadata Replace Processing Rules**

- 2000 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
 2001 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.
- 2002 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use
 2003 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the
 2004 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.
- 2005 • The Discovery Service **MUST** limit the operation to only those metadata elements stored by the WSP (a WSP
 2006 **MUST NOT** be able to replace metadata elements stored at the same Discovery Service by other WSPs). There
 2007 **MUST NOT** be any indication on the response as to whether or not other such elements exist.
- 2008 • The Discovery Service **SHOULD** validate that the replacement SvcMD contains the same logical service as the
 2009 original SvcMD. By "logical service" we mean to allow for different versions, but **not** allow differences in the
 2010 basic service type. For example, a calendar service SvcMD **SHOULD** not be allowed to replace a contact book
 2011 service SvcMD.
- 2012 The Discovery Service **SHOULD** also validate that the replacement SvcMD only contains entries for a single
 2013 logical service (as described above). For example, a single SvcMD **SHOULD** not contain data for both a contact
 2014 book service and a calendar service.
- 2015 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.
- 2016 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the replacement request and **MAY**
 2017 include a secondary-level status code of *Invalid*.

- 2018 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The
2019 Discovery Service MUST enforce this atomicity.
- 2020 • Once replaced, the previous service metadata element MUST NOT be subsequently used by the DS to mint ID-
2021 WSF EPRs. However, WSPs should be prepared to receive requests from WSCs from clients who previously
2022 obtained ID-WSF EPRs minted from the prior metadata which haven't expired.
- 2023 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code
2024 MUST be *Failed*.
- 2025 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Invalid*, or *NotFound* as a second-
2026 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no
2027 second-level status code will appear.

2028 **3.10. Operation: *MetadataDelete***

2029 The *MetadataDelete* operation is used by the WSP to delete previously registered metadata elements in the Discovery
2030 Service.

2031 **3.10.1. wsa:Action values for *MetadataDelete* Messages**

2032 <SvcMDDelete> messages MUST include a <wsa:Action> SOAP header with the value of
2033 "urn:liberty:disco:2006-08:SvcMDDelete".

2034 <SvcMDDeleteResponse> messages MUST include a <wsa:Action> SOAP header with the value of
2035 "urn:liberty:disco:2006-08:SvcMDDeleteResponse".

2036 **3.10.2. SvcMDDelete Message**

2037 The <SvcMDDelete> is called with one or more <SvcMDID> elements to delete the specified list of service metadata
2038 descriptions.

```
2039 <!-- Delete operation on Service Metadata -->
2040
2041
2042 <xs:element name="SvcMDDelete" type="SvcMDDeleteType"/>
2043
2044 <xs:complexType name="SvcMDDeleteType">
2045   <xs:sequence>
2046     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
2047   </xs:sequence>
2048   <xs:anyAttribute namespace="##other" processContents="lax"/>
2049 </xs:complexType>
2050
2051
```

2052 **Figure 31. <SvcMDDelete> — Schema Fragment**

2053 An example message body containing a <SvcMDDelete> message follows. This request deletes a single service
2054 metadata description.

```
2055 <ds: SvcMDDelete>
2056   <ds: SvcMDID>2323872</ds: SvcMDID>
2057 </ds: SvcMDDelete>
2058
2059
```

2060 **Example 19. <SvcMDDelete> Message**

2061 3.10.3. SvcMDDeleteResponse Message

2062 This response to the <SvcMDDelete> request contains the following elements and attributes.

- 2063 • <lu:Status>: Contains status code; see processing rules.

```

2064
2065 <!-- Response for delete operation on Service Metadata -->
2066
2067 <xs:element name="SvcMDDeleteResponse" type="SvcMDDeleteResponseType" />
2068
2069 <xs:complexType name="SvcMDDeleteResponseType">
2070   <xs:sequence>
2071     <xs:element ref="lu:Status" />
2072   </xs:sequence>
2073   <xs:anyAttribute namespace="##other" processContents="lax" />
2074 </xs:complexType>
2075
2076

```

2077 **Figure 32. <SvcMDDeleteResponse> — Schema Fragment**

```

2078
2079 <ds:SvcMDDeleteResponse>
2080   <lu:Status code="OK" />
2081 </ds:SvcMDDeleteResponse>
2082

```

2083 **Example 20. <SvcMDDeleteResponse> Message**

2084 3.10.4. Metadata Delete Processing Rules

- 2085 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that
2086 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.
- 2087 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use
2088 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the
2089 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.
- 2090 • If the service metadata being deleted is still associated with one or more principals, the Discovery Service **MUST**
2091 automatically remove such associations (i.e. the delete of metadata cascades to delete the associations).
- 2092 • Once deleted, the service metadata element **MUST NOT** be subsequently used by the DS to mint ID-WSF EPRs.
2093 However, WSPs should be prepared to receive requests from WSCs from clients who previously obtained ID-WSF
2094 EPRs minted from the metadata which haven't expired.
- 2095 • The Discovery Service **MUST** limit the operation to only those metadata elements stored by the WSP (a WSP
2096 **MUST NOT** be able to delete metadata elements stored at the same Discovery Service by other WSPs). There
2097 **MUST NOT** be any indication on the response as to whether or not other such elements exist.
- 2098 • The Discovery Service **MUST** treat attempts to delete non-existent metadata elements as a successful no-op. This
2099 applies whether or not there are other existing metadata elements being deleted in the same request (so a request to
2100 delete a single metadata element that doesn't exist will succeed, even though the Discovery Service does not have
2101 to actually delete the record).
- 2102 • This operation **MUST** be atomic and if successful, all portions of the request **MUST** have succeeded. If any
2103 portion of the request fails, the entire request must fail.

2104 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code
2105 MUST be *Failed*.

2106 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* as a second-level status code. The
2107 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code
2108 will appear.

2109 **3.11. option Value for Response Authentication**

2110 The ID-WSF EPR `<SecurityContext>` element provides a way for services to indicate to clients what mecha-
2111 nisms are necessary for the client to authenticate itself to the service via the `<SecurityMechID>` element. The
2112 `<SecurityMechID>` values defined by [LibertySecMech] also indicate whether the service uses peer entity authenti-
2113 cation (for example, server-side SSL/TLS). However, a web service client may need to know whether the service will
2114 use message authentication (that is, whether the service will sign the response message) and may not be willing to use
2115 a service which does not sign its responses.

2116 To avoid situations where a client requests data and then discovers it does not trust it because it is not signed, an
2117 `<Option>` value is defined:

2118 `urn:liberty:disco:2006-08:options:security-response-x509`

2119 If a service instance always authenticates its response messages according to the "X.509 v3 Certificate Message
2120 Authentication" mechanism in [LibertySecMech], registrations of ID-WSF EPRs describing the service instance
2121 SHOULD include this option value. Otherwise, its registered ID-WSF EPRs MUST NOT include this option value.
2122 Clients MAY include this option value in `<Query>` messages in order to locate only services which always authenticate
2123 their response messages. A service MAY authenticate its response messages even if this option value was not included
2124 in its description at the Discovery Service instance.

2125 In case the service also supports a previous version of the security mechanism specification [LibertySecMech11],
2126 it should be able to register two different endpoints at the Discovery Service, each of them with different Options
2127 values—one according to [LibertySecMech], the other one according to [LibertySecMech11]. This information
2128 will aid the client in determining which version of the WSS-SMS specification ([wss-sms-draft] and/or [wss-sms]) is
2129 supported by the service, and the service will act accordingly, depending on the ID-WSF EPR used by the client. Note
2130 that this behavior only applies to the case when the client's request does not use message authentication mechanisms.

2131 Otherwise, it should be possible for the service to determine the version of the WSS-SMS specification supported by
2132 the client by simply analyzing the `<wsse:Security>` header present in the request.

2133 In general, it is recommended that services do not sign their responses unless they positively know that clients are able
2134 to perform message authentication and are aware of the version of the WSS-SMS spec used by that client.

2135 **3.12. Including Keys in the ModifyResponse Message**

2136 The Discovery Service instance may need to generate signed security tokens in `<QueryResponse>` messages for the
2137 ID-WSF EPRs in question (which are later included in a message to a WSP). The WSP which receives the signed
2138 security tokens from a client needs to be able to verify the Discovery service instance's signature on the security tokens.
2139 Typically the metadata (see [SAMLMeta2]) for the Discovery service instance is sufficient for such information. In
2140 certain situations, such as when the Discovery service instance is hosted on a LUAD (see [LibertyClientProfiles]), it
2141 may not be feasible to assign the LUAD a ProviderID with which to obtain metadata. However, the key material
2142 still needs to be made available to service instances which register ID-WSF EPRs with the Discovery Service which
2143 include security mechanisms requiring such tokens.

2144 The Discovery Service instance may include a `<Keys>` element in the `<ModifyResponse>` in order to provide such
2145 keys.

2146 The Discovery Service instance **SHOULD ONLY** include the <Keys> element in <ModifyResponse> messages if
2147 it has no <ProviderID> and the <Modify> message included an ID-WSF EPR for which the Discovery Service
2148 instance intends to generate signed security tokens.

```
2149  
2150 <!-- Keys Element - For use in ModifyResponse -->  
2151  
2152 <xs:element name="Keys" type="KeysType"/>  
2153  
2154 <xs:complexType name="KeysType">  
2155   <xs:sequence>  
2156     <xs:element ref="md:KeyDescriptor"  
2157       minOccurs="1"  
2158       maxOccurs="unbounded"/>  
2159   </xs:sequence>  
2160 </xs:complexType>  
2161  
2162
```

2163 **Figure 33. <Keys> — Schema Fragment**

2164 The <Keys> element appears as a child of the <ModifyResponse> element. It contains one or more
2165 <KeyDescriptor> elements.

2166 **3.13. Discovery Service Example Messages (NON-Normative)**

2167 This section walks through a series of messages to show examples of inputs and outputs. The information in this
2168 section is **NOT** normative with respect to the specification (in cases where the other normative section(s) of the
2169 specification conflict with this section, the normative section(s) will prevail).

2170 Notes about this sequence:

- 2171 • It is an actual network capture of a real session between two independent liberty implementations.
- 2172 • The messages are from a **test sequence** which exercises the features of the Discovery Service. We do **not** expect
2173 that any real world situation would result in this sequence of messages (or anything similar to this sequence of
2174 messages).
- 2175 • All of these messages were invoked using the same SAML Assertion to establish the invocation context with the
2176 principal as the subject and the WSP in the subject confirmation. Note that some of the processing rules for the
2177 Discovery Service interfaces (those that manage the service metadata) require that such an invocation context be
2178 interpreted as a WSP invoker invocation context.
- 2179 • The initial state is that the principal has a single service associated with their identity (an instance of the ID-WSF
2180 People Service).
- 2181 • The messages **are** part of a sequence that was invoked in this order and one request (such as a new registration)
2182 can and usually does impact the results of subsequent requests.
- 2183 • The first request and response messages are shown as a full SOAP-bound ID-* message (with all of the SOAP
2184 headers). The remaining request and response messages are shown with just the ID-* message component (i.e.
2185 only the contents within the <S:Body> element are shown).
- 2186 • The messages have been formatted for easier reading (pretty-printing) and data has been ellided (such as the
2187 contents of SAML assertions) for brevity.
- 2188 • Many of the service types used in these example messages do **not** exist as real ID-* services and are only used
2189 here as testing input to exercise the Discovery Service.

2190 3.13.1. Query People Service

2191 Query for the People Service

```

2192 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
2193     xmlns:wss="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd"
2194     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2195     xmlns:sb2="urn:liberty:sb:2006-08"
2196     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2197     xmlns:wsa="http://www.w3.org/2005/08/addressing">
2198 <S:Header>
2199     <wsa:MessageID S:mustUnderstand="true"
2200         S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="msgHdrID">
2201         uuid:asdqwer-238asf-44353608-000b8c14
2202     </wsa:MessageID>
2203     <wsa:To S:mustUnderstand="true"
2204         S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaToID">
2205         https://s-ds.liberty-iop.org:8681/DISCO-S
2206     </wsa:To>
2207     <wsa:Action S:mustUnderstand="true"
2208         S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaActionID">
2209         urn:liberty:disco:2006-08:Query
2210     </wsa:Action>
2211     <wsse:Security S:mustUnderstand="true"
2212         S:actor="http://schemas.xmlsoap.org/soap/actor/next">
2213         <sa:Assertion
2214             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2215             ID="CRED6q6HqDRRCAPsq3L8d_sh"
2216             IssueInstant="2006-04-06T15:39:12Z"
2217             Version="2.0">
2218             ... assertion data was here ...
2219         </sa:Assertion>
2220         <wsu:Timestamp wsu:Id="WsuTimestampID">
2221             <wsu:Created>2006-04-06T15:38:48Z</wsu:Created>
2222         </wsu:Timestamp>
2223     </wsse:Security>
2224 </S:Header>
2225 <S:Body wsu:id="msgBodyID">
2226     <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2227         <disco:RequestedService>
2228             <disco:ServiceType>urn:liberty:ps:2006-08</disco:ServiceType>
2229             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2230             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2231         </disco:RequestedService>
2232     </disco:Query>
2233 </S:Body>
2234 </S:Envelope>

```

2235 Things to note about this query:

- 2236 • it is a query of a service type (in this case, a particular version of the ID-WSF People Service)
- 2237 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for
- 2238 communicating with the specified service

2239 The response from the Discovery Service:

```

2240 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
2241     xmlns:lib="urn:liberty:iff:2003-08">
2242 <soap:Header>
2243     <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing" id="MID">
2244         uuid:IKvqaaaE0dZ5Hhg1IQZ1
2245     </wsa:MessageID>
2246     <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
2247         uuid:asdqwer-238asf-44353608-000b8c14
2248     </wsa:RelatesTo>

```

```

2249 <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
2250   urn:liberty:disco:2006-08:QueryResponse
2251 </wsa:Action>
2252 <wsa:ReplyTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
2253   <wsa:Address>
2254     http://www.w3.org/2005/03/addressing/role/anonymous
2255   </wsa:Address>
2256 </wsa:ReplyTo>
2257 <wsse:Security xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd">
2258   <wsu:Timestamp xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd">
2259     <wsu:Created>2006-04-06T15:39:13Z</wsu:Created>
2260   </wsu:Timestamp>
2261 </wsse:Security>
2262 <sb2:Sender xmlns:sb2="urn:liberty:sb:2006-08"
2263   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2264   providerID="https://s-ds.liberty-iop.org:8681/idp.xml"
2265   soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
2266   wsu:Id="PRV" />
2267 <sb2:Framework
2268   xmlns:sb2="urn:liberty:sb:2006-08"
2269   version="2.0"
2270   soap:actor="http://schemas.xmlsoap.org/soap/actor/next" />
2271 </soap:Header>
2272 <soap:Body>
2273   <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2274     <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK" />
2275     <wsa:EndpointReference
2276       xmlns:wsa="http://www.w3.org/2005/08/addressing"
2277       xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2278       notOnOrAfter="2006-04-06T17:39:14Z"
2279       wsu:Id="EPRIDIo61485WDEA701qHi4Ey">
2280     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2281     <wsa:Metadata>
2282       <disco:Abstract>SYmfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2283       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0" />
2284       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2285       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2286       <disco:SecurityContext>
2287         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2288         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2289           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2290           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2291             ID="CREDI4cINqS4SV1HPm_xOoGh"
2292             IssueInstant="2006-04-06T15:39:14Z"
2293             Version="2.0">
2294             ... assertion data was here ...
2295           </sa:Assertion>
2296         </sec:Token>
2297       </disco:SecurityContext>
2298     </wsa:Metadata>
2299   </wsa:EndpointReference>
2300 </disco:QueryResponse>
2301 </soap:Body>
2302 </soap:Envelope>

```

2303 Things to note about this response:

- 2304 • the query was successful (status code is OK).
- 2305 • there is a single ID-WSF EPR in the response for the service type specified in the request
- 2306 • the details within the assertion were edited out to save space

2307 3.13.2. Query provider

2308 Query for the same service using the ProviderID of the WSP.

```
2309 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2310 <disco:RequestedService>
2311   <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2312   <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2313   <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2314 </disco:RequestedService>
2315 </disco:Query>
```

2316 Things to note about this query:

- 2317 • it is a query of a ProviderID, so all services provided by that ProviderID would be returned (and depending upon
2318 the invocation context of the ID-WSF framework, this may be all services provided by that provider for a particular
2319 principal or just all services).
- 2320 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for
2321 communicating with the specified service

2322 Server Response:

```
2323 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2324 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2325 <wsa:EndpointReference
2326   xmlns:wsa="http://www.w3.org/2005/08/addressing"
2327   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2328   notOnOrAfter="2006-04-06T17:39:18Z"
2329   wsu:Id="EPRIDyGxcYpQ8Gace5y8lDm7Z">
2330 <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2331 <wsa:Metadata>
2332 <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2333 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2334 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2335 <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2336 <disco:SecurityContext>
2337 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2338 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2339   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2340 <sa:Assertion
2341   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2342   ID="CREDOEV7U7-mJk02pEVJAn--"
2343   IssueInstant="2006-04-06T15:39:18Z"
2344   Version="2.0">
2345   ... assertion data was here ...
2346 </sa:Assertion>
2347 </sec:Token>
2348 </disco:SecurityContext>
2349 </wsa:Metadata>
2350 </wsa:EndpointReference>
2351 </disco:QueryResponse>
```

2352 Things to note about this response:

- 2353 • the query was successful (status code is OK).
- 2354 • there is a single ID-WSF EPR in the response denoting the single service offered by the provider specified in the
2355 request for this user.
- 2356 • The ID-WSF EPR is **almost** identical to the ID-WSF EPR returned in the previous request. The differences are
2357 only in the timestamps and the element IDs.

2358 3.13.3. Query (empty)

2359 A Query without specifying any search criteria (which should return all services available to the user).

2360 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq"/>

2361 Things to note about this query:

- 2362 • you still need to include the <disco:Query> element in the request, just that its contents are empty.

2363 Server Response:

```
2364 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2365 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2366 <wsa:EndpointReference
2367   xmlns:wsa="http://www.w3.org/2005/08/addressing"
2368   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2369   notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDt3MOElDHPL0EBD8whEvw">
2370 <wsa:Address>https://s-ds.liberty-iop.org:8681/DISCO-S</wsa:Address>
2371 <wsa:Metadata>
2372 <disco:Abstract>SYMfiam Discovery Service</disco:Abstract>
2373 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2374 <disco:ProviderID>https://s-ds.liberty-iop.org:8681/idp.xml</disco:ProviderID>
2375 <disco:ServiceType>urn:liberty:disco:2006-08</disco:ServiceType>
2376 <disco:SecurityContext>
2377 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2378 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2379   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2380 <sa:Assertion
2381   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2382   ID="CREDr2tN6rxYICAXxD6CtenC"
2383   IssueInstant="2006-04-06T15:39:21Z" Version="2.0">
2384   ... assertion data was here ...
2385 </sa:Assertion>
2386 </sec:Token>
2387 </disco:SecurityContext>
2388 </wsa:Metadata>
2389 </wsa:EndpointReference>
2390 <wsa:EndpointReference
2391   xmlns:wsa="http://www.w3.org/2005/08/addressing"
2392   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2393   notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDQBqOfWHDP3GKFSqqnZyj">
2394 <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2395 <wsa:Metadata>
2396 <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2397 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2398 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2399 <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2400 <disco:SecurityContext>
2401 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2402 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2403   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2404 <sa:Assertion
2405   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2406   ID="CREDAsw9RdingIu3m4HtUxX5"
2407   IssueInstant="2006-04-06T15:39:22Z" Version="2.0">
2408   ... assertion data was here ...
2409 </sa:Assertion>
2410 </sec:Token>
2411 </disco:SecurityContext>
2412 </wsa:Metadata>
2413 </wsa:EndpointReference>
2414 </disco:QueryResponse>
```

2415 Things to note about this response:

- 2416 • the query was successful (status code is OK).
- 2417 • two ID-WSF EPRs were returned, one for the Discovery Service and one for the People Service.
- 2418 • Discovery Service instances will usually expose ID-WSF EPRs which point to themselves in this way in order to
- 2419 expose alternative invocation methods and/or allow the client to obtain newer credentials.

2420 3.13.4. Query People Service and Provider

2421 A discovery query specifying both the Provider ID and the Service Type (so that only services of that type by that

2422 provider are returned)

```
2423 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2424   <disco:RequestedService>
2425     <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2426     <disco:ProviderID>https://s-wsp.1.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2427     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2428     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2429   </disco:RequestedService>
2430 </disco:Query>
```

2431 Things to note about this query:

- 2432 • both the <disco:ServiceType> element and the <disco:ProviderID> element are included.
- 2433 • The provider and service type are the same ones we've been using so we should get the same response we
- 2434 had on earlier requests.

2435 Server Response:

```
2436 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2437   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2438   <wsa:EndpointReference
2439     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2440     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2441     notOnOrAfter="2006-04-06T17:39:26Z" wsu:Id="EPRIDwrdfxWpBRhCXsEMW1cjo">
2442     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2443     <wsa:Metadata>
2444       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2445       <sbfi:Framework xmlns:sbfi="urn:liberty:sbfi" version="2.0"/>
2446       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2447       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2448       <disco:SecurityContext>
2449         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2450         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2451           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2452           <sa:Assertion
2453             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2454             ID="CREDZbnmfBjqRINmQeWvyxCG"
2455             IssueInstant="2006-04-06T15:39:26Z" Version="2.0">
2456             ... assertion data was here ...
2457           </sa:Assertion>
2458         </sec:Token>
2459       </disco:SecurityContext>
2460     </wsa:Metadata>
2461   </wsa:EndpointReference>
2462 </disco:QueryResponse>
```

2463 Things to note about this response:

- 2464 • the query was successful (status code is OK).

- 2465 • The ID-WSF EPR is **almost** exactly the same as the ID-WSF EPR returned in the first 2 examples above.
2466 The differences are only in the timestamps and the element IDs. This is because both EPRs are for the same
2467 principal accessing the same service (we just discovered the EPR through a different query).

2468 3.13.5. SvcMDQuery (empty)

2469 A query for all SvcMD stored at the DS on behalf of the invoking provider.

```
2470 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08"/>
```

2471 Things to note about this query:

- 2472 • it's empty – meaning that all registered SvcMDs are requested.
2473 • the query is executed in the context of the provider invoking the query (identified in the invocation context specified
2474 in the ID-WSF framework headers)

2475 Server Response:

```
2476 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">  
2477 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>  
2478 <disco:SvcMD svcMDID="SVCMDIDy_7yFABwuFPkgYIjKCjv">  
2479 <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>  
2480 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>  
2481 <disco:ServiceContext>  
2482 <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>  
2483 <disco:EndpointContext>  
2484 <disco:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</disco:Address>  
2485 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>  
2486 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>  
2487 </disco:EndpointContext>  
2488 </disco:ServiceContext>  
2489 </disco:SvcMD>  
2490 </disco:SvcMDQueryResponse>
```

2491 Things to note about this response:

- 2492 • the query was successful (status code is OK).
2493 • one SvcMD was returned for the ID-WSF People Service (which is the only SvcMD registered by the invoking
2494 provider).

2495 3.13.6. SvcMDQuery w/Bad SvcMDID

2496 A service metadata query using an invalid SvcMDID.

```
2497 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">  
2498 <disco:SvcMDID>123</disco:SvcMDID>  
2499 </disco:SvcMDQuery>
```

2500 Things to note about this query:

- 2501 • 123 is an invalid SvcMDID

2502 Server Response:

```
2503 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2504   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2505     <lu:Status code="NoResults"/>
2506   </lu:Status>
2507 </disco:SvcMDQueryResponse>
```

2508 Things to note about this response:

- 2509 • the query failed (status code is Failed).
- 2510 • The optional sub-status is included which indicates that there were *NoResults* for the query.

2511 3.13.7. SvcMDRegister w/single SvcMD

2512 Registration of a single service metadata instance in the DS.

```
2513 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2514   <disco:SvcMD>
2515     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2516     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2517     <disco:ServiceContext>
2518       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2519       <disco:EndpointContext>
2520         <disco:Address>https://payment.testing.com</disco:Address>
2521         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2522         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2523         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2524         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2525       </disco:EndpointContext>
2526     </disco:ServiceContext>
2527   </disco:SvcMD>
2528 </disco:SvcMDRegister>
```

2529 Things to note about this request:

- 2530 • a single SvcMD is registered
- 2531 • no `svcMDID` attribute is specified on the SvcMD during registration (it will be assigned by the DS if the request is
2532 successful).
- 2533 • the service being registered is a test payment service provided by the same provider we've seen earlier, supports a
2534 single service and framework version and exposes three different security mechanisms.

2535 Server Response:

```
2536 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2537   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2538   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2539 </disco:SvcMDRegisterResponse>
```

2540 Things to note about this response:

- 2541 • the registration was successful (status code is OK).
- 2542 • The SvcMD was assigned the `svcMDID SVCMDIDg9WP0thd_HvPd427KY9M`.

2543 **3.13.8. SvcMDQuery w/Good SvcMDID**

2544 Query the SvcMD that we just registered using the SvcMDID that was returned in the response.

```
2545 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2546   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2547 </disco:SvcMDQuery>
```

2548 Things to note about this query:

- 2549 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous
2550 step.

2551 Server Response:

```
2552 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2553   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2554   <disco:SvcMD svcMDID="SVCMDIDg9WP0thd_HvPd427KY9M">
2555     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2556     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2557     <disco:ServiceContext>
2558       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2559       <disco:EndpointContext>
2560         <disco:Address>https://payment.testing.com</disco:Address>
2561         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2562         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2563         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2564         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2565       </disco:EndpointContext>
2566     </disco:ServiceContext>
2567   </disco:SvcMD>
2568 </disco:SvcMDQueryResponse>
```

2569 Things to note about this response:

- 2570 • the query was successful (status code is OK).
- 2571 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2572 • The remaining data is identical to that registered by the provider in the previous request.

2573 **3.13.9. SvcMDDelete w/Good SvcMDID**

2574 Delete the Service Metadata that we just registered and queried.

```
2575 <disco:SvcMDDelete xmlns:disco="urn:liberty:disco:2006-08">
2576   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2577 </disco:SvcMDDelete>
```

2578 Things to note about this query:

- 2579 • the SvcMDID that we obtained in the registration above is used

2580 Server Response:

```
2581 <disco:SvcMDDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2582   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2583 </disco:SvcMDDeleteResponse>
```

2584 Things to note about this response:

- 2585 • the delete was successful (status code is OK).

2586 **3.13.10. SvcMDelete w/Already Deleted SvcMDID**

2587 Delete the same service metadata that we just deleted.

```
2588 <disco:SvcMDelete xmlns:disco="urn:liberty:disco:2006-08">
2589   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2590 </disco:SvcMDelete>
```

2591 Things to note about this query:

- 2592 • the SvcMDID that we already deleted is specified

2593 Server Response:

```
2594 <disco:SvcMDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2595   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2596 </disco:SvcMDeleteResponse>
```

2597 Things to note about this response:

- 2598 • the delete was successful (status code is OK), even though the SvcMD was already deleted as the delete of a
2599 non-existent SvcMD is defined by this specification to be successful.

2600 **3.13.11. SvcMDRegister w/Complex SvcMD**

2601 A basic registration of a SvcMD with the Discovery Service. This is a little bit different in that the SvcMD is rather
2602 complex (which is good for testing the Query interface).

```
2603 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2604   <disco:SvcMD>
2605     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2606     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2607     <disco:ServiceContext>
2608       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
2609       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
2610       <disco:EndpointContext>
2611         <disco:Address>https://old-calendars.testing.com</disco:Address>
2612         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
2613         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:Bearer</disco:SecurityMechID>
2614         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:null</disco:SecurityMechID>
2615       </disco:EndpointContext>
2616       <disco:EndpointContext>
2617         <disco:Address>https://old2-calendars.testing.com</disco:Address>
2618         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2619         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2620         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2621       </disco:EndpointContext>
2622       <disco:EndpointContext>
2623         <disco:Address>http://old-calendars.testing.com</disco:Address>
2624         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2625         <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
2626         <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2627       </disco:EndpointContext>
2628     </disco:ServiceContext>
2629   </disco:ServiceContext>
2630   <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2631   <disco:EndpointContext>
2632     <disco:Address>https://calendars.testing.com</disco:Address>
2633     <disco:Address>https://calendars.testing.backup.com</disco:Address>
2634     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2635     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2636     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2637   </disco:EndpointContext>
2638   <disco:EndpointContext>
```

```
2639     <disco:Address>http://calendars.testing.com</disco:Address>
2640     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2641     <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2642   </disco:EndpointContext>
2643 </disco:ServiceContext>
2644 </disco:SvcMD>
2645 </disco:SvcMDRegister>
```

2646 Things to note about this query:

- 2647 • it's a rather complex SvcMD, but registration is still the same.

2648 Server Response:

```
2649 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2650   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2651   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2652 </disco:SvcMDRegisterResponse>
```

2653 Things to note about this response:

- 2654 • the registration was successful (status code is OK).
- 2655 • The SvcMD was assigned the svcMDID *SVCMDIDmifyjzIKO6tNd8evymnL*.
- 2656 • Now the fun begins as this SvcMD let's us exercise many of the interesting portions of the Discovery Query
2657 interface.

2658 **3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD**

2659 Query the SvcMD that we just registered using the SvcMDID that was returned in the response to make sure it was
2660 registered correctly.

```
2661 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2662   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2663 </disco:SvcMDQuery>
```

2664 Things to note about this query:

- 2665 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous
2666 step.

2667 Server Response:

```

2668 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2669   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2670   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2671     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2672     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2673     <disco:ServiceContext>
2674       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
2675       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
2676       <disco:EndpointContext>
2677         <disco:Address>https://old-calendars.testing.com</disco:Address>
2678         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
2679         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:Bearer</disco:SecurityMechID>
2680         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:null</disco:SecurityMechID>
2681       </disco:EndpointContext>
2682       <disco:EndpointContext>
2683         <disco:Address>https://old2-calendars.testing.com</disco:Address>
2684         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2685         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2686         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2687       </disco:EndpointContext>
2688       <disco:EndpointContext>
2689         <disco:Address>http://old-calendars.testing.com</disco:Address>
2690         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2691         <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
2692         <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2693       </disco:EndpointContext>
2694     </disco:ServiceContext>
2695     <disco:ServiceContext>
2696       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2697       <disco:EndpointContext>
2698         <disco:Address>https://calendars.testing.com</disco:Address>
2699         <disco:Address>https://calendars.testing.backup.com</disco:Address>
2700         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2701         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2702         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2703       </disco:EndpointContext>
2704       <disco:EndpointContext>
2705         <disco:Address>http://calendars.testing.com</disco:Address>
2706         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2707         <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2708       </disco:EndpointContext>
2709     </disco:ServiceContext>
2710   </disco:SvcMD>
2711 </disco:SvcMDQueryResponse>

```

2712 Things to note about this response:

- 2713 • the query was successful (status code is OK).
- 2714 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2715 • The remaining data is identical to that registered by the provider in the previous request. Key here is that the
- 2716 ordering and grouping of elements has been maintained.

2717 **3.13.13. SvcMDReplace of existing SvcMD**

2718 Replace the complex SvcMD that we just registered with a simpler version

```

2719 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
2720   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2721     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2722     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2723     <disco:ServiceContext>

```

```
2724 <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2725 <disco:EndpointContext>
2726 <disco:Address>https://payment.testing.com</disco:Address>
2727 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2728 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2729 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2730 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2731 </disco:EndpointContext>
2732 </disco:ServiceContext>
2733 </disco:SvcMD>
2734 </disco:SvcMDReplace>
```

2735 Things to note about this request:

- 2736 • the svcMDID attribute has the value obtained during the previous registration so that registration should be
2737 replaced.

2738 Server Response:

```
2739 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
2740 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2741 </disco:SvcMDReplaceResponse>
```

2742 Things to note about this response:

- 2743 • the replacement was successful (status code is OK).
- 2744 • no other data is returned. The SvcMDID does not change when the SvcMD is replaced.

2745 3.13.14. SvcMDQuery of Replaced SvcMDID

2746 Query the SvcMD that we just replaced.

```
2747 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2748 <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2749 </disco:SvcMDQuery>
```

2750 Things to note about this query:

- 2751 • We use the same svcMDID that we had for the previous SvcMD as the value doesn't change when we do a
2752 replacement.

2753 Server Response:

```
2754 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2755 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2756 <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2757 <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2758 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2759 <disco:ServiceContext>
2760 <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2761 <disco:EndpointContext>
2762 <disco:Address>https://payment.testing.com</disco:Address>
2763 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2764 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2765 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2766 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2767 </disco:EndpointContext>
2768 </disco:ServiceContext>
2769 </disco:SvcMD>
2770 </disco:SvcMDQueryResponse>
```

2771 Things to note about this response:

2772 • the query was successful (status code is OK).

2773 • the new SvcMD is returned.

2774 3.13.15. SvcMDDelete of replaced SvcMD

2775 Delete the Service Metadata that we just replaced (to clean up after ourselves).

```
2776 <disco: SvcMDDelete xmlns: disco="urn: liberty: disco: 2006-08">
2777   <disco: SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco: SvcMDID>
2778 </disco: SvcMDDelete>
```

2779 Things to note about this query:

2780 • the SvcMDID that we used in the replacement above is used

2781 Server Response:

```
2782 <disco: SvcMDDeleteResponse xmlns: disco="urn: liberty: disco: 2006-08">
2783   <lu: Status xmlns: lu="urn: liberty: util: 2006-08" code="OK"/>
2784 </disco: SvcMDDeleteResponse>
```

2785 Things to note about this response:

2786 • the delete was successful (status code is OK).

2787 3.13.16. SvcMDRegister w/multiple SvcMDs

2788 A registration of 3 different SvcMD elements of varying levels of complexity

```
2789 <disco: SvcMDRegister xmlns: disco="urn: liberty: disco: 2006-08">
2790   <disco: SvcMD>
2791     <disco: Abstract>TestDisco Test Calendar Service</disco: Abstract>
2792     <disco: ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco: ProviderID>
2793     <disco: ServiceContext>
2794       <disco: ServiceType>urn:x-test:cal:2006-01</disco: ServiceType>
2795       <disco: ServiceType>urn:x-test:cal:2006-09</disco: ServiceType>
2796       <disco: EndpointContext>
2797         <disco: Address>https://1-calendars.testing.com</disco: Address>
2798         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.1"/>
2799         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2800         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: null</disco: SecurityMechID>
2801       </disco: EndpointContext>
2802       <disco: EndpointContext>
2803         <disco: Address>https://2-calendars.testing.com</disco: Address>
2804         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2805         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2806         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: null</disco: SecurityMechID>
2807       </disco: EndpointContext>
2808       <disco: EndpointContext>
2809         <disco: Address>http://3-calendars.testing.com</disco: Address>
2810         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2811         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: Bearer</disco: SecurityMechID>
2812         <disco: SecurityMechID>urn: liberty: security: 2006-08: null: SAMLV2</disco: SecurityMechID>
2813       </disco: EndpointContext>
2814     </disco: ServiceContext>
2815   </disco: SvcMD>
2816   <disco: SvcMD>
2817     <disco: ServiceType>urn:x-test:cal:2008-03</disco: ServiceType>
2818     <disco: EndpointContext>
2819       <disco: Address>https://4-calendars.testing.com</disco: Address>
2820       <disco: Address>https://5-calendars.testing.backup.com</disco: Address>
2821       <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2822       <disco: SecurityMechID>urn: liberty: security: 2006-08: TLS: SAMLV2</disco: SecurityMechID>
2823       <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
```

```
2823     </disco:EndpointContext>
2824     <disco:EndpointContext>
2825         <disco:Address>http://6-calendars.testing.com</disco:Address>
2826         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2827         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2828     </disco:EndpointContext>
2829 </disco:ServiceContext>
2830 </disco:SvcMD>
2831 <disco:SvcMD>
2832     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2833     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2834     <disco:ServiceContext>
2835         <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2836         <disco:EndpointContext>
2837             <disco:Address>https://payment.testing.com</disco:Address>
2838             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2839             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2840             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2841             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2842         </disco:EndpointContext>
2843     </disco:ServiceContext>
2844 </disco:SvcMD>
2845 <disco:SvcMD>
2846     <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
2847     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2848     <disco:ServiceContext>
2849         <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
2850         <disco:Options>
2851             <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
2852             <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
2853             <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
2854         </disco:Options>
2855         <disco:EndpointContext>
2856             <disco:Address>https://test2.atm.CA.US.testing.com</disco:Address>
2857             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2858             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2859             <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2860         </disco:EndpointContext>
2861         <disco:EndpointContext>
2862             <disco:Address>https://readers.atm.CA.US.testing.com</disco:Address>
2863             <disco:Address>https://readers.atm.NY.US.testing.com</disco:Address>
2864             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2865             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2866             <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2867             <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
2868         </disco:EndpointContext>
2869         <disco:EndpointContext>
2870             <disco:Address>https://writers.atm.testing.com</disco:Address>
2871             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2872             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2873             <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
2874             <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
2875         </disco:EndpointContext>
2876     </disco:ServiceContext>
2877 </disco:SvcMD>
2878 </disco:SvcMDRegister>
```

2879 Things to note about this registration:

- 2880 • there SvcMDs use many of the features of the data structures to define various contexts in which the service can
2881 be reached and which actions and/or options are available at said services.
- 2882 • We do **NOT** represent that the SvcMDs registered here are typical or normal. They were explicitly created to
2883 examine/test some of the intricacies of handling queries against the SvcMD data structure.

2884 Server Response:

```
2885 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2886   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2887   <disco:SvcMDID>SVCMDIDMFPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2888   <disco:SvcMDID>SVCMDIDeZQGkXuw750_uh3Q9OLO</disco:SvcMDID>
2889   <disco:SvcMDID>SVCMDIDrpG8SJpeUdSmla_ZSFUN</disco:SvcMDID>
2890 </disco:SvcMDRegisterResponse >
```

2891 Things to note about this response:

- 2892 • the registration was successful (status code is OK).
- 2893 • the svcMDID for each of the added elements is returned in sequence. The first SvcMDID to the first <SvcMD> in
- 2894 the request, the second to the second, and so forth.

2895 **3.13.17. Query Calendar Service**

2896 Query for the test calendar service for this user.

```
2897 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2898   <disco:RequestedService>
2899     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2900     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2901   </disco:RequestedService>
2902 </disco:Query>
```

2903 Things to note about this query:

- 2904 • the service type on this query is one of those specified in one of the SvcMDs that were just registered in the
- 2905 previous call.

2906 Server Response:

```
2907 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2908   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2909     <lu:Status code="NoResults"/>
2910   </lu:Status>
2911 </disco:QueryResponse>
```

2912 Things to note about this response:

- 2913 • the query failed (status code was Failed). This is because while the SvcMD has been *registered*, it has **not** been
- 2914 *associated* with the user.

2915 **3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD**

2916 Associate a single SvcMD with the current principal.

```
2917 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">
2918   <disco:SvcMDID>SVCMDIDMFPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2919 </disco:SvcMDAssociationAdd>
```

2920 Things to note about this query:

- 2921 • the SvcMDID specified is the SvcMDID assigned to the test calendar service registered above (the first SvcMD in
- 2922 the multi-SvcMD registration).

2923 Server Response:

```
2924 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">
2925   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2926 </disco:SvcMDAssociationAddResponse>
```

2927 Things to note about this response:

- 2928 • the query was successful (status code is OK).
- 2929 • this service is now associated with the principal and available to subsequent Discovery Service queries.

2930 3.13.19. SvcMDAssociationQuery w/SvcMDID

2931 Query the SvcMD Associations to see if that SvcMD is now associated with the principal

```
2932 <disco:SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08">
2933   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2934 </disco:SvcMDAssociationQuery>
```

2935 Things to note about this query:

- 2936 • the SvcMDID provided is the SvcMDID that was just associated with the user in the previous request

2937 Server Response:

```
2938 <disco:SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2939   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2940   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2941 </disco:SvcMDAssociationQueryResponse>
```

2942 Things to note about this response:

- 2943 • the query was successful (status code is OK).
- 2944 • the matching SvcMDIDs were returned (since you can query more than one and not all may have matched on a
- 2945 successful query)

2946 3.13.20. SvcMDAssociationQuery w/o SvcMDID

2947 Query all SvcMD Associations for the current principal.

```
2948 <disco:SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08"/>
```

2949 Things to note about this query:

- 2950 • No SvcMDIDs are specified which causes all SvcMD associations that were created by the invoking provider to
- 2951 be listed.

2952 Server Response:

```
2953 <disco:SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2954   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2955   <disco:SvcMDID>SVCMDIDy_7yFABwuFPkgYIjKcjv</disco:SvcMDID>
2956   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2957 </disco:SvcMDAssociationQueryResponse>
```

2958 Things to note about this response:

- 2959 • the query was successful (status code is OK).

- 2960 • two SvcMDIDs were returned. One for the service recently associated and one for the ID-WSF People Service
2961 that had been previously associated. (This call, as well as the previous registration and association calls, was made
2962 in the context of that provider so both should be visible).

2963 3.13.21. Query Calendar Service (again)

2964 Query for the test calendar service for this user (which previously failed, but now the SvcMD has been associated with
2965 this principal).

```
2966 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2967   <disco:RequestedService>
2968     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2969     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2970   </disco:RequestedService>
2971 </disco:Query>
```

2972 Things to note about this query:

- 2973 • again we're looking for the test calendar service (which has now been associated with the principal).
- 2974 • This query was constructed to be resolvable only by the data within the 2nd <ServiceContext> element (it's the
2975 only one with the ServiceType "urn:x-test:cal:2008-03") and within there, the 2nd <EndpointContext> element
2976 (it's the only one with the SecurithMechID ...:null:SAMLV2).

2977 Server Response:

```
2978 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2979   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2980   <wsa:EndpointReference
2981     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2982     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2983     notOnOrAfter="2006-04-06T17:40:28Z"
2984     wsu:Id="EPRID3jLhZ6fsjx3xFNF22Hyx">
2985     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
2986     <wsa:Metadata>
2987       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2988       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2989       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2990       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2991       <disco:SecurityContext>
2992         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2993         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2994           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2995           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2996             ID="CRED-HgB2SRBPptovTK7ckof"
2997             IssueInstant="2006-04-06T15:40:28Z"
2998             Version="2.0">
2999             ... assertion data was here ...
3000           </sa:Assertion>
3001         </sec:Token>
3002       </disco:SecurityContext>
3003     </wsa:Metadata>
3004   </wsa:EndpointReference>
3005 </disco:QueryResponse>
```

3006 Things to note about this response:

- 3007 • the query was successful (status code is OK).
- 3008 • the ID-WSF EPR was generated from the expected SvcMD elements (the unique address *http://6-*
3009 *calendars.testing.com* shows this).

3010 3.13.22. Query Calendar Service w/Action

3011 Query for a Calendar Service including an <Action> element in the request.

```
3012 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3013   <disco:RequestedService>
3014     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3015     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3016     <disco:Action>urn:x-test:cal:2008-03:GetMeeting</disco:Action>
3017   </disco:RequestedService>
3018 </disco:Query>
```

3019 Things to note about this query:

3020 • the action specified on the request is **not** explicitly listed in the registered SvcMD.

3021 Server Response:

```
3022 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3023   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3024   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3025     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3026     notOnOrAfter="2006-04-06T17:41:01Z"
3027     wsu:Id="EPRIDq9XRiWZpP_tftRcutOP">
3028     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3029     <wsa:Metadata>
3030       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3031       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3032       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3033       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3034       <disco:SecurityContext>
3035         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3036         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3037           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3038           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3039             ID="CREDst8YWUPHbqUdMKjnV_I7"
3040             IssueInstant="2006-04-06T15:41:01Z"
3041             Version="2.0">
3042             ... assertion data was here ...
3043           </sa:Assertion>
3044         </sec:Token>
3045       </disco:SecurityContext>
3046     </wsa:Metadata>
3047   </wsa:EndpointReference>
3048 </disco:QueryResponse>
```

3049 Things to note about this response:

3050 • the query was successful (status code is OK).

3051 • the returned ID-WSF EPR is essentially identical to the ID-WSF EPR returned in the previous query (only differing
3052 in timestamps and element IDs).

3053 • the <Action> specified on the request was considered to be matched because **no** <Action> elements were
3054 specified in the registered SvcMD – which by definition means that the SvcMD matches all possible <Action>
3055 values.

3056 3.13.23. Query Calendar Service w/resultsType=all

3057 Query for the Calendar Service specifying the "...:TLS:SAMLV2" SecurityMechID and resultsType=all

```
3058 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3059   <disco:RequestedService resultsType="all">
3060     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3061     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3062   </disco:RequestedService>
3063 </disco:Query>
```

3064 Things to note about this query:

3065 • resultsType setting of "all" indicates that the requestor wants all possible results, not just a limited match. This is
 3066 typically done when the client wants to choose which of the results to use.

3067 • This query was constructed to be resolvable only by the data within the 1st <EndpointContext> element (the Sec-
 3068 urityMechID ...:TLS:SAMLV2) in the 2nd <ServiceContext> element (the ServiceType "urn:x-test:cal:2008-
 3069 03").

3070 Server Response:

```
3071 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3072   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3073   <wsa:EndpointReference
3074     xmlns:wsa="http://www.w3.org/2005/08/addressing"
3075     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3076     notOnOrAfter="2006-04-06T17:41:05Z" wsu:Id="EPRIDgNpGPkjrRsZVIC63VOMt">
3077     <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3078   <wsa:Metadata>
3079     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3080     <sbf:Framework xmlns:sbf="urn:liberty:sbf" version="2.0"/>
3081     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3082     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3083     <disco:SecurityContext>
3084       <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3085       <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3086         usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3087         <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3088           ID="CREDYA0WkksXWLutGxWWsF2m"
3089           IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3090           ... assertion data was here ...
3091         </sa:Assertion>
3092       </sec:Token>
3093     </disco:SecurityContext>
3094   </wsa:Metadata>
3095 </wsa:EndpointReference>
3096 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3097   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3098   notOnOrAfter="2006-04-06T17:41:06Z" wsu:Id="EPRIDeBlMvrPuq-TphWEqQ7G0">
3099   <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3100 <wsa:Metadata>
3101   <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3102   <sbf:Framework xmlns:sbf="urn:liberty:sbf" version="2.0"/>
3103   <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3104   <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3105   <disco:SecurityContext>
3106     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3107     <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3108       usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3109       <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3110         ID="CRED816B4EvhFszmG01Ab5F7"
3111         IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3112         ... assertion data was here ...
3113       </sa:Assertion>
```

```

3114     </sec:Token>
3115     </disco:SecurityContext>
3116     </wsa:Metadata>
3117     </wsa:EndpointReference>
3118 </disco:QueryResponse>

```

3119 Things to note about this response:

- 3120 • the query was successful (status code is OK).
- 3121 • as expected 2 ID-WSF EPRs were returned because the two endpoints which matched the search criteria cannot
- 3122 be placed into the same EPR.

3123 3.13.24. Query for all Calendar Service EPRs

3124 A query of all of the data available for the calendar service

```

3125 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3126   <disco:RequestedService resultsType="all">
3127     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3128     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3129     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3130     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3131     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3132     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3133     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3134     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3135   </disco:RequestedService>
3136 </disco:Query>

```

3137 Things to note about this query:

- 3138 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3139 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3140 Server Response:

```

3141 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3142   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
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="EPRIDC08vEmUOfarryqg3zo8j">
3146     <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3147     <wsa:Metadata>
3148       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3149       <sbfl:Framework xmlns:sbfl="urn:liberty:sbfl" version="2.1"/>
3150       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3151       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3152       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3153       <disco:SecurityContext>
3154         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3155         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3156           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3157           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3158             ID="CRED1QVGIZghandZqdE90QRa"
3159             IssueInstant="2006-04-06T15:41:09Z" Version="2.0">
3160             ... assertion data was here ...
3161           </sa:Assertion>
3162         </sec:Token>
3163       </disco:SecurityContext>
3164       <disco:SecurityContext>
3165         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3166       </disco:SecurityContext>

```

```

3167     </wsa:Metadata>
3168 </wsa:EndpointReference>
3169 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3170     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3171     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDqEcv9d0T43OTQLdhB116">
3172   <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3173   <wsa:Metadata>
3174     ... Metatdata was here ...
3175   </wsa:Metadata>
3176 </wsa:EndpointReference>
3177 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3178     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3179     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDRpIps8-wMwgf4A_7DsHS">
3180   <wsa:Address>http://3-calendars.testing.com</wsa:Address>
3181   <wsa:Metadata>
3182     ... Metatdata was here ...
3183   </wsa:Metadata>
3184 </wsa:EndpointReference>
3185 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3186     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3187     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDkL15ahJdrutDV6gMPLyr">
3188   <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3189   <wsa:Metadata>
3190     ... Metatdata was here ...
3191   </wsa:Metadata>
3192 </wsa:EndpointReference>
3193 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3194     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3195     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID0TmCLBCVoYZV7_R8jgky">
3196   <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3197   <wsa:Metadata>
3198     ... Metatdata was here ...
3199   </wsa:Metadata>
3200 </wsa:EndpointReference>
3201 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3202     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3203     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID9nsZ7UfgG5UMMymJLRp">
3204   <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3205   <wsa:Metadata>
3206     ... Metatdata was here ...
3207   </wsa:Metadata>
3208 </wsa:EndpointReference>
3209 </disco:QueryResponse>

```

3210 Things to note about this response:

- 3211 • the query was successful (status code is OK).
- 3212 • Much of the data in this response was elided in order to not waste alot of paper. The first EPR in the response is
3213 shown fairly completely.
- 3214 • There are 6 EPRs (the minimum way to represent all of the data in the SvcMD that matched the requested
3215 parameters).
- 3216 • The first 3 EPRs have two service types (for the 2006-01 and the 2006-09 versions of the calendar service) as a
3217 single ID-WSF EPR may have multiple service types if they are all related to the same logical service.

3218 3.13.25. Query for one Calendar Service EPRs

3219 A query for the first EPR out of all those within the Calendar service (to show the impact of the `resultsType` setting
3220 of *only-one*).

```

3221 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3222   <disco:RequestedService resultsType="only-one">

```

```
3223 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3224 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3225 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3226 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3227 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3228 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3229 <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3230 <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3231 </disco:RequestedService>
3232 </disco:Query>
```

3233 Things to note about this query:

- 3234 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3235 • The `resultsType` attribute is set to *"only-one"* indicating that the Discovery Service should only return the first
- 3236 matching ID-WSF EPR.

3237 Server Response:

```
3238 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3239 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3240 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3241   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3242   notOnOrAfter="2006-04-06T17:41:14Z" wsu:Id="EPRIDVh71zxFCQu4yWBOKLPzH">
3243 <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3244 <wsa:Metadata>
3245 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3246 <sbfb:Framework xmlns:sbfb="urn:liberty:sbfb" version="2.1"/>
3247 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3248 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3249 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3250 <disco:SecurityContext>
3251 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3252 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3253   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3254 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3255   ID="CREDYjkZsJiWfQwKf5gaJoze"
3256   IssueInstant="2006-04-06T15:41:14Z" Version="2.0">
3257   ... assertion data was here ...
3258 </sa:Assertion>
3259 </sec:Token>
3260 </disco:SecurityContext>
3261 <disco:SecurityContext>
3262 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3263 </disco:SecurityContext>
3264 </wsa:Metadata>
3265 </wsa:EndpointReference>
3266 </disco:QueryResponse>
```

3267 Things to note about this response:

- 3268 • the query was successful (status code is OK).
- 3269 • Only one ID-WSF EPR was returned (because of the query parameters) (as compared to the 6 returned in the
- 3270 previous example with the same query other than the `resultsType` attribute).
- 3271 • The one ID-WSF EPR that is returned is the **first** EPR that would have otherwise been returned.

3272 3.13.26. Query specific version of Calendar Service

3273 A query for the 2006-09 version of the Calendar Service.

```
3274 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3275 <disco:RequestedService>
3276   <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3277   <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3278 </disco:RequestedService>
3279 </disco:Query>
```

3280 Things to note about this query:

- 3281 • This query was constructed to be resolvable only by the data within the 1st <ServiceContext> element
3282 (it's the only one with the ServiceType "urn:x-test:cal:2006-09") and within there, both the 1st and the 2nd
3283 <EndpointContext> element (they are the only ones with the SecurithMechID ...:TLS:Bearer).

3284 Server Response:

```
3285 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3286 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3287 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3288   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3289   notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDVsvRlCWheodUvbDFelh">
3290 <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3291 <wsa:Metadata>
3292 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3293 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
3294 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3295 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3296 <disco:SecurityContext>
3297 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3298 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3299   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3300 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3301   ID="CREDFy-6w5S8iOVearNxJur_"
3302   IssueInstant="2006-04-06T15:41:19Z" Version="2.0">
3303   ... assertion data was here ...
3304 </sa:Assertion>
3305 </sec:Token>
3306 </disco:SecurityContext>
3307 </wsa:Metadata>
3308 </wsa:EndpointReference>
3309 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3310   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3311   notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDDKWLXiXVXpSJJi3oY3ge">
3312 <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3313 <wsa:Metadata>
3314 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3315 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3316 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3317 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3318 <disco:SecurityContext>
3319 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3320 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3321   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3322 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3323   ID="CREDW_y92CoMn7x57YOWzbnB"
3324   IssueInstant="2006-04-06T15:41:20Z" Version="2.0">
3325   ... assertion data was here ...
3326 </sa:Assertion>
3327 </sec:Token>
3328 </disco:SecurityContext>
3329 </wsa:Metadata>
3330 </wsa:EndpointReference>
3331 </disco:QueryResponse>
```

3332 Things to note about this response:

- 3333 • the query was successful (status code is OK).
- 3334 • Two ID-WSF EPRs were returned by the Discovery Service because the matching data had two different endpoints
3335 which must be represented in separate EPRs.
- 3336 • Because there was no `resultsType` specified on the request, the DS could have returned just the first ID-WSF
3337 EPR if they chose to as that ID-WSF EPR meets the basic requirements of the request. In this particular instance
3338 the Discovery Service acted as if "all" had been specified, but that should not be depended upon.
- 3339 If you need a particular `resultsType` behavior, you need to specify it on the request.
- 3340 • The Discovery Service could have used the same assertion for the two ID-WSF EPRs if appropriate. In such a
3341 case, the one of the ID-WSF EPRs would have had a `<sec:Token>` element with a `ref` attribute containing a
3342 pointer to the `<sec:Token>` in the other ID-WSF EPR.

3343 3.13.27. SvcMDReplace Calendar Service

3344 Replace the SvcMD for the Calendar service with a much simpler SvcMD.

```
3345 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
3346   <disco:SvcMD svcMDID="SVCMDIDMfPMblwcRSiwnu8D3BGO">
3347     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3348     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3349     <disco:ServiceContext>
3350       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3351       <disco:EndpointContext>
3352         <disco:Address>https://calendar.testing.com</disco:Address>
3353         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3354         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3355       </disco:EndpointContext>
3356     </disco:ServiceContext>
3357   </disco:SvcMD>
3358 </disco:SvcMDReplace>
```

3359 Things to note about this query:

- 3360 • the SvcMDID is the SvcMDID for the complex Calendar Service SvcMD that we have been querying over the past
3361 few requests.
- 3362 • the new SvcMD is much simpler.

3363 Server Response:

```
3364 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
3365   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3366 </disco:SvcMDReplaceResponse>
```

3367 Things to note about this response:

- 3368 • the replacement was successful (status code is OK).

3369 **3.13.28. Query old Calendar Service**

3370 Query the Calendar Service data that was replaced.

```
3371 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3372   <disco:RequestedService>
3373     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3374     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3375   </disco:RequestedService>
3376 </disco:Query>
```

3377 Things to note about this query:

3378 • this is the same query we ran earlier that obtained results.

3379 Server Response:

```
3380 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3381   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3382     <lu:Status code="NoResults"/>
3383   </lu:Status>
3384 </disco:QueryResponse>
```

3385 Things to note about this response:

3386 • the query failed (status code is Failed)

3387 • the sub-status is NoResults - indicating that no matching data was found

3388 • The new (replacement) SvcMD for the Calendar Service has taken effect for the principal without the need for it
3389 to be associated with the principal (as it is already associated).

3390 **3.13.29. Query for all Calendar Service EPRs**

3391 A query of all of the data available for the calendar service (same query we ran a few requests ago).

```
3392 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3393   <disco:RequestedService resultsType="all">
3394     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3395     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3396     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3397     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3398     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3399     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3400     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3401     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3402   </disco:RequestedService>
3403 </disco:Query>
```

3404 Things to note about this query:

3405 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.

3406 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3407 Server Response:

```
3408 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3409   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3410   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3411     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3412     notOnOrAfter="2006-04-06T17:41:32Z" wsu:Id="EPRID23j-JGWAXusqiV80ArzC">
3413     <wsa:Address>https://calendar.testing.com</wsa:Address>
3414     <wsa:Metadata>
3415       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3416       <sbef:Framework xmlns:sbef="urn:liberty:sb" version="2.0"/>
3417       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3418       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3419       <disco:SecurityContext>
3420         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3421         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3422           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3423           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3424             ID="CREDxHDqbb4F1TV7jSYxlmxq"
3425             IssueInstant="2006-04-06T15:41:32Z" Version="2.0">
3426             ... assertion data was here ...
3427           </sa:Assertion>
3428         </sec:Token>
3429       </disco:SecurityContext>
3430     </wsa:Metadata>
3431   </wsa:EndpointReference>
3432 </disco:QueryResponse>
```

3433 Things to note about this response:

- 3434 • the query was successful (status code is OK).
- 3435 • Only the data from the replacement SvcMD is represented in the one ID-WSF EPR in the results.

3436 **3.13.30. SvcMDAssociationAdd the ATM Service SvcMD**

3437 Associate the ATM Service SvcMD with the current principal.

```
3438 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">
3439   <disco:SvcMDID>SVCMDIDrpG8SJpeUdSmla_ZSFUN</disco:SvcMDID>
3440 </disco:SvcMDAssociationAdd>
```

3441 Things to note about this query:

- 3442 • the SvcMDID specified is the SvcMDID assigned to the test ATM service registered above (the third SvcMD in the multi-SvcMD registration that was done earlier).

3444 Server Response:

```
3445 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">
3446   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3447 </disco:SvcMDAssociationAddResponse>
```

3448 Things to note about this response:

- 3449 • the association was successful (status code is OK).
- 3450 • the ATM Service SvcMD is now associated with the principal and available to subsequent Discovery Service queries.

3452 3.13.31. Query ATM Service w/resultsType=best

3453 Query for the ATM Service with the resultsType setting of "best".

```
3454 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3455   <disco:RequestedService resultsType="best">
3456     <disco:ServiceType urn:x-test:atm:2003-03/>
3457     <disco:SecurityMechID urn:liberty:security:2006-08:TLS:SAMLV2/>
3458   </disco:RequestedService>
3459 </disco:Query>
```

3460 Things to note about this query:

- 3461 • No <disco:Action>s are specified which implies the caller wants to have access to all operations at the provider.
- 3462 This is important for the ATM Service because in the SvcMD, each endpoint was registered with a subset of actions
- 3463 (no one endpoint has them all, so the results will take several EPRs).

3464 Server Response:

```
3465 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3466   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3467   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3468     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3469     notOnOrAfter="2006-04-06T17:41:43Z" wsu:Id="EPRIDfmT9HOSbmMs3jZ1_qSuY">
3470     <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
3471     <wsa:Metadata>
3472       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3473       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3474       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3475       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3476       <disco:SecurityContext>
3477         <disco:SecurityMechID urn:liberty:security:2006-08:TLS:SAMLV2/>
3478         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3479           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3480           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3481             ID="CREDS8x8pCKTOzaQMI14fkel"
3482             IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3483             ... assertion data was here ...
3484           </sa:Assertion>
3485         </sec:Token>
3486       </disco:SecurityContext>
3487       <disco:Options>
3488         <disco:Option urn:x-test:atm:options:testopt1/>
3489         <disco:Option urn:x-test:atm:options:testopt2/>
3490         <disco:Option urn:x-test:atm:options:testopt3/>
3491       </disco:Options>
3492       <disco:Action urn:x-test:atm:2007-11:GetBalance/>
3493     </wsa:Metadata>
3494   </wsa:EndpointReference>
3495   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3496     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3497     notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDdHYSEKB8_w5bPicTPe8o">
3498     <wsa:Address>https://readers.atm.CA.US.testing.com</wsa:Address>
3499     <wsa:Metadata>
3500       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3501       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3502       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3503       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3504       <disco:SecurityContext>
3505         <disco:SecurityMechID urn:liberty:security:2006-08:TLS:SAMLV2/>
3506         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3507           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3508           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3509             ID="CRED4nQ1FHP3vzuoqRhTMZrf"
3510             IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
```

```

3511     ... assertion data was here ...
3512     </sa:Assertion>
3513     </sec:Token>
3514     </disco:SecurityContext>
3515     <disco:Options>
3516         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3517         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3518         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3519     </disco:Options>
3520     <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3521     <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
3522     </wsa:Metadata>
3523 </wsa:EndpointReference>
3524 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3525     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3526     notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDggifVjR-zSAxkokPyfCo">
3527     <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3528     <wsa:Metadata>
3529         <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3530         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3531         <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3532         <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3533         <disco:SecurityContext>
3534             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3535             <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3536                 usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3537                 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3538                     ID="CREDZNTfblH6VxPNrpwawtF6"
3539                     IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3540                     ... assertion data was here ...
3541                 </sa:Assertion>
3542             </sec:Token>
3543         </disco:SecurityContext>
3544         <disco:Options>
3545             <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3546             <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3547             <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3548         </disco:Options>
3549         <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3550         <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3551     </wsa:Metadata>
3552 </wsa:EndpointReference>
3553 </disco:QueryResponse>

```

3554 Things to note about this response:

- 3555 • the query was successful (status code is OK).
- 3556 • It took three ID-WSF EPRs to represent the set of actions at the ATM Service.
- 3557 • One might think that because the "...GetBalance" action is on both the 1st and 2nd ID-WSF EPR and it is the only
3558 action on the 1st ID-WSF EPR, the results could have excluded that ID-WSF EPR and the client would still get to
3559 all of the resources at the ATM Service.
- 3560 However, the WSP placed the 1st "...GetBalance" action in the first <EndpointContext> and therefore gives it a
3561 higher priority in the results.
- 3562 • The Discovery service could have left off the "...GetBalance" action on the 2nd ID-WSF EPR but that wouldn't
3563 have been much of a savings and so it was included. Clients should NOT depend upon this type of behavior.
- 3564 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the
3565 ID-WSF EPRs generated from that SvcMD.

3566 3.13.32. Query ATM Service w/Withdraw Action

3567 Query for the ATM Service where "...Withdraw" action is available.

```
3568 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3569   <disco:RequestedService resultsType="all">
3570     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3571     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3572     <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3573   </disco:RequestedService>
3574 </disco:Query>
```

3575 Things to note about this query:

- 3576 • The <disco:Action> element is specified with the "urn:x-test:atm:2007-11:Withdraw") action value. So the
3577 client only intends to use this operation at the ATM Service.

3578 Server Response:

```
3579 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3580   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3581   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3582     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3583     notOnOrAfter="2006-04-06T17:41:52Z" wsu:Id="EPRIDkueZCk5N4IpSmX-0BD-9">
3584     <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3585     <wsa:Metadata>
3586       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3587       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3588       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3589       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3590       <disco:SecurityContext>
3591         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3592         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3593           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3594           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3595             ID="CREDYfLT1S7tcZ8LkuU-zlrs"
3596             IssueInstant="2006-04-06T15:41:52Z" Version="2.0">
3597             ... assertion data was here ...
3598           </sa:Assertion>
3599         </sec:Token>
3600       </disco:SecurityContext>
3601       <disco:Options>
3602         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3603         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3604         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3605       </disco:Options>
3606       <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3607       <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3608     </wsa:Metadata>
3609   </wsa:EndpointReference>
3610 </disco:QueryResponse>
```

3611 Things to note about this response:

- 3612 • the query was successful (status code is OK).
- 3613 • Only one ID-WSF EPR was returned which contained the endpoint where the "...Withdraw" action is available.
- 3614 • The Discovery service could have left off the "...Transfer" action but that wouldn't have been much of a savings
3615 and so it was included. Clients should NOT depend upon this type of behavior.
- 3616 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the
3617 ID-WSF EPRs generated from that SvcMD.

3618 3.13.33. Query ATM Service w/Option

3619 Query for the ATM service specifying an option

```
3620 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3621 <disco:RequestedService resultsType="only-one">
3622   <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3623   <disco:Options>
3624     <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3625   </disco:Options>
3626   <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3627 </disco:RequestedService>
3628 </disco:Query>
```

3629 Things to note about this query:

- 3630 • The `resultsType` attribute is set to *"only-one"* indicating that the Discovery Service should only return the first
3631 matching ID-WSF EPR.
- 3632 • the `<Option>` element is included with one of the values present in the SvcMD for the ATM Service.

3633 Server Response:

```
3634 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3635 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3636 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3637   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3638   notOnOrAfter="2006-04-06T17:42:19Z" wsu:Id="EPRIDzKyaJ_h5Qb2jLjLjq59E">
3639 <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
3640 <wsa:Metadata>
3641 <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3642 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3643 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3644 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3645 <disco:SecurityContext>
3646 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3647 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3648   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3649 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3650   ID="CRED0FAqvGgoeySpLTZHbJa7"
3651   IssueInstant="2006-04-06T15:42:19Z" Version="2.0">
3652   ... assertion data was here ...
3653 </sa:Assertion>
3654 </sec:Token>
3655 </disco:SecurityContext>
3656 <disco:Options>
3657 <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3658 <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3659 <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3660 </disco:Options>
3661 <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3662 </wsa:Metadata>
3663 </wsa:EndpointReference>
3664 </disco:QueryResponse>
```

3665 Things to note about this response:

- 3666 • the query was successful (status code is OK).
- 3667 • The one option specified in the request matched one of the options specified in the SvcMD, so that is considered a
3668 match. The caller does not have to specify all of the options in the SvcMD (but the SvcMD does have to have all
3669 of the options listed on the request).

- 3670 • Even though only one option was specified on the request, all of the options listed in the SvcMD are included in
3671 the response.

3672 3.13.34. Query ATM Service w/unknown Option

3673 Query for the ATM service with an option that doesn't exist in the SvcMD.

```
3674 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">  
3675 <disco:RequestedService resultsType="all">  
3676 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>  
3677 <disco:Options>  
3678 <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>  
3679 </disco:Options>  
3680 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>  
3681 </disco:RequestedService>  
3682 </disco:Query>
```

3683 Things to note about this query:

- 3684 • The option specified is **not** in the ATM Service SvcMD.

3685 Server Response:

```
3686 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">  
3687 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">  
3688 <lu:Status code="NoResults" />  
3689 </lu:Status>  
3690 </disco:QueryResponse>
```

3691 Things to note about this response:

- 3692 • the query failed (status code is Failed).
3693 • The sub-status was "NoResults" indicating no data matched the requested parameters

3694 3.13.35. Query ATM Service w/good and bad Option

3695 Query for the ATM service with an option that does exist and an option that doesn't exist in the SvcMD.

```
3696 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">  
3697 <disco:RequestedService resultsType="all">  
3698 <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>  
3699 <disco:Options>  
3700 <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>  
3701 <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>  
3702 </disco:Options>  
3703 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>  
3704 </disco:RequestedService>  
3705 </disco:Query>
```

3706 Things to note about this query:

- 3707 • both specified options have to be available for this request to be considered matched (if, on the other hand, the
3708 request had the two <disco:Option> elements in separate <disco:Options> containers, a SvcMD could
3709 match either one).

3710 Server Response:

```
3711 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">  
3712   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">  
3713     <lu:Status code="NoResults" />  
3714   </lu:Status>  
3715 </disco:QueryResponse>
```

3716 Things to note about this response:

- 3717 • the query failed (status code is Failed).
- 3718 • The sub-status was "NoResults" indicating no data matched the requested parameters

3719 4. Discovery Service ID-WSF EPR conveyed via a Security Token

3720 In both single sign-on and web services environments, many recipients of a security token find the need to subsequently
3721 invoke the identified principal's Discovery Service in order to discover and invoke identity services on behalf of said
3722 principal. For example, a SAML SP upon receiving an SSO assertion may want to discover and invoke the principals
3723 Profile Service and would need the Discovery Service ID-WSF EPR in order to do so.

3724 In the SSO environment, this concept is often referred to as the "Discovery Service Bootstrap" in that the SP is using
3725 the data in the SSO assertion to bootstrap into the ID-WSF environment.

3726 The need for this Discovery Service ID-WSF EPR is not restricted to SSO environments as any WSP that is invoked
3727 by a WSC may in turn need to act as a WSC and invoke other WSPs in order to fulfill the requested operation. For
3728 example, a Profile Service WSP may need to invoke the Interaction Service in order to request consent from the user
3729 before releasing data to a WSC.

3730 This section describes the recommended interoperable method for an Identity Provider and/or Discovery Service
3731 can embed an ID-WSF EPR for the Discovery Service within security and/or Identity tokens that they issue.
3732 Unfortunately, because of the variance in structure and formats of various tokens, the model used tends to be specific
3733 to the format of the security token. The remainder of this section documents how this is accomplished within some
3734 specific token formats.

3735 4.1. EPR Generation Rules

3736 The Discovery Service Bootstrap ID-WSF EPR which is placed into any security token must be generated according
3737 to the following rules:

3738 • The `<wsa:EndpointReference>` that MAY contain `<SecurityContext>` element(s) in turn containing
3739 `<sec:Token>` elements containing embedded security tokens, which are necessary to access the Discovery Ser-
3740 vice instance(s).

3741 • The `<sec:Token>` element MAY instead include a reference to an external security token using a
3742 `<wsse:SecurityTokenReference>` containing a non-relative URI reference to a security token.

3743 • The `<sec:Token>` element's `ref` attribute MAY instead refer to local security token available elsewhere in the
3744 same security token (such as another ID-WSF EPR within the security token). These references SHOULD only
3745 refer to elements within the security token carrying the ID-WSF EPR so that the reference will remain valid if the
3746 security token is separated from any message carrying the token.

3747 It is even possible (and in some cases typical) for the reference to be to the enveloping security token itself (the
3748 security token that contains this ID-WSF EPR) In such cases, the enveloping security token SHOULD carry the
3749 necessary information to support its consumption at the Discovery Service (as well as the information necessary
3750 for consumption at its primary relying party (the SP/WSP)).

3751 For example, with a SAML Assertion, this includes:

3752 • A second `<Audience>` element with the Discovery Service's ProviderID.

3753 • A subject confirmation method that the relying party can meet. This will frequently be
3754 `urn:oasis:names:tc:SAML:2.0:cm:bearer` in which case the same confirmation can be used by
3755 both parties. However, the assertion could contain multiple confirmation methods one for the initial party to
3756 use when invoking the relying party and one for the relying party to use when invoking the DS.

3757 This will allow the Discovery Service to validate the assertion using the normal assertion processing rules without
3758 having to manage some form of exception for self issued assertions.

3759 4.2. SAML 2.0 Security Tokens

3760 In a SAML 2.0 Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the
3761 <saml2:AttributeStatement> element in a <saml2:Assertion>.

3762 The <saml2:AttributeStatement> SHOULD be constructed according to the following rules:

3763 • The Name attribute of the <saml2:Attribute> element MUST be:

3764 *urn:liberty:disco:2006-08:DiscoveryEPR*

3765 • The NameFormat attribute of the <saml2:Attribute> element MUST be:

3766 *urn:oasis:names:tc:SAML:2.0:attrname-format:uri*

3767 • One or more <saml2:AttributeValue> elements MUST be included which each containing a single
3768 <wsa:EndpointReference> element identifying a Discovery Service instance(s). These Discovery Ser-
3769 vice instances SHOULD offer identity services for the Principal identified in the Subject element inside the
3770 <saml2:Assertion>.

3771 An example <saml2:AttributeStatement> that might be found in a SAMLv2 <saml2:Assertion> follows.
3772 The example includes a <sec:Token> element which has a reference to the surrounding assertion.

```
3773
3774 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3775   <Attribute Name="urn:liberty:disco:2006-08:DiscoveryEPR"
3776     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
3777     <AttributeValue>
3778       <wsa:EndpointReference>
3779         <wsa:Address>https://example.com/disco/</wsa:Address>
3780
3781         <wsa:Metadata>
3782           <Abstract>
3783             The Principal's Discovery Service Resource
3784           </Abstract>
3785
3786           <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3787
3788           <ProviderID>http://example.com/</ProviderID>
3789
3790           <SecurityContext>
3791             <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer </SecurityMechID>
3792             <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3793             </SecurityContext>
3794           </wsa:Metadata>
3795         </wsa:EndpointReference>
3796       </AttributeValue>
3797     </Attribute>
3798   </AttributeStatement>
3799
```

3800 **Example 21.** <AttributeStatement> that might be found in a SAMLv2 AuthnResponse

3801 In all cases, this <AttributeStatement> MUST carry an ID-WSF EPR for the Liberty Discovery Service. Any
3802 other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

3803 4.3. SAML 1.x (Liberty ID-FF) Security Tokens

3804 In a SAML 1.x Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the
3805 <saml:AttributeStatement> element in a <saml:Assertion>.

3806 The <saml:AttributeStatement> SHOULD be constructed according to the following rules:

- 3807 • For the <saml:Attribute> element:
- 3808 • The AttributeName attribute MUST be "DiscoveryEPR".
- 3809 • The AttributeNamespace attribute MUST be "urn:liberty:disco:2006-08".
- 3810 • The <Subject> element of the <saml:AttributeStatement> element MUST carry the identity of the
3811 principal whose Discovery Service is referenced by this EPR and SHOULD be the same identity in the subject of
3812 the other statements in the <saml:Assertion>.
- 3813 • One or more <saml:AttributeValue> elements MUST be included which each containing a single
3814 <wsa:EndpointReference> element identifying a Discovery Service instance(s). These Discovery Ser-
3815 vice instances SHOULD offer identity services for the Principal identified in the Subject element inside this
3816 <saml:AttributeStatment>.
- 3817 An example <saml:AttributeStatement> that might be found in a SAML 1.1 <saml:Assertion> follows. The
3818 example includes a <sec:Token> element which has a reference to the surrounding assertion.

```

3819
3820 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
3821   <Subject>
3822     <NameIdentifier Format="urn:liberty:iff:nameid:federated">
3823       d0CQF8elJTDLmzEO
3824     </NameIdentifier>
3825   </Subject>
3826   <Attribute AttributeName="DiscoveryEPR"
3827     AttributeNamespace="urn:liberty:disco:2006-08">
3828     <AttributeValue>
3829       <wsa:EndpointReference>
3830         <wsa:Address>https://example.com/disco/</wsa:Address>
3831
3832         <wsa:Metadata>
3833           <Abstract>
3834             The Principal's Discovery Service Resource
3835           </Abstract>
3836
3837           <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3838
3839           <ProviderID>http://example.com/</ProviderID>
3840
3841           <SecurityContext>
3842             <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer</SecurityMechID>
3843             <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3844               </SecurityContext>
3845           </wsa:Metadata>
3846         </wsa:EndpointReference>
3847       </AttributeValue>
3848     </Attribute>
3849   </AttributeStatement>
3850

```

3851 **Example 22. <AttributeStatement> that might be found in a SAML 1.1 AuthnResponse**

- 3852 In all cases, this <AttributeStatement> MUST only carry an ID-WSF EPR for the Liberty Discovery Service.
3853 Any other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

3854 5. ID-WSF 1.x Resource Offering conveyed in an EPR

3855 In order to support the discovery and subsequent invocation of ID-WSF 1.0 and 1.1 services it may be necessary for
3856 the Discovery Service to carry the ID-WSF 1.x Resource Offering information within the ID-WSF EPR.

3857 The process involves taking the fields that would normally be present in the Resource Offering and placing them into
3858 the appropriate fields within the EPR according to the following rules:

- 3859 • The <ResourceID> element and/or the <EncryptedResourceID> element are placed into the <Metadata>
3860 element as-is.
- 3861 • The <ServiceType> element in the <ServiceInstance> element is placed into the <Metadata> element.
- 3862 • The <ProviderID> element in the <ServiceInstance> element is placed into the <Metadata> element.
- 3863 • The <SecurityMechID> element in ServiceInstance/Description is placed into the
3864 <SecurityContext> element (and will be combined with other SecurityMechIDs based upon whether or
3865 not they share the same endpoint *and* credential (or do not use a credential)).
- 3866 • The data from the <Endpoint> element in ServiceInstance/Description is placed into the <Address>
3867 element. Note that if there are multiple distinct <Endpoint>s they must be placed into different ID-WSF EPRs
3868 rather than being able to be placed into a single EPR like they were in an RO.
- 3869 • The <SoapAction> element in ServiceInstance/Description is placed into the <Metadata> element.
- 3870 • Options are placed into the <Metadata> element.
- 3871 • Abstract is placed into the <Metadata> element.
- 3872 • Credentials, which in the days of the Resource Offering were carried elsewhere in the message and referenced
3873 from the ServiceInstance/Description element are now carried directly within the EPR in a <sec:Token>
3874 element in the <SecurityContext> element.

3875 In addition, the ID-WSF EPR MUST also include at least one <sbfl:Framework> element with the appropriate value
3876 (1.0 or 1.1) in the version attribute for the ID-WSF version being used.

3877 As an example, let's start with an example ID-WSF 1.x Resource Offering:

```
3878  
3879 <ResourceOffering>  
3880   <ResourceID> 123 </ResourceID>  
3881   <ServiceInstance>  
3882     <ServiceType>urn:liberty:idsis-pp:2003-08</ServiceType>  
3883     <ProviderID>http://pp.services.aol.com</ProviderID>  
3884     <Description CredentialRef="1">  
3885       <SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</SecurityMechID>  
3886       <Endpoint>https://ep1.pp.service.aol.com</Endpoint>  
3887     </Description>  
3888     <Description>  
3889       <SecurityMechID>urn:liberty:security:2006-08:Client-TLS:Null</SecurityMechID>  
3890       <Endpoint>https://ep1.pp.service.aol.com</Endpoint>  
3891     </Description>  
3892   </ServiceInstance>  
3893 </ResourceOffering>  
3894 <Credentials>  
3895   <saml1:Assertion AssertionID="1" ...>  
3896     Assertion data goes here  
3897   </saml1:Assertion>  
3898 </Credentials>  
3899
```

3900 Translating this using the above rules would result in the following ID-WSF EPR:

```
3901
3902 <wsa:EndpointReference>
3903   <wsa:Address>https://ep1.pp.services.aol.com</wsa:Address >
3904   <wsa:Metadata>
3905     <ds1:ResourceID>123</ds1:ResourceID>
3906     <ds2:ProviderID>http://pp.services.aol.com</ds2:ProviderID>
3907     <ds2:ServiceType>urn:liberty:idsis-pp:2003-08</ds2:ServiceType>
3908     <ds2:FrameworkVersion="1.1" />
3909     <ds2:SecurityContext>
3910       <ds2:SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</ds2:SecurityMechID>
3911       <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3912         <saml1:Assertion AssertionID="1" ... >
3913           ... assertion data goes here ...
3914         </saml1:Assertion>
3915       </sec:Token>
3916     </ds2:SecurityContext>
3917   </ds2:SecurityContext>
3918   <ds2:SecurityMechID>
3919     urn:liberty:security:2006-08:Client-TLS:Null
3920   </ds2:SecurityMechID>
3921 </ds2:SecurityContext>
3922 </wsa:Metadata>
3923 </wsa:EndpointReference>
3924
```

3925 And subsequently, the invocation of the ID-WSF 1.x service would look to be something along the lines of (assuming
3926 that the WSC chose to use the "...:TLS:bearer" Security Mechanism):

```
3927
3928 <?xml version="1.0" encoding="utf-8" ?>
3929 <S:Envelope...
3930   <S:Header>
3931     <sb:Correlation S:mustUnderstand="1"
3932       messageID=uuid:958312848-29348938-232342121
3933       timestamp="2003-06-06T18:29:18Z" />
3934     <wsse:Security>
3935       <saml1:Assertion AssertionID="1" ... >
3936         ... assertion data goes here ...
3937       </saml1:Assertion>
3938     </wsse:Security>
3939   </S:Header>
3940   <S:Body>
3941     <pp:Query>
3942       <pp:ResourceID>123</pp:ResourceID>
3943       <pp:QueryItem>
3944         ... query data goes here ...
3945       </pp:QueryItem>
3946     </pp:Query>
3947   </S:Body>
3948 </S:Envelope>
3949
```

3950 **6. Acknowledgments**

3951 Many people have made contributions to this specification as it has evolved over time. The original specification was
3952 written by John Beatty with subsequent versions "inked" by Jonathan Sergent and later, Jeff Hodges and now myself.

3953 The changes made in this latest release of the specification are due in a large part to the work of Jeff Hodges, Robert
3954 Aarts, John Kemp, Gary Ellison and Greg Whitehead. Many others, including those that are listed as contributors
3955 on the cover page, have also played a part in this and earlier releases of the specification. Many thanks to all who
3956 participated (and apologies if I have forgotten to mention your name).

3957 References

3958 Normative

- 3959 [LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version v2.0, Liberty Alliance Project (30 July,
3960 2006). <http://www.projectliberty.org/specs>
- 3961 [LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version v2.0, Liberty
3962 Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 3963 [LibertySecMech11] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance
3964 Project (18 April 2004). <http://www.projectliberty.org/specs/>
- 3965 [LibertySecMech20SAML] Hirsch, Frederick, eds. "ID-WSF 2.0 SecMech SAML Profile," Version v2.0, Liberty
3966 Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 3967 [LibertyAuthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. "Liberty ID-WSF Authentication,
3968 Single Sign-On, and Identity Mapping Services Specification," Version v2.0, Liberty Alliance Project (30
3969 July, 2006). <http://www.projectliberty.org/specs>
- 3970 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Lib-
3971 erty ID-WSF SOAP Binding Specification," Version 2.0, Liberty Alliance Project (30 July, 2006).
3972 <http://www.projectliberty.org/specs>
- 3973 [LibertyIDWSF20SCR] Whitehead, Greg, eds. Version 1.0, Liberty Alliance Project (30 July, 2006).
3974 <http://www.projectliberty.org/specs>
- 3975 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
3976 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 3977 [RFC3986] Berners-Lee, T., Fielding, R., Masinter, L., eds. (January 2005). "Uniform Resource Identifier
3978 (URI): Generic Syntax," RFC 3986 (Obsoletes RFC2732, RFC2396, RFC1808) (Updates RFC1738) (Also
3979 STD0066) (Status: STANDARD), The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc3986.txt>
- 3980 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions
3981 and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OA-
3982 SIS Standard, Organization for the Advancement of Structured Information Standards [http://docs.oasis-
open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-
3983 open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 3984 [SAMLMeta2] Cantor, Scott, Moreh, Jahan, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Metadata for the OA-
3985 SIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for
3986 the Advancement of Structured Information Standards [http://docs.oasis-open.org/security/saml/v2.0/saml-
metadata-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-
3987 metadata-2.0-os.pdf)
- 3988 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October
3989 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium
3990 <http://www.w3.org/TR/xmlschema-1/>
- 3991 [WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds.
3992 World Wide Web Consortium W3C Recommendation (9 May 2006). [http://www.w3.org/TR/2006/REC-ws-
addr-core-20060509/](http://www.w3.org/TR/2006/REC-ws-
3993 addr-core-20060509/)
- 3994 [WSAv1.0-SOAP] "WS-Addressing 1.0 SOAP Binding," Gudgin, Martin, Hadley, Marc, eds. World Wide Web Con-
3995 sultium W3C Recommendation (9 May 2006). <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

- 3996 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith,
3997 ith, Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001).
3998 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 3999 [wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web
4000 Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for
4001 the Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/2004/01/oasis-200401-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
4002 [wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 4003 [wss-sms-draft] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 30,
4004 2003). "Web Services Security: SOAP Message Security," Draft WSS-SOAPMessageSecurity-14-
4005 06-2003, Organization for the Advancement of Structured Information Standards [http://www.oasis-](http://www.oasis-open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf)
4006 [open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf](http://www.oasis-open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf)
- 4007 [xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (10 December 2002). "XML Encryption Syntax and Process-
4008 ing," W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlenc-core/>

4009 Informative

- 4010 [LibertyPAOS] Aarts, Robert, Kemp, John, eds. "Liberty Reverse HTTP Binding for SOAP Specification," Version
4011 2.0, Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 4012 [LibertyClientProfiles] Aarts, Robert, Kainulainen, Jukka, Kemp, John, eds. Version v2.0, Liberty Alliance Project
4013 (30 July, 2006). <http://www.projectliberty.org/specs>

4014 A. Discovery Service Version 2.0 XSD

```
4015
4016 <?xml version="1.0" encoding="UTF-8"?>
4017 <xs:schema targetNamespace="urn:liberty:disco:2006-08"
4018   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
4019   xmlns:sb="urn:liberty:sb:2006-08"
4020   xmlns:sbf="urn:liberty:sb"
4021   xmlns:sec="urn:liberty:security:2006-08"
4022   xmlns:lu="urn:liberty:util:2006-08"
4023   xmlns:wsa="http://www.w3.org/2005/08/addressing"
4024   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
4025   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
4026   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
4027   xmlns:xs="http://www.w3.org/2001/XMLSchema"
4028
4029   xmlns="urn:liberty:disco:2006-08"
4030   elementFormDefault="qualified"
4031   attributeFormDefault="unqualified"
4032 >
4033
4034 <xs:import namespace="urn:liberty:util:2006-08"
4035   schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
4036
4037 <xs:import namespace="urn:liberty:sb:2006-08"
4038   schemaLocation="liberty-idwsf-soap-binding-v2.0.xsd"/>
4039
4040 <xs:import namespace="urn:liberty:sb"
4041   schemaLocation="liberty-idwsf-soap-binding.xsd"/>
4042
4043 <xs:import namespace="http://www.w3.org/2005/08/addressing"
4044   schemaLocation="ws-addr-1.0.xsd"/>
4045
4046 <xs:import namespace="urn:oasis:names:tc:SAML:2.0:metadata"
4047   schemaLocation="saml-schema-metadata-2.0.xsd"/>
4048
4049 <xs:import namespace="urn:liberty:security:2006-08"
4050   schemaLocation="liberty-idwsf-security-mechanisms-v2.0.xsd"/>
4051
4052 <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-s
4053 ecect-1.0.xsd"
4054   schemaLocation="wss-secext-1.0.xsd"/>
4055
4056 <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecuri
4057 ty-utility-1.0.xsd"
4058   schemaLocation="wss-util-1.0.xsd"/>
4059
4060 <xs:annotation>
4061   <xs:documentation>
4062     XML Schema from Liberty Discovery Service Specification.
4063   </xs:documentation>
4064   <xs:documentation>### NOTICE ###
4065
4066     Copyright (c) 2006 Liberty Alliance participants, see
4067     http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
4068
4069   </xs:documentation>
4070 </xs:annotation>
4071
4072 <!-- **** Discovery Service Data Elements & Types **** -->
4073
4074 <!-- The data elements and types in this section are used to
4075 embellish WS-Addressing Endpoint References (EPRs).
4076 They are placed in the /wsa:EndpointReference/Metadata
4077 element. Specific usage and cardinalities are stipulated
4078 in the Discovery Service v2.0 Specification. -->
4079
```

```

4080
4081 <!-- Abstract: natural-language description of service -->
4082
4083 <xs:element name="Abstract" type="xs:string"/>
4084
4085 <!-- Provider ID -->
4086
4087 <xs:element name="ProviderID" type="xs:anyURI"/>
4088
4089 <!-- Service Type -->
4090
4091 <xs:element name="ServiceType" type="xs:anyURI"/>
4092
4093 <!-- Framework Description -->
4094
4095 <xs:element name="Framework" type="sbf:FrameworkType" />
4096
4097 <!-- EPR Expiration Timestamp -->
4098
4099 <xs:attribute name="NotOnOrAfter" type="xs:dateTime"/>
4100
4101 <!-- Security Context Container -->
4102
4103 <xs:element name="SecurityContext">
4104   <xs:complexType>
4105     <xs:sequence>
4106       <xs:element ref="SecurityMechID"
4107         minOccurs="1"
4108         maxOccurs="unbounded" />
4109
4110       <xs:element ref="sec:Token"
4111         minOccurs="0"
4112         maxOccurs="unbounded" />
4113     </xs:sequence>
4114   </xs:complexType>
4115 </xs:element>
4116
4117 <!-- Security Mechanism ID -->
4118
4119 <xs:element name="SecurityMechID" type="xs:anyURI"/>
4120
4121 <!-- Options -->
4122
4123 <xs:element name="Options" type="OptionsType"/>
4124
4125 <xs:element name="Option" type="xs:anyURI" />
4126
4127 <xs:complexType name="OptionsType">
4128   <xs:sequence>
4129     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded" />
4130   </xs:sequence>
4131 </xs:complexType>
4132
4133 <!-- Address -->
4134
4135 <xs:element name="Address" type="xs:anyURI"/>
4136
4137 <!-- Action(s) - the interfaces available at this service -->
4138
4139 <xs:element name="Action" type="xs:anyURI" />
4140 <!-- Keys Element - For use in ModifyResponse -->
4141
4142 <xs:element name="Keys" type="KeysType"/>
4143
4144 <xs:complexType name="KeysType">
4145   <xs:sequence>
4146     <xs:element ref="md:KeyDescriptor"

```

```

4147         minOccurs="1"
4148         maxOccurs="unbounded" />
4149     </xs:sequence>
4150 </xs:complexType>
4151
4152 <!-- Service Metadata (SvcMD) - metadata about service instance -->
4153
4154 <xs:element name="SvcMD" type="SvcMetadataType" />
4155 <xs:complexType name="SvcMetadataType">
4156     <xs:sequence>
4157         <xs:element ref="Abstract" />
4158         <xs:element ref="ProviderID" />
4159         <xs:element ref="ServiceContext" maxOccurs="unbounded" />
4160     </xs:sequence>
4161     <xs:attribute name="svcMDID" type="xs:string" use="optional" />
4162 </xs:complexType>
4163
4164 <!-- ServiceContext - describes service type/option/endpoint context -->
4165 <xs:element name="ServiceContext" type="ServiceContextType" />
4166 <xs:complexType name="ServiceContextType">
4167     <xs:sequence>
4168         <xs:element ref="ServiceType" maxOccurs="unbounded" />
4169         <xs:element ref="Options" minOccurs="0"
4170             maxOccurs="unbounded" />
4171         <xs:element ref="EndpointContext" maxOccurs="unbounded" />
4172     </xs:sequence>
4173 </xs:complexType>
4174
4175 <!-- EndpointContext - describes endpoints used to access service -->
4176 <xs:element name="EndpointContext" type="EndpointContextType" />
4177 <xs:complexType name="EndpointContextType">
4178     <xs:sequence>
4179         <xs:element ref="Address" maxOccurs="unbounded" />
4180         <xs:element ref="sbf:Framework" maxOccurs="unbounded" />
4181         <xs:element ref="SecurityMechID" maxOccurs="unbounded" />
4182         <xs:element ref="Action" minOccurs="0"
4183             maxOccurs="unbounded" />
4184     </xs:sequence>
4185 </xs:complexType>
4186
4187 <!-- SvcMD ID element used to refer to Service Metadata elements -->
4188 <xs:element name="SvcMDID" type="xs:string" />
4189
4190 <!-- **** Discovery Service Protocol Messages Elements & Types **** -->
4191
4192 <!-- Query Message Element & Type -->
4193
4194 <xs:element name="Query" type="QueryType" />
4195
4196 <xs:complexType name="QueryType">
4197     <xs:sequence>
4198         <xs:element name="RequestedService"
4199             type="RequestedServiceType"
4200             minOccurs="0"
4201             maxOccurs="unbounded" />
4202     </xs:sequence>
4203
4204     <xs:anyAttribute namespace="##other" processContents="lax" />
4205 </xs:complexType>
4206
4207 <xs:complexType name="RequestedServiceType">
4208     <xs:sequence>
4209         <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
4210
4211         <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
4212
4213         <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
    
```

```

4214
4215     <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
4216
4217     <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
4218
4219     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
4220
4221     <xs:any namespace="##other"
4222         processContents="lax"
4223         minOccurs="0"
4224         maxOccurs="unbounded" />
4225
4226 </xs:sequence>
4227
4228 <xs:attribute name="reqID" type="xs:string" use="optional" />
4229 <xs:attribute name="resultsType" type="xs:string" use="optional" />
4230
4231 </xs:complexType>
4232
4233 <!-- QueryResponse Message Element & Type -->
4234
4235 <xs:element name="QueryResponse" type="QueryResponseType" />
4236
4237 <xs:complexType name="QueryResponseType">
4238     <xs:sequence>
4239         <xs:element ref="lu:Status" />
4240
4241         <xs:element ref="wsa:EndpointReference"
4242             minOccurs="0"
4243             maxOccurs="unbounded" />
4244     </xs:sequence>
4245     <xs:anyAttribute namespace="##other" processContents="lax" />
4246 </xs:complexType>
4247
4248
4249 <!--                                     -->
4250 <!-- DS Interfaces for SvcMD Associations      -->
4251 <!--                                     -->
4252 <!-- These interfaces support the adding, deleting, -->
4253 <!-- querying SvcMD Associations for a principal. -->
4254 <!--                                     -->
4255
4256 <!-- SvcMDAssociationAdd operation -->
4257
4258 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType" />
4259
4260 <xs:complexType name="SvcMDAssociationAddType">
4261     <xs:sequence>
4262         <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4263     </xs:sequence>
4264     <xs:anyAttribute namespace="##other" processContents="lax" />
4265 </xs:complexType>
4266 <!-- Response for SvcMDAssociationAdd operation -->
4267
4268 <xs:element name="SvcMDAssociationAddResponse"
4269     type="SvcMDAssociationAddResponseType" />
4270
4271 <xs:complexType name="SvcMDAssociationAddResponseType">
4272     <xs:sequence>
4273         <xs:element ref="lu:Status" />
4274     </xs:sequence>
4275     <xs:anyAttribute namespace="##other" processContents="lax" />
4276 </xs:complexType>
4277 <!-- SvcMDAssociationDelete operation -->
4278
4279 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType" />
4280

```

```
4281 <xs:complexType name="SvcMDAssociationDeleteType">
4282   <xs:sequence>
4283     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4284   </xs:sequence>
4285   <xs:anyAttribute namespace="##other" processContents="lax"/>
4286 </xs:complexType>
4287 <!-- Response for SvcMDAssociationDelete operation -->
4288
4289 <xs:element name="SvcMDAssociationDeleteResponse"
4290   type="SvcMDAssociationDeleteResponseType"/>
4291
4292 <xs:complexType name="SvcMDAssociationDeleteResponseType">
4293   <xs:sequence>
4294     <xs:element ref="lu:Status" />
4295   </xs:sequence>
4296   <xs:anyAttribute namespace="##other" processContents="lax"/>
4297 </xs:complexType>
4298 <!-- SvcMDAssociationQuery operation -->
4299
4300 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType"/>
4301
4302 <xs:complexType name="SvcMDAssociationQueryType">
4303   <xs:sequence>
4304     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4305   </xs:sequence>
4306   <xs:anyAttribute namespace="##other" processContents="lax"/>
4307 </xs:complexType>
4308 <!-- Response for SvcMDAssociationQuery operation -->
4309
4310 <xs:element name="SvcMDAssociationQueryResponse"
4311   type="SvcMDAssociationQueryResponseType"/>
4312
4313 <xs:complexType name="SvcMDAssociationQueryResponseType">
4314   <xs:sequence>
4315     <xs:element ref="lu:Status" />
4316     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4317   </xs:sequence>
4318   <xs:anyAttribute namespace="##other" processContents="lax"/>
4319 </xs:complexType>
4320
4321 <!--           -->
4322 <!-- DS Interfaces for Service Metadata Management -->
4323 <!--           -->
4324 <!-- These interfaces document a create, replace, -->
4325 <!-- delete, and query interface for the service -->
4326 <!-- metadata which is later associated with a -->
4327 <!-- principal. -->
4328 <!--           -->
4329
4330 <!-- Register operation for Service Metadata -->
4331
4332 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
4333
4334 <xs:complexType name="SvcMDRegisterType">
4335   <xs:sequence>
4336     <xs:element ref="SvcMD" maxOccurs="unbounded" />
4337   </xs:sequence>
4338   <xs:anyAttribute namespace="##other" processContents="lax"/>
4339 </xs:complexType>
4340
4341 <!-- Response for SvcMDRegister operation -->
4342
4343 <xs:element name="SvcMDRegisterResponse"
4344   type="SvcMDRegisterResponseType"/>
4345
4346 <xs:complexType name="SvcMDRegisterResponseType">
4347   <xs:sequence>
```

```

4348
4349     <xs:element ref="lu:Status" />
4350     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4351     <xs:element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
4352
4353 </xs:sequence>
4354 <xs:anyAttribute namespace="##other" processContents="lax"/>
4355 </xs:complexType>
4356
4357 <!-- Delete operation on Service Metadata -->
4358
4359 <xs:element name="SvcMDDelete" type="SvcMDDeleteType"/>
4360
4361 <xs:complexType name="SvcMDDeleteType">
4362 <xs:sequence>
4363 <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4364 </xs:sequence>
4365 <xs:anyAttribute namespace="##other" processContents="lax"/>
4366 </xs:complexType>
4367
4368 <!-- Response for delete operation on Service Metadata -->
4369
4370 <xs:element name="SvcMDDeleteResponse" type="SvcMDDeleteResponseType"/>
4371
4372 <xs:complexType name="SvcMDDeleteResponseType">
4373 <xs:sequence>
4374 <xs:element ref="lu:Status" />
4375 </xs:sequence>
4376 <xs:anyAttribute namespace="##other" processContents="lax"/>
4377 </xs:complexType>
4378
4379 <!-- Query operation on Service Metadata -->
4380
4381 <xs:element name="SvcMDQuery" type="SvcMDQueryType"/>
4382
4383 <xs:complexType name="SvcMDQueryType">
4384 <xs:sequence>
4385 <xs:element ref="SvcMDID"
4386 minOccurs="0"
4387 maxOccurs="unbounded" />
4388 </xs:sequence>
4389 <xs:anyAttribute namespace="##other" processContents="lax"/>
4390 </xs:complexType>
4391
4392 <!-- Response for Query operation on Service Metadata -->
4393
4394 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType"/>
4395
4396 <xs:complexType name="SvcMDQueryResponseType">
4397 <xs:sequence>
4398 <xs:element ref="lu:Status" />
4399 <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
4400 </xs:sequence>
4401 <xs:anyAttribute namespace="##other" processContents="lax"/>
4402 </xs:complexType>
4403
4404 <!-- Replace operation on Service Metadata -->
4405
4406 <xs:element name="SvcMDReplace" type="SvcMDReplaceType"/>
4407
4408 <xs:complexType name="SvcMDReplaceType">
4409 <xs:sequence>
4410 <xs:element ref="SvcMD" maxOccurs="unbounded" />
4411 </xs:sequence>
4412 <xs:anyAttribute namespace="##other" processContents="lax"/>
4413 </xs:complexType>
4414
    
```

```
4415 <!-- Response for SvcMDReplace operation -->
4416
4417 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType"/>
4418
4419 <xs:complexType name="SvcMDReplaceResponseType">
4420   <xs:sequence>
4421     <xs:element ref="lu:Status" />
4422   </xs:sequence>
4423   <xs:anyAttribute namespace="##other" processContents="lax"/>
4424 </xs:complexType>
4425
4426 </xs:schema>
4427
4428
```

4429 B. Discovery Service WSDL

```
4430
4431 <?xml version="1.0"?>
4432 <definitions name="disco-svc"
4433   targetNamespace="urn:liberty:disco:2006-08"
4434   xmlns:tns="urn:liberty:disco:2006-08"
4435   xmlns="http://schemas.xmlsoap.org/wsdl/"
4436   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4437   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4438   xmlns:sb="urn:liberty:sb:2006-08"
4439   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
4440   xmlns:disco="urn:liberty:disco:2006-08"
4441   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4442   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
4443     http://schemas.xmlsoap.org/wsdl/
4444     http://www.w3.org/2006/02/addressing/wsdl
4445     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
4446
4447 <!-- Abstract WSDL for Liberty Discovery Service v2.0 Specification -->
4448
4449   <xsd:documentation>
4450
4451     XML Schema from Liberty Discovery Service Specification.
4452
4453     ### NOTICE ###
4454
4455     Copyright (c) 2004-2006 Liberty Alliance participants, see
4456     http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
4457
4458   </xsd:documentation>
4459
4460 <types>
4461   <xsd:schema>
4462     <xsd:import namespace="urn:liberty:disco:2006-08"
4463       schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
4464   </xsd:schema>
4465 </types>
4466
4467 <message name="Query">
4468   <part name="body" element="disco:Query"/>
4469 </message>
4470 <message name="QueryResponse">
4471   <part name="body" element="disco:QueryResponse"/>
4472 </message>
4473
4474 <message name="SvcMDAssociationAdd">
4475   <part name="body" element="disco:SvcMDAssociationAdd"/>
4476 </message>
4477 <message name="SvcMDAssociationAddResponse">
4478   <part name="body" element="disco:SvcMDAssociationAddResponse"/>
4479 </message>
4480
4481 <message name="SvcMDAssociationQuery">
4482   <part name="body" element="disco:SvcMDAssociationQuery"/>
4483 </message>
4484 <message name="SvcMDAssociationQueryResponse">
4485   <part name="body" element="disco:SvcMDAssociationQueryResponse"/>
4486 </message>
4487
4488 <message name="SvcMDAssociationDelete">
4489   <part name="body" element="disco:SvcMDAssociationDelete"/>
4490 </message>
4491 <message name="SvcMDAssociationDeleteResponse">
4492   <part name="body" element="disco:SvcMDAssociationDeleteResponse"/>
4493 </message>
4494
```

```
4495 <message name="SvcMDRegister">
4496   <part name="body" element="disco:SvcMDRegister" />
4497 </message>
4498 <message name="SvcMDRegisterResponse">
4499   <part name="body" element="disco:SvcMDRegisterResponse" />
4500 </message>
4501
4502 <message name="SvcMDQuery">
4503   <part name="body" element="disco:SvcMDQuery" />
4504 </message>
4505 <message name="SvcMDQueryResponse">
4506   <part name="body" element="disco:SvcMDQueryResponse" />
4507 </message>
4508
4509 <message name="SvcMDReplace">
4510   <part name="body" element="disco:SvcMDReplace" />
4511 </message>
4512 <message name="SvcMDReplaceResponse">
4513   <part name="body" element="disco:SvcMDReplaceResponse" />
4514 </message>
4515
4516 <message name="SvcMDDelete">
4517   <part name="body" element="disco:SvcMDDelete" />
4518 </message>
4519 <message name="SvcMDDeleteResponse">
4520   <part name="body" element="disco:SvcMDDeleteResponse" />
4521 </message>
4522
4523
4524 <portType name="DiscoveryPort">
4525
4526   <operation name="DiscoveryQuery">
4527     <input message="tns:Query"
4528       wsaw:Action="urn:liberty:disco:2006-08:Query" />
4529     <output message="tns:QueryResponse"
4530       wsaw:Action="urn:liberty:disco:2006-08:QueryResponse" />
4531   </operation>
4532
4533   <operation name="MDAssociationAdd">
4534     <input message="tns:SvcMDAssociationAdd"
4535       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationAdd" />
4536     <output message="tns:SvcMDAssociationAddResponse"
4537       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationAddResponse" />
4538   </operation>
4539
4540   <operation name="MDAssociationQuery">
4541     <input message="tns:SvcMDAssociationQuery"
4542       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationQuery" />
4543     <output message="tns:SvcMDAssociationQueryResponse"
4544       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationQueryResponse" />
4545   </operation>
4546
4547   <operation name="MDAssociationDelete">
4548     <input message="tns:SvcMDAssociationDelete"
4549       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationDelete" />
4550     <output message="tns:SvcMDAssociationDeleteResponse"
4551       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationDeleteResponse" />
4552   </operation>
4553
4554   <operation name="MetadataRegister">
4555     <input message="tns:SvcMDRegister"
4556       wsaw:Action="urn:liberty:disco:2006-08:SvcMDRegister" />
4557     <output message="tns:SvcMDRegisterResponse"
4558       wsaw:Action="urn:liberty:disco:2006-08:SvcMDRegisterResponse" />
4559   </operation>
4560
4561   <operation name="MetadataQuery">
```

```
4562     <input message="tns:SvcMDQuery"
4563         wsaw:Action="urn:liberty:disco:2006-08:SvcMDQuery" />
4564     <output message="tns:SvcMDQueryResponse"
4565         wsaw:Action="urn:liberty:disco:2006-08:SvcMDQueryResponse" />
4566 </operation>
4567
4568 <operation name="MetadataReplace">
4569     <input message="tns:SvcMDReplace"
4570         wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplace" />
4571     <output message="tns:SvcMDReplaceResponse"
4572         wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplaceResponse" />
4573 </operation>
4574
4575 <operation name="MetadataDelete">
4576     <input message="tns:SvcMDDelete"
4577         wsaw:Action="urn:liberty:disco:2006-08:SvcMDDelete" />
4578     <output message="tns:SvcMDDeleteResponse"
4579         wsaw:Action="urn:liberty:disco:2006-08:SvcMDDeleteResponse" />
4580 </operation>
4581
4582 </portType>
4583
4584 <!--
4585 An example of a binding and service that can be used with this
4586 abstract service description is provided below.
4587 -->
4588
4589 <binding name="DiscoveryBinding" type="tns:DiscoveryPort">
4590
4591     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
4592
4593     <operation name="DiscoveryQuery">
4594         <soap:operation soapAction="urn:liberty:disco:2006-08:Query" />
4595         <input> <soap:body use="literal"/> </input>
4596         <output> <soap:body use="literal"/> </output>
4597     </operation>
4598
4599     <operation name="MDAssociationAdd">
4600         <soap:operation
4601             soapAction="urn:liberty:disco:2006-08:SvcMDAssociationAdd" />
4602         <input> <soap:body use="literal"/> </input>
4603         <output> <soap:body use="literal"/> </output>
4604     </operation>
4605
4606     <operation name="MDAssociationQuery">
4607         <soap:operation
4608             soapAction="urn:liberty:disco:2006-08:SvcMDAssociationQuery" />
4609         <input> <soap:body use="literal"/> </input>
4610         <output> <soap:body use="literal"/> </output>
4611     </operation>
4612
4613     <operation name="MDAssociationDelete">
4614         <soap:operation
4615             soapAction="urn:liberty:disco:2006-08:SvcMDAssociationDelete" />
4616         <input> <soap:body use="literal"/> </input>
4617         <output> <soap:body use="literal"/> </output>
4618     </operation>
4619
4620     <operation name="MetadataRegister">
4621         <soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDRegister" />
4622         <input> <soap:body use="literal"/> </input>
4623         <output> <soap:body use="literal"/> </output>
4624     </operation>
4625
4626     <operation name="MetadataQuery">
```

```
4629     <soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDQuery" />
4630     <input> <soap:body use="literal" /> </input>
4631     <output> <soap:body use="literal" /> </output>
4632 </operation>
4633
4634 <operation name="MetadataReplace">
4635     <soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDReplace" />
4636     <input> <soap:body use="literal" /> </input>
4637     <output> <soap:body use="literal" /> </output>
4638 </operation>
4639
4640 <operation name="MetadataDelete">
4641     <soap:operation soapAction="urn:liberty:disco:2006-08:SvcMDDelete" />
4642     <input> <soap:body use="literal" /> </input>
4643     <output> <soap:body use="literal" /> </output>
4644 </operation>
4645
4646 </binding>
4647
4648 <service name="DiscoveryService">
4649
4650     <port name="DiscoveryPort" binding="tns:DiscoveryBinding">
4651
4652         <!-- Modify with the REAL SOAP endpoint -->
4653
4654         <soap:address location="http://example.com/discovery" />
4655
4656     </port>
4657
4658 </service>
4659
4660 </definitions>
4661
4662
4663
```